A Constraint Based Approach to Interleaving Planning and Execution for Multirobot Coordination

Mary Koes

mberna@cs.cmu.edu

The Robotics Institute
Carnegie Mellon University

December 14, 2005

Thesis Committee:
Katia Sycara, co-chair
Illah Nourbakhsh, co-chair
Anthony Stentz
Edmund Durfee, University of Michigan

Abstract

Enabling multiple robots to work together as a team is a difficult problem. Robots must decide amongst themselves who should work on which goals and at what time each goal should be achieved. Since the team is situated in some physical environment, the robots must consider travel time in these decisions. This is particularly challenging in time critical domains where goal rewards decrease over time and for tightly coupled coordination where multiple robots must work together on each goal. Further complications arise when the system is subjected to additional constraints on the ordering of the goals, the use of resources, or the allocation of robots to goals. Optimal team behavior can only be achieved when robots simultaneously consider path planning, task allocation, scheduling, and these additional system constraints. In dynamic and uncertain environments, robots need to reevaluate these decisions as they discover new information. Communication failures may mean that robots are unable to consult as a whole team while replanning. The proposed thesis addresses these challenges with four main points.

First, we propose modeling the problem facing the robot team, a time critical tight coordination team planning problem, using a mathematical programming model, or, more specifically, as a mixed integer linear programming problem. We describe algorithms for combining traditional solution techniques for mathematical programming models with domain specific heuristics resulting in a system that is declarative, transparent to humans, efficient, progressive, and capable of providing error bounds on solution quality.

Second, we show that it is possible to reason about a variety of constraints on the goals, robots, and resources of the system within the proposed framework. We describe a general first order logic constraint language for multirobot coordination which we incorporate into our system.

Third, the proposed thesis provides a framework for interleaving planning and execution in a dynamic environment. By adopting a novel model for communication failure in robot teams, robots can explicitly reason about the effects on team performance of changes in the team plan when some of the team is out of communication and unaware of these changes. The tradeoff between adapting the plan to optimally respond to new information and preserving the original plan for consistency can be incorporated into our proposed mixed integer linear programming model in a principled manner, thus improving team performance.

Similarly, a number of various tradeoffs can be represented in the declarative mathematical programming model. We propose addressing one more such tradeoff and reasoning about the robustness of team plans to certain types of failure or new information.

Preliminary results in an abstract simulator indicate the feasibility of this approach and the benefits of some of these extensions. The proposed thesis will further investigate the effects of reasoning about constraints, communication failures, and plan robustness on a variety of benchmarks and on realistic robots in a high fidelity simulator.

Contents

1 Introduction

Multirobot teams are useful in environments in which it is dangerous, expensive, or impractical to send humans. A number of such target application domains have been identified including reconnaissance [22], gallery monitoring and security sweeps [45], space exploration [67], and the inspiration domain for this work, urban search and rescue (USAR) [58]. The robots must work together to accomplish a set of goals shared by the team. The allocation and scheduling of robots to goals is a *team planning problem*. We are interested in a number of challenges which arise in USAR and other complex domains when solving the team planning problem.

Complex domains may require multiple robots to collaborate on a single *joint goal* if no individual robot is capable of performing the goal independently or when team performance may be improved by this collaboration. A simple example from the USAR domain involves one robot with a spotlight and a second robot with a high resolution camera working together to check a dark ceiling for structural stability.

Time critical domains such as search and rescue, where lives are at stake, require careful consideration of plan efficiency. For teams of robots operating in physical environments, coordination inefficiencies may by magnified by the spatial constraints inherent to the problem. The time needed for robots to traverse the environment often dominates team performance. Consequently, it is beneficial to dedicate some time to generating an optimal or near optimal plan rather than executing a highly suboptimal plan. Since planning itself delays execution, it is important that this planning be efficient.

Search and rescue and other rich domains include a wide variety of additional constraints on the goals, robots, and resources. A coordination architecture for these domains must be able to accommodate these constraints. Some constraints arise naturally from the problem, for example, *a robot may not travel further than its battery capacity allows.* Other system constraints facilitate human robot team interaction by allowing human responders more direct control over system performance than if the only input is goal rewards as illustrated by the simple but important example of *turn off the circuit breakers before completing any other goals.* Finally, system constraints may improve efficiency through more expressive modeling.

Even when robots are able to start with an optimal plan, few domains are so benign that this original plan can be successfully executed. As robots fail, new robots join the team, the environmental model changes, goals are added and removed, etc., the team must react appropriately, interleaving planning and execution. Due to the physical nature of the environment, robots may not be in full communication at any given time. This communication failure necessitates distributed replanning.

1.1 Objective

The proposed thesis seeks to answer the question,

> *Can agents improve* ***team performance*** *in* ***time critical*** *domains with* ***tight coordination*** *by reasoning about the* ***heterogeneous capabilities*** *of team members,* ***system constraints, communication failures*** *and* ***expected replanning catalysts****?*

A **team** consists of a set of members that share a set of goals. The team composition may change over time as members are added and removed from the team. Similarly, the goals of the team may change over time as new goals are added and old goals are met or discarded. We limit our research to teams in which individual team members have no selfish goals so that all actions are for the good of the team.

In **time critical** domains, team performance is not only dependent on which goals are accomplished but when these goals are accomplished. Goal rewards are functions of time. Search and rescue is a natural example of a time critical domain since victims are more likely to die as time passes.

Tight coordination between robots is required when a subset of the team goals are best achieved by multiple robots working together on an individual goal. This is in contrast to loosely coupled coordination in which all goals may be independently accomplished by individual robots [24] [18].

We use **agent** to refer to the members of the team. In general, an agent is a computational entity with some degree of autonomy and intelligence capable of interacting with other agents. We comprehend both software and physical agents (robots) in the term. As a software agent may be considered a robot with no physical capabilities, we will use the terms robot and agent interchangeably.

The central question stated leads to several subsidiary questions:

- *Can we create an efficient planning system with optimality guarantees for time critical tightly coupled team planning problems?*

  Before improving team performance through intelligent reasoning, we need a planning system. This system should have the following properties.

  Efficiency The system must be able to generate feasible plans quickly (in a few seconds).

  Optimality The coordination framework should converge to the optimal solution.

  Progressive Given additional time, the planner should improve the solution quality.

  Error bounds The planner should be able to generate a solution that is guaranteed to be within some margin of the optimal solution or, alternatively, specify the maximum distance from optimality of a generated solution.

  Transparency The coordination framework should be transparent to humans interacting with the system. Robots should be able to explain their actions in terms of the specified goals of the system. Although we consider transparency in our design criteria, the topic of human-robot interaction is an interesting problem but outside the scope of the proposed thesis.

- *How should robots reason about heterogeneity in time critical tight coordination team planning problems?*

  Robot teams in challenging environments such as a disaster site will necessarily have **heterogeneous capabilities** as cost limitations, power consumption, and size constraints require tradeoffs between mobility and capabilities. The environment may require robots with diverse modalities (e.g. ground, air, water) to successfully navigate the environment. When certain resources are scarce (i.e. only one robot has a particular sensor demanded by many goals), explicitly reasoning about team heterogeneity may improve performance.

- *What system constraints should be supported by the planning system and how can they be incorporated in a principled manner?*

  System constraints most generally include any restrictions on the solution space. In the context of this research, system constraints may include goals, robots, or system resources. These constraints may arise naturally from the environment such as constraints on how far a robot may travel. Alternatively, they may arise from complex goal models as commonly found in the planning literature where the goals can be accomplished in a number of different possibilities. System constraints may also arise from the arbitrary preferences of the user specifying the goals. For example, in the search and rescue domain, a rescue worker may restrict aerial robots from working on goals in some sector due to security concerns. These system constraints should be explicitly considered during planning to guarantee that they are all met and to maximize system performance.

- *How should robots react to communication failures in order to maximize expected team performance?*

  Communication failures complicate coordination efforts when agents are unable to share information which may affect team performance. A centralized approach to coordination is infeasible if some subset of the team is unable to communicate with a master planner. The inconsistencies in team beliefs introduced by these communication failures may negatively affect team performance unless these inconsistencies are considered while replanning.

- *Can robots use models for robot failure, communication failure, etc. to create plans that are robust to these failures and so increase team performance?*

  Replanning catalysts are events that necessitate replanning. Examples include the failure of a robot, the addition of a new goal, or the discovery that it will take twice as long to travel to a goal as expected. As each event is discovered, the previous plan must be refined to accommodate the new information. In extreme cases, this may result in system thrashing. To avoid this behavior, the system can prefer plans with robustness to the detection of certain expected replanning catalysts. For example, if robots frequently fail, it may be best to initially send two robots to accomplish one goal.

1.2 Approach

This thesis proposal addresses these problems in the context of constraint optimization. Our approach is to build a mathematical programming model of the time critical tight coordination team planning problem. More specifically, we model the problem as a mixed integer linear programming problem. This approach has a number of advantages. The declarative representation of the mathematical model preserves the original semantics of the problem and thus the transparency. The heterogeneous capabilities of the robots, spatial and temporal constraints inherent to the problem, and additional system constraints can be represented in this model. This approach also allows the flexibility to combine multiple objective functions in a principled way which facilitates advanced reasoning on communication failures and expected replanning catalysts. We are also able to leverage the extensive work of the operations research community in efficient solution techniques.

We have developed a **C**onstraint **O**ptimization **Co**ordination **A**rchitecture, COCoA, which maps the team planning problem to a mixed integer linear programming problem and then combines domain specific heuristics and standard integer programming algorithms to efficiently generate optimal or near optimal plans.

As previously discussed, since the robots are operating in a dynamic, uncertain environment, planning and execution must be interleaved. The communication failures common to robot systems depending on a wireless network may negatively affect team performance during replanning. Although this communication failure is a common phenomenon in multirobot coordination, models for these communication breakdowns are relatively primitive. Multirobot coordination architectures that address the problem assume any given message is lost with some uniform probability. However, this is unrealistic since robot communication is frequently based on TCP which is a reliable protocol. For robots using wireless Ethernet for communication, there is a finite range for communication that is determined by the properties of the environment and the power of the wireless signal. We propose modeling these limited ranges of reliable communication which result in subsets of the team with good local communication but little or no communication between subsets. In this thesis proposal, we use the term *fractured subteam* to refer to subsets of robots in the original team that are able to communicate with each other, directly or indirectly, but not with the rest of the team.

While the nature of tight coordination problems does not easily allow robots to take advantage of distributed planning [24], the fractures in the team make centralized planning infeasible forcing distributed replanning. We propose a hybrid approach for coordination in which robots within a fractured subteam replan centrally in order to efficiently generate near optimal plans but each individual robot is capable of independently responding to deviations in the plan. Coordination within a fractured team is challenging because the fractured subteams introduce a new dynamic variable into the system which must be considered while replanning. Since other subteams are unaware of disturbances within the system, care must be taken to minimize disruption to the schedules of these other robots.

The final point in our approach is to plan for the eventuality of replanning catalysts by biasing the system towards plans that are robust to these events. This has not yet been implemented. We hope that by incorporating models of these events into the objective function of the mixed integer linear programming problem team performance will be improved, provided the models for the replanning catalysts match the actual occurrence of these events.

1.3 Contributions

The proposed thesis would provide the following contributions to the robotics community:

Multirobot coordination through constraint optimization The mathematical programming model of the coordination problem described in this thesis proposal is a novel approach to multirobot coordination. We also describe an algorithm for combining traditional constraint optimization solution techniques with domain specific approaches such as market based coordination.

Time critical tight coordination team planning problem This thesis proposal formalizes and analyzes the time critical tight coordination team planning problem. We present a set of benchmarks that illustrate the complexities of the problem space. We empirically compare various algorithms on these benchmarks.

System constraints The first order logic constraint language described in this thesis proposal captures the important aspects of many multirobot coordination domains. We provide an implementation for this constraint language. The proposed thesis will also analyze the effect of various system constraints on team performance.

Communication failure In this thesis proposal we describe a new model for communication failure in spatially distributed robot teams. These communication failures result in *fractured subteams* and necessitate a distributed approach to coordination. We describe a hybrid system with distributed replanning with opportunistic centralization. We present an empirical analysis of communication failures on team performance.

Selective disruption minimization This thesis proposal describes selective disruption minimization as a method for improving team performance by attempting to avoid changes to schedules of certain robots on the team in order to reduce the cognitive load of humans interacting with those robots or to avoid inconsistent plans due to communication failures. We show that selective disruption minimization can be incorporated in the mathematical programming model of the team planning problem in a principled manner.

Robust plan selection The proposed thesis will describe a method for selecting plans that are robust to certain expected types of failure by, for example, allowing robots extra time to travel to goals or allocating surplus robots to a goal in case one robot fails. We will analyze the effect of robust plan selection on team performance both when the actual failures match the model of expected failures and when the expected failures do not occur.

1.4 Organization of proposal

The remainder of this document is organized as follows. In the next section, we summarize and evaluate related work in the fields of multirobot and multiagent systems. In Section 3, we describe COCoA and the already completed steps towards addressing the questions posed by the proposed thesis. We outline our proposed path to answering the rest of the questions in Section 4 and the expected schedule for this work in Section 5. The contributions of the proposed thesis are summarized in Section 6.

2 Related Work

Our work draws on research in a number of fields but particularly from the multirobot and multiagent research communities. Although many of the goals of these two communities are identical at a high level, there are a number of issues in designing a coordination architecture for physical robots that may be ignored with software agents. For example, robots must reason about both spatial and temporal constraints when coordinating which are not generally relevant in coordinating software agents. Consequently the two groups have developed different ideologies and methodologies.

2.1 Multirobot coordination

A wide range of application domains for multirobot coordination coupled with recent advances in single robot control have motivated much research in robot teamwork in recent years, a trend that is likely to continue. We distinguish between *implicit* coordination in which complex collective behavior emerges from the interactions of the set of agents following simple yet carefully crafted rules or behaviors and *explicit* coordination in which robots coordinate their actions intentionally.

Implicit Coordination

One approach to coordinating the actions of many robots is to allow the robots to self organize, such as animals in a swarm. By having each robot follow simple rules, macroscopic behavior emerges. The swarm approach seeks to have a set of simple agents with completely distributed control. The individual agent behavior is designed such that global behavior emerges from local interactions among agents and the environment. Appropriately, this approach is often termed *emergent coordination.*

This biologically inspired approach requires the system designer to determine exactly what each robot should do and hard code this behavior into the robot intelligence. One advantage is that the robots then need relatively little of their own intelligence. Furthermore, this tactic works well in biology where simple creature such as termites have evolved to accomplish complex feats such as building termite mounds following simple rules. Several researchers have successfully demonstrated this technique for collective intelligence in domains such as foraging, flocking, and predator-prey relationships [54] [5] and carrying a box between multiple robots [13].

Although self organizing approaches to coordination are in many ways an elegant solution to multirobot coordination, there are a number of drawbacks that make it inappropriate for the time critical tight coordination team planning problem. First, it is difficult and, in some domains, impossible to design distributed agents and their actions such that local reactions result in goal achievement. Second, these systems lack any explicit notion of teamwork which is a drawback when the system must interact with humans. Finally, since complex behaviors are not explicitly programmed in but rather emerge from the complex interactions of robots, guaranteeing system performance in emergent systems is usually not possible. Mathematically, the macroscopic behavior of the system may be analyzed by modeling the individual agents as stochastic and Markovian, as shown by Lerman and Galstyan, [47]. This method requires every relevant state of the system and the transitions between these states to be modeled by a variable and a rate equation that describes how that variable changes over time. This is impractical for complex tasks or controllers and cannot take into account memory, learning, or decision making by agents. While emergent behavior can be robust to certain types of environmental dynamics if the controllers are properly designed, it is difficult to infuse knowledge about a specific environment into the system (i.e. avoid a specific corridor because it is blocked off).

Explicit Coordination

In contrast to implicit coordination, under explicit coordination robots can intentionally cooperate to achieve a goal. The popularity this approach and diversity of proposed solutions has led several researchers to propose taxonomies to characterize the various approaches [28], [4], [35], [87]. The taxonomy described by Gerkey and Matarić [35], though incomplete since it only addresses the multirobot task allocation problem, is helpful in understanding the problem space and roughly characterizing several approaches. This taxonomy focuses on the problems addressed rather than the particular solution. The taxonomy analyzes coordination approaches based on whether the robots are capable of performing one (ST) or more (MT) tasks at a time, whether goals require exactly one (SR) or more (MR) robots to be achieved, and whether task allocations are instantaneous (IA) or include time-extended scheduling (TA). The first axis in the taxonomy is relatively uninteresting since few robots are capable of performing multiple tasks simultaneously and the solutions for single task robots extend trivially to multitask robots. However, we examine work in multirobot coordination in terms of the other two axes of the taxonomy. The time critical tight coordination team planning problem addressed by the proposed thesis falls into the single task, multirobot, time extended category ST-MR-TA.

A number of systems address the ST-SR-IA problem. Werger and Matarić's BLE system uses a greedy algorithm and behavior based paradigm for coordination [83]. In Parker's ALLIANCE system [59], robots maintain levels of impatience and acquiescence concerning all available tasks. The computational load of individual robots is increased since they must model all other robots. Correct tuning of the parameters (L-ALLIANCE) can increase system performance. MURDOCH [33], GOFER [14], and M+ [9] also address ST-SR-IA problems using an extension of the contract net protocol [69].

In the robot soccer domain, a number of approaches have been developed for the closely related problem of role allocation. A unique approach to coordination is through the generation of artificial potential fields. Potential fields are well known for obstacle avoidance since they can be precomputed and then are relatively inexpensive to use and repair, even when the environment changes. Potential fields have the general drawback that they may guide the robots to local extrema. As demonstrated in the robot soccer domain by Vail and Veloso [80][79] and Tews and Wyeth [77], potential fields can also be used to determine the actions of the individual robots. Defining the potential field may be difficult for complex domains.

Gerkey and Matarić show that MT-MR-IA problems are instances of the set covering problem [35] which is $\mathcal{NP}$ hard. Tang and Parker present ASyMTRe [75] (and a distributed version ASyMTRe-D [76]) with an anytime algorithm for solving this class of problems. This class of problems is also related to the coalition formation problem in multiagent systems, though coalition formation has an additional set of problems as discussed in section 2.2. This class of problems does not deal with scheduling or the time critical nature of the problem addressed by the proposed thesis. The Hoplites framework [45] may also be classified in the class of MT-MR-IA problems though the Hoplites framework was developed for a more specific class of tight coordination problems than addressed by ASyMTRe or the proposed thesis which abstracts away the tight coordination of robots actually working together on a goal. Gerkey, Thrun, and Gordon recently proposed Parallel Stochastic Hill Climbing or Parish [32] which combines stochastic action selection with simple heuristics to solve the multi-robot pursuit evasion problem, an instance of ST-MR-IA class of problems.

The gaining popularity of market based approaches within the robotics community warrants a closer look at this method for coordination. Market frameworks are based on the contract net protocol [69] in which rational agents submit bids based on the cost of completing a task and receive some reward for successfully completing a task. This is an efficient method for coordinating teams of robots and has been demonstrated effectively in a number of systems [25][34][18][33][14][9]. TraderBots [22], is one of the most successful of these systems and supports some complex tasks [89], limited opportunistic centralization [25], heterogeneous robots and time varying rewards [67]. Despite these successes, market based approaches have some limitations. Dias et. al. write in [24] *Perhaps the biggest drawbacks of market-based approaches currently are the lack of performance guarantees and the lack of formalization in designing appropriate cost and revenue functions to capture domain requirements.* This lack of performance guarantees is unacceptable in certain time critical domains. Designing appropriate cost and revenue function can be a challenging problem as noted above. The opaqueness of these functions make it difficult for humans to understand the system. Furthermore, market systems typically address only the task allocation problem and not when tasks should be scheduled.

Summary of multirobot coordination

The set of problems requiring multiple robots on goals and time extended assignment is not adequately addressed in the multirobot literature. This thesis proposal addresses this class of problems. Not included in the taxonomy is the ability of a system to meet various constraints on goals and resources which this proposal also addresses. We classified work in the multirobot community into implicit and explicit coordination. In reality, however, most of the systems mentioned under explicit coordination represent hybrid approaches to coordination. For example, in a market framework, robots do not have an explicit team model. Instead, the system designers carefully craft local cost and reward functions and coordination emerges from the interactions of the robots through auctions.

2.2 Multiagent coordination

Work in the multiagent systems community spans a broad range of application domains. The breadth of the research is perhaps due to the definition of an agent as any computational entity with some degree of autonomy and intelligence capable of interacting with other agents. We limit our investigation of the field primarily to cooperative agents. Tambe et. al. [74] identify four competing approaches to teamwork within the multiagent community: belief-desire-intention (BDI), distributed constraint optimization (DCOP), distributed POMDPs, and auctions or game theoretic approaches.

Belief-desire-intention

The multiagent system community has addressed a number of problems that are fundamental to this research including what it means to be a team and the roles of commitments, conventions, and accountability in teams. Bratman proposed the belief-desire-intention (BDI) model in 1987 [12]. Beliefs represent the agent's information about the current state of the world. Desires represent the agent's ideal state of the world. Intentions represent desires that the agent plans to realize. The BDI model is founded on theories of rational action in humans and so may enable better transparency for humans.

There are a number of formal models for teamwork. Bratman provides three requirements for teamwork or shared cooperative activity including mutual responsiveness, mutual commitment to a joint activity, and mutual support [11]. These characteristics of teamwork motivate the approach adopted in this thesis proposal specifically the selective disruption minimization. A number of teamwork theories have captured these characteristics including SharedPlans [38], joint intentions [19], and joint responsibility [42]. These teamwork theories have been implemented in a number of systems including STEAM (joint intentions and SharedPlans) [73], GRATE (joint responsibility) [44], and SPIRE (SharedPlans) [71].

Partial Global Planning (PGP) [29] and Generalized Partial Global Planning (GPGP) and the related TÆMS framework [48] use approximation algorithms to optimize team performance in a distributed soft real-time fashion. Goals are represented in an extended AND/OR tree. Agents communicate their high level goals and update their models of other agents and then seek to optimize their own local activities with some consideration of the activities of other agents using a hill climbing procedure. These frameworks do not address the task allocation problem though they provide good solutions to coordinating the schedules of agents sharing resources.

Jennings and Campos proposed the idea of socially responsible agents in 1997 [43]. They define a Social Level of abstraction which sits immediately above Allen Newell's Knowledge Level. Socially responsible agents retain their local autonomy but interact with, and are inherently linked to, the larger community. This approach differs from purely selfless agents who only consider helping their fellows ([15], [1]). Socially responsible agents demonstrate mathematically and empirically that having an agent reason about the global cost and reward of its actions as well as individual cost and reward results in improved system performance. This work is very abstract, however, and does not provide for situations where the robot may have a set of choices or a continuum of choices. Although it has been somewhat expanded upon([70], [26], [27]), it has not been used on real robots or in complex domains.

Distributed constraint optimization

Distributed constraint reasoning concerns itself with problems in which knowledge about the variables and constraints in the problem is distributed among many agents. There are a number of reasons why the distributed problem may not be trivially reduced to a centralized problem.

Privacy One of the most compelling reasons for distributed algorithms is that security or privacy concerns may make the sharing of private information undesirable.

Communication costs Sometimes the costs of translating an agent's representation of the problem into an exchangeable format and communicating this information is prohibitively high.

Efficiency Finally, the structure of some problems allows a more efficient solution through the exploitation of distributed computing resources.

The class of constraint satisfaction problems (CSPs) deals with finding legal values for a set of variables. Each variable V_i has has a set of possible values or domain, D_i. In general, a domain may be discrete or continuous. Unary constraints restrict the allowable values in the domain of a single variable while binary constraints relate pairs of variables [64].

The field of distributed constraint satisfaction concerns itself with CSPs in which the variables and constraints are distributed among a set of autonomous agents. Yokoo et. al. present a formulation for the problem in which domains are finite and discrete, each agent has exactly one variable, all constraints are binary, and each agent knows all the constraints relevant to its variable [88]. They present an asynchronous backtracking algorithm for this problem formulation which can be extended to allow each agent to have an arbitrary number of variables.

Problems in which the emphasis is in finding optimal rather than feasible solutions may require distributed constraint optimization (DCOP). Like CSPs, constraint optimization problems consist of a set of variables, each with a domain D_i. However, binary constraints now represent the *cost* of an allocation between pairs of variables. Constraint optimization algorithms seek to minimize the overall system cost. Like work in distributed constraint satisfaction, distributed constraint optimization problem formulations assume that domains are finite and discrete, each agent has exactly one variable, all constraints are binary, and each agent knows the constraints relevant to its variable. The number of DCOP algorithms developed over the years attests to the difficulty and popularity of DCOP problems which are $\mathcal{NP}$-complete [56].

Hirayama and Yokoo developed Synchronous Branch and Bound [39] which has the drawback that execution must be synchronized and sequential. ADOPT was developed by Modi et. al. [56] to address the synchronization problem. OptAPO by Mailler and Lesser [53] uses partial centralization or cooperative mediation. An agent is dynamically appointed as the mediator for some subset of the problem but the agents try to minimize the number of centralized constraints. Petcu and Faltings have developed DPOP which uses dynamic programming to improve performance [61].

These algorithms have been successfully demonstrated in several applications including meeting scheduling [55], distributed sensor networks [57], and to the multiagent plan coordination problem (MPCP) [21]. There are a number of obstacles in extending the work to the time critical tight coordination team planning problem of interest in this thesis proposal, however. Existing work in DCOP lacks support for real valued variables which is a problem for robots which operate in an environment that is continuous in both space and time. Discretization can lead to inefficiencies and intractably large problem sizes. More importantly, the much of the motivation for distributed constraint reasoning is lacking in multirobot coordination. There is no need for privacy in a team of robots only working towards the common good. The time critical tight coordination team planning problem can be represented and communicated compactly. In fact, since robots may lose communication with their teammates and need to reason about their commitments, it is desirable to share all the constraints and the team plan initially. Finally, argument for efficiency does not apply to this problem. DCOP is well suited for loosely coupled problems (such as MPCPs) but not for tight coordination. Even small time critical tight coordination team planning problems required hours to find a feasible solution in tests using ADOPT. Centralization becomes more advantageous as degree of interagent dependencies increases. By first centralizing the problem, we are able to take advantage of more sophisticated algorithms than branch and bound search. In particular, we use both the linear programming algorithms and integer linear programming algorithms in the commercially available solver, CPLEX [40] which significantly improves efficiency in team plan generation.

In addition to optimal solution techniques, several heuristic techniques for solving DCOP problems have been proposed. Token-based algorithms have been demonstrated in complex domains including search and rescue [66] and UAV coordination [85]. LA-DCOP [66] is a task allocation algorithm for large scale teams that attempts to maximize the team's expected utility given probabilistic information by computing a minimum capability threshold for each task. The problem formulation proposed by LA-DCOP interleaves scheduling and task allocation and does not explicitly consider path planning during task allocation but instead incorporates travel time considerations with other fitness criteria into a single axis for robot heterogeneity. Furthermore, the assumptions underlying the probabilistic representation of team capabilities may break down in domains with fewer robots decreasing team performance. Token-based algorithms also fail to provide the performance guarantees necessary for time critical domains as they do not consider path planning and separate planning and scheduling, leading to inefficiencies.

Distributed POMDPs

Markov Decision Problems (MDPs) describe the problem of choosing an optimal policy in a stochastic environment where the agent has enough knowledge to determine its state and all transition models at all times. MDPs assume that the world has the Markov property meaning that the transition models from any given state depend only on the state and not the previous history [64]. If the agent is unable to directly observe the world state and must instead rely on noisy observations of the environment to estimate the current world state, the resulting problem is a Partially Observable Markov Decision Problem (POMDP) [17]. Both MDPs and POMDPs deal with finding optimal policies for a single agent. These frameworks can be extended to multiple agents with full observability using the Multi-Agent Markov Decision Problem (MMDP) framework [10], multiple agents with collectively full observability using the Decentralized Markov Decision Problem (DEC-MDP) framework [8], and multiple agents with collective partial observability with the Decentralized Partially Observable Markov Decision Problem (DEC-POMDP) framework [8], Markov Team Decision Problem (MTDP) framework [62], or the Partially Observable Identical Payoff Stochastic Game (POIPSG) framework [60]. Pynadath and Tambe provide the COM-MTDP model for reasoning about the cost of communication [62] but do not explicitly model communication failure.

While the problem of planning in uncertainty is highly relevant to robotic systems, the additional complexity of these approaches (DEC-POMDPs are NEXP-complete [8]) make them unsuitable for many domains particularly when accurate models for the uncertainty are unavailable. Our approach is more analogous to the human method for planning where the agents form a plan using the expected state of the world and then react to dynamic events rather than computing a policy for all possible world states. Furthermore, the assumptions that the world has the Markov property and a finite number of states do not easily apply to the problem of interest since robots operate in a continuous time domain with time varying reward functions and rewards depend on the order that the goals are accomplished.

Auctions and game theoretic approaches

The research of the multirobot and multiagent communities has mingled in developing market based systems as an efficient means for coordination. Though there is considerable overlap between the problems addressed by each group, robots and agents each present a unique set of challenges which guide the design of market based systems. Robots interact with a physical environment where performance is often dominated by locomotion challenges. Spatial constraints, more limited communication, and uncertainty are some of the challenges that drive multirobot research. Agents are often self interested, come from different designers, interact in an open system, and have privacy, truthfulness, and security concerns. Since our problem is a subset of the multirobot coordination problem, we focus our attention on this line of research which we addressed under *Explicit Coordination* in Section 2.1. Here we discuss a very limited subset of the research in the multiagent community in this area.

The field of coalition formation in the multiagent systems domain is related to the ST/MT-MR-IA class of problems. Shehory and Kraus ([68]) present distributed anytime algorithms with error bounds in the multiagent domain. The coalition formation problem does not address reasoning about spatial constraints or time varying reward, though it must address another set of difficult problems such as trust, fairness, etc. Since multiagent systems are concerned with rational agents, they do not explicitly maximize team utility but instead use market mechanisms to promote stability and efficiency [50]. In contrast the approach in this thesis proposal has no model of individual robot utility but explicitly maximizes team utility.

Combinatorial auctions allow buyers to express the value of winning a set of items which may be greater than the sum of the individual items in which case the auction goods are called *complementary*. This is frequently illustrated in the multirobot field where it is cheaper to accept goals that are located close to one another. When agents are permitted to bid on bundles of goods in an auction, deciding who wins what, known as the clearing problem for combinatorial auctions, is $\mathcal{NP}$-complete an inapproximable [65]. One solution is to allow goods to be reauctioned such as in the TRACONET system developed by Sandholm [65]. Systems that allow both combinatorial auctions and coalition formation have been demonstrated in the multiagent community [86][49].

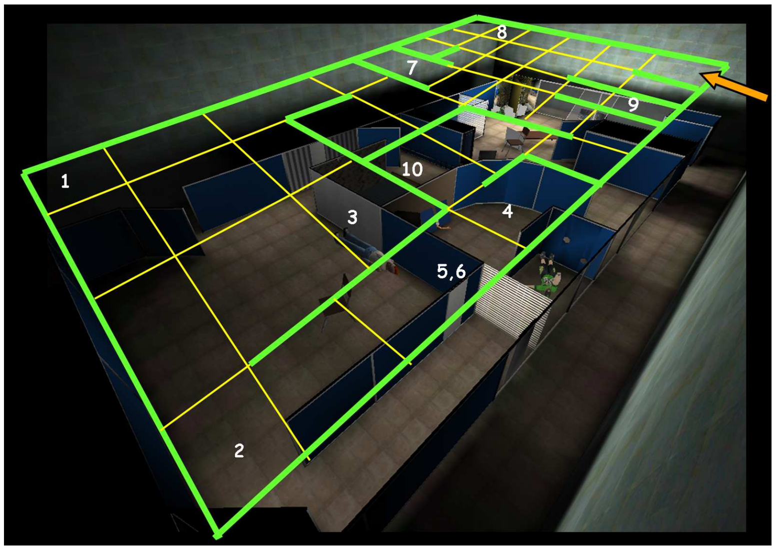
3 Completed Work

The motivation for the proposed thesis is illustrated in the following example. When disaster strikes an urban area, rescue workers have 48 hours to find trapped victims before the probability of survival becomes essentially zero. This high pressure situation is further complicated by the risk to the rescue workers of entering a collapsed and possibly unstable structure. Employing robots to search parts of the structure has the potential to save both rescue workers' and victims' lives. Since robots are "disposable," they can explore areas too risky for humans, mitigating the risk to rescuers and decreasing the search time. They can also be designed to fit through spaces too small for humans or dogs. Let us suppose we have a set of robots at an urban search and rescue (USAR) disaster site.

Although with feelings running high, different people may have different opinions of how to deploy the robots, let us assume that a chief operator provides the robot team with a blueprint of the building with certain areas marked as higher probability for survivor locations. He would like the robots to construct a map of the structure as it is now with any hazardous features labeled, establish the presence or absence of victims in each area, and, most importantly, find any victims *as soon as possible.*

The robots are deployed and fan out, heading to the most promising locations first. The robots are many different sizes and shapes, equipped with different sensors. They cooperate to use their heterogeneity to their advantage.

One of the robots discovers a victim with its camera. There is no motion and the robot is unable to tell whether or not the victim is alive. It alerts its teammates of its discovery and several other robots explore the surrounding area in greater details, hoping to find other victims. It also requests help from a robot with capabilities to sense whether or not the victim is still alive since this will greatly affect the decision of whether or not to deploy human rescue workers to the location immediately.



(a) USARsim NIST Yellow arena with overlayed abstract representation

Robot	Capabilities
1	1
2	2
3	1,3
4	2
5	1,2

(b) Robots in example

Figure 1: The environment (a) is discretized (yellow lines) and represented as a graph where the costs of the edges signify the distance between the centers of adjacent nodes if there is no wall (green) between them. The example has 10 tasks (white) and 5 robots (b) that all enter at the node marked by the arrow.

When one robot detects sound from a duct, it requests help from its teammates with thermal imaging capabilities. One such robot has a very accurate thermal imaging device, an infrared camera, but is currently committed to investigating the previously identified victim. This robot offers to check quickly for signs of heat as it drives by the entrance to the corridor with the duct. Even at this distance, the infrared camera shows that there heat coming from inside the duct.

Since the robot who discovered the sound is too bulky to enter the confined area, it solicits the help of another, smaller robot to explore the duct. The small robot was mapping a crawl space, but quickly switches tasks.

Investigation of the duct reveals a four foot drop down to the probably victim location. This drop would make it impossible to retrieve the small robot. Although robots are valuable, the possibility of finding a trapped victim outweighs the cost and the decision is made to sacrifice the small robot. The small robot travels down the duct and discovers no victim but only a cat.

The rest of the robots explore the space as quickly as possible, looking for victims. Initially, they pass through the space as quickly as possible, looking for obvious signs of life. Next, they make a second, much more careful pass to see if they missed anyone.

The potential benefits with this team of robots is enormous, particularly if they could make the sorts of decisions described above autonomously, since they may not be within easy communication with their human team members. It must also be easy to change the goals of the search. For example, after 48 hours and finding all the victims, the rescue coordinator may decide to explore the stability of the structure with the robots so they can decide how to best clean up the mess.

This motivational example illustrates a number of complexities of the domain. As the first step towards finding a solution, we formally describe the problem.

3.1 Problem Formalization

Given a map of the environment with tasks to be accomplished and a set of robots with heterogeneous capabilities, we need an approach to formalize the problem and characterize the solution. Figure 1(a) shows a sample environment overlayed with an abstraction of the environment the robots can use for planning. We define the resulting time critical tight coordination team planning problem as follows:

Definition **3.1.** A **time critical tight coordination team planning problem** is denoted by ρ and consists of a five-tuple, $\langle \mathcal{R}, \mathcal{G}, E, \mathcal{C}, T_{max} \rangle$ where $\mathcal{R} = \{R^1, R^2, ..., R^N\}$ is the set of robots on the team, $\mathcal{G} = \{G^1, G^2, ..., G^M\}$ is the set of goals to be accomplished, E is the robots' representation of the physical environment, $\mathcal{C}$ is the set of additional constraints on the system as described in section 3.5, and T_{max} is the time limit for achieving the goals.

Internal to this representation is the assumption that there exists a set of K capabilities relevant to the completion of the set of goals. Each robot is characterized by its current position and a set of binary capabilities it possesses, S^n where S^n is a vector whose length is equal to the cardinality, K, of the set of capabilities relevant to this environment.

Each goal is similarly characterized by its location in the environment, the capabilities required to achieve the goal, the expected time required to achieve the goal, and the reward. We can then characterize the m-th goal as a tuple $G^m := (N^m, d^m, Q^m(t), C^m)$ where C^m, the capabilities required to achieve goal m, is a vector whose length is equal to K and $C^{m_k} = 1$ if capability k is required to achieve task m and 0 otherwise. The duration requirement means that all capabilities must be satisfied by the set of robots, $\mathcal{R}$, at location N^m for the entire continuous duration, d^m, in order for the goal to be achieved. Rewards, Q^m, are time varying and there exists some time limit, T_{max}, after which time all rewards are 0.

The system constraints, $\mathcal{C}$, may include temporal order constraints on the goals, constraints on shared resources, or constraints on the assignments of robots to goals. We describe a first order constraint language for combining goal, robot, and resource constraints in section 3.5.

While the problem formulation imposes no restrictions on the representation of the environment as long as robots can perform path planning in it, we find it useful to discretize the environment and represent it as a graph with nodes and edges which allows path planning using standard graph algorithms. Another possibility is to use an occupancy grid representation and apply D^* search [23].

The solution to the problem described above is called the *team plan* though it considers not only the allocation of robots to goals but also the schedule of when each goal is to be accomplished. In order to preserve the semantics of the original problem while maintaining a level of abstraction suitable for a robot with a typical three-tiered architecture, for example [58], we define a team plan as a set of *tightly coupled schedules* containing one schedule for each robot. We preserve the original problem semantics through the goal oriented representation shown in figure 2.

Definition **3.2.** A **tightly coupled schedule** represents the commitments of an individual robot, R^n to the team and consists of a set of steps, $\mathcal{S}_{R^n} = \{S_1, S_2, ...S_\omega\}$. Each step is defined as

$$S_i = \langle G^i, P, T_{start}, T_{travel}, T_{wait}, T_{work}, \mathcal{D} \rangle$$

where G^i is the goal that the robot works on during step i, P is the path the robot plans to take to get to the goal, T_{start} is the time at which it begins the step, T_{travel} is the time allocated to traverse path P, T_{wait} is the time spent waiting for the other robots working on the joint goal to arrive, T_{work} is the amount of time allocated to achieving the goal, and $\mathcal{D}$ contains the coordination commitments for this step.

Coordination commitments explicitly represent the requirements of the robot for this goal. An example coordination commitment might be to provide capability 1 in order to achieve goal i. If another robot has the coordination commitment of providing capability 2 for goal i, an inter-robot coordination commitment exists. Coordination commitments signify not only *what* the robot must do but *why* it is important.

The number of steps in a robot's schedule is the *planning horizon*, denoted ω. It is possible for different robots to have different planning horizons. It is also possible for robots to have no goal to work on during the later steps in which case the goal may be the *idle* goal.

Definition **3.3.** A **team plan** consists of the set of tightly coupled schedules for all robots on the team, $\mathcal{T} = \{\mathcal{S}_{R^1}, \mathcal{S}_{R^2}, ...\mathcal{S}_{R^n}\}$.

A sample team plan is illustrated in figure 2. Due to inaccuracies and uncertainties in the models of elements in the time critical tight coordination team planning problem (e.g. incorrect path length estimates) and dynamic events (e.g. a wall collapsing to block a path) it is improbable that the team plan will be

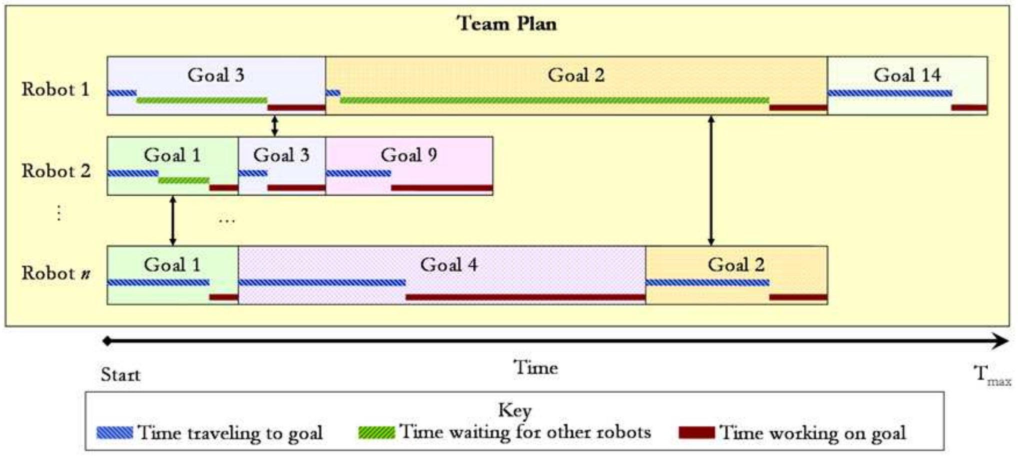


Figure 2: A team plan consists of the tightly coupled schedules from all robots in the team. The tightly coupled schedules for robots 1, 2, and n are illustrated above. All robots have three steps in their schedule so the planning horizon, $\omega = 3$. Each step contains $\langle G^i, P, T_{start}, T_{travel}, T_{wait}, T_{work}, \mathcal{D}\rangle$, where G^i is the goal the robot will work on during this step, P is the path the robot traverses to get to the goal (not shown here), T_{start} is the time at which the step begins, T_{travel} is the time traveling to the goal, T_{wait} is the time waiting for other robots, T_{work} is the time working on the goal, and $\mathcal{D}$ are the coordination commitments for the step. Inter-robot coordination commitments are illustrated with arrows.

successfully executed. In Section 3.6 we discuss how robot teams can interleave planning and execution. Despite the expectation that the plan will fail, this formalization has a number of benefits. For example, we can choose to adopt algorithms that are optimal over a short planning horizon without devoting as much computation to long term plans. We describe such an algorithm in Section 3.4.1. Alternatively, the formalization allows temporally invariant reasoning. For example, it may be necessary to schedule a certain goal far in the future. By planning the goal in advance, the team can accomplish this goal even if the robots are unable to communicate until they rendezvous to complete the goal though the intervening steps in the schedules of the various robots may significantly differ from the original team plan.

3.2 Mathematical Programming Model

This thesis proposes to build a mathematical programming model of the time critical team planning problem, the solution to which is a schedule in absolute time that can be executed by the robots, is shown in figure 2.

The term program does not refer to a computer program but rather *a plan or system under which action may be taken toward a goal* (Merriam-Webster Online Dictionary). A mathematical programming model therefore is a problem that requires some sort of *optimization*, either minimizing or maximizing some quantity. Mathematical programming models contain one or more decision variables for which we seek values so as to optimize the *objective function*. These decision variables may be subject to certain constraints or bounds [84] [37]. Within mathematical programming, there are a number of special classes of problems, the most commonly used of which are listed below in increasing order of difficulty to solve [40]:

- **Linear Programs** (LP) are of the form

$$\begin{array}{rll} \text{Maximize} & c^T x & \text{objective function} \\ \text{Subject to} & Ax \le b & \text{constraints} \\ & l \le x \le u & \text{bounds} \end{array}$$

  Linear programs are so called because all decision variable terms are linear (neither in the objective function nor in the constraints do we find terms such as x_1^2, $1/x_2$ or $x_1 x_2$). A number of algorithms have been developed for solving linear programs including primal simplex, dual simplex, network simplex, and the primal and dual log barrier algorithms [40]. Linear programs can be solved in polynomial time.

- **Mixed Integer Linear Programs** (MILP) are of the same form as linear programs but some of the decision variables, x are constrained to be integer. While LPs can be solved in polynomial time, MILPs are in $\mathcal{NP}$. MILPs are sometimes denoted as Mixed Integer Programs or MIPs since historically integer programs were assumed to be linear unless otherwise noted [37].
- **Quadratic Programs** (QP) are of the form

$$\begin{array}{rll} \text{Maximize} & c^T x + \frac{1}{2}x^T Q x & \text{objective function} \\ \text{Subject to} & Ax \leq b & \text{constraints} \\ & l \leq x \leq u & \text{bounds} \end{array}$$

  Quadratic programs can be solved using the primal and dual simplex algorithms or primal dual log barrier algorithm [40] though the quadratic function is required to be convex if the function objective is a minimization or concave when the objective function is maximized.
- **Quadratic Constrained Programs** (QCP) are of the form

$$\begin{array}{rll} \text{Maximize} & c^T x + \frac{1}{2}x^T Q x & \text{objective function} \\ \text{Subject to} & Ax \leq b & \text{constraints} \\ & \frac{1}{2}x^T Q_k x + a_k x \leq b_k & \text{quadratic constraint} \\ & l \leq x \leq u & \text{bounds} \end{array}$$

  The constraints are also permitted to be quadratic functions.
- **Mixed Integer Quadratic Programs** (MIQP) are of the same form as QPs except that some of the variables are constrained to be integer.
- **Mixed Integer Quadratic Constrained Programs** (MIQCP) are of the same form as QCPs except that some of the variables are constrained to be integer.

We use the simplest class of optimization problems with sufficient expressiveness for representing the problem. By modeling goal rewards as decreasing linearly with time, the problem can be modeled as a MILP. Although there are multiple ways to model time varying rewards, we assume that there is some time limit, T_{max}, at which time all rewards are 0. A continuous time model allows efficient modeling of the scheduling problem. Although the problem is still $\mathcal{NP}$-hard, modeling it as an MILP allows us to take advantage of CPLEX's advanced linear programming and mixed integer programming algorithms.

There are several challenges in building a MILP model of the time critical tight coordination team planning problem. To illustrate this, consider our initial naïve problem formulation in which we discretize both the environment and time and create a variable for every node in the environment for every timestep. For all but the smallest environments, this is computationally infeasible. This is because constraint optimization algorithms are inefficient at solving the path planning problem [7]. Likewise, discretizing time causes problem complexity to increase inversely with the time step size and directly with T_{max}. Discrete time models may be suboptimal if the time step size is not sufficiently small.

We overcome these obstacles by introducing a goal oriented representation that is independent of environment size and uses a continuous time model. The goal oriented representation is illustrated in our definitions of a team plan and a tightly coupled schedule (Figure 2). This representation enables us to preprocess path planning since each robot trajectory is abstracted to a single variable, the amount of time allocated to travel to a goal. Path planning is precomputed by building a goal map (Figure 3), a fully connected graph where nodes represent the set of goals and each edge represents the travel time for a robot to get from the location of one goal to another. Additionally, the goal map contains a start node and an idle node. The start node represents the starting position of a robot and has a directed edge to each goal node. Every node has an edge with zero cost to the idle node. Each robot's tightly coupled schedule represents a path through the goal map that can be mapped back onto a path through the original environment. The idle node has no transitions out since a robot would never schedule a timeslot as idle unless there were no more available tasks for it to work on.

If the environment is discretized and represented as a graph with n nodes, the goal map can be constructed using the Floyd Warshall algorithm ($O(n^3)$) [20].

We will now describe our MILP mathematical model of the problem. Note that R^n and G^m correspond to robots or goals while subscripts denote a variable that points to a robot or goal. Variables are either binary or real valued and fall into one of three classes:

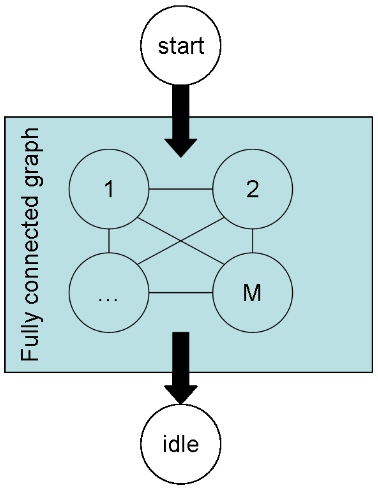


Figure 3: A goal map contains a node for every goal in the problem plus a start node and an idle node. The goal nodes are fully connected and each edge represents the time for a robot to travel from the location of one goal to the location of the second goal. In general, the edge costs may differ depending on the capabilities of a robot.

- Goal variables include binary variables G_m and R_nG_m that denote whether the goal is scheduled and whether robot n works on goal m respectively. Goal variables G_m_start and $R_nG_m\_start$ are real valued variables for the absolute time at which the goal is scheduled or the robot is scheduled to work on the goal. $R_nG_m\_do$ is a real valued variable denoting the amount of time robot n dedicates to achieving goal m. All real valued variables take on the value 0 if their binary counterparts are 0. Additionally, we define antivariables $G_m\_start^*$ and $R_nG_m\_start^*$ that have the same value as their corresponding variables but take on the value of T_{max} if their binary counterparts are 0.
- Schedule variables include binary variables $R_nO_iP_{(g_1,g_2)}$ that denote whether or not robot n travels from goal 1 to goal 2 in its i^{th} timeslot. The time spent traveling, waiting, and working on the goal during a given timeslot are represented with real valued variables $R_nO_i\_travel$, $R_nO_i\_startDo$, and $R_nO_i\_do$. $R_nO_i\_startDo$ is an absolute time while $R_nO_i\_travel$ and $R_nO_i\_do$ represent amounts of time.
- Linking variables $R_nG_mO_i$ (binary) and $R_nG_mO_i\_do$ and $R_nG_mO_i\_startDo$ (real) are used to establish constraints between the schedule and goal variables.

We want to maximize solution utility which is the sum the reward functions from all tasks accomplished evaluated at the time at which they were accomplished. We assume that rewards decrease linearly with time.

$$Utility = \sum_{m \text{ s.t. } G^m \in \mathcal{G}} \frac{T_{max} - G_m\_start^*}{T_{max}} Q^m$$

We subject this objective function to the following constraints.

- Robots start at an abstract start node:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}} : R_nG_{start}O_0 = 1$$

- Robots must work on exactly one task at a time:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, 1 \le i \le \omega} : \sum_{m \text{ s.t. } G^m \in \mathcal{G}} R_nG_mO_i = 1$$

- Except the idle task, a robot may not work on a task more than once:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} : \sum_{1 \le i \le \omega} R_nG_mO_i \le 1$$

- Network constraint 1: In order to take the path from g_1 to g_2 at this timestep, the robot must have been at node g_1 at the end of last timestep:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, g_1 \text{ s.t. } G^{g_1} \in \mathcal{G}, 1 \leq i \leq \omega} : \sum_{g_2 \text{ s.t. } G^{g_2} \in \mathcal{G}} R_n O_i P_{(g_1, g_2)} = R_n G_{g_1} O_{i-1}$$

- Network constraint 2: In order to take path from g_1 to g_2 at this timestep, the robot must end up at node g_2:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, g_2 \text{ s.t. } G^{g_2} \in \mathcal{G}, 1 \leq i \leq \omega} : \sum_{g_1 \text{ s.t. } G^{g_1} \in \mathcal{G}} R_n O_i P_{(g_1, g_2)} = R_n G_{g_2} O_i$$

- Network constraint 3: The time allocated for travel in this timestep for this robot must at least equal the time necessary for this robot to traverse path (g_1, g_2). Since this is robot dependent, we can have different costs for different classes of robots (e.g. large, fast wheeled machines versus small, legged, climbing robots).

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, g_1, g_2 \text{ s.t. } G^{g_1}, G^{g_2} \in \mathcal{G}, 1 \leq i \leq \omega} : R_n O_i P_{(g_1, g_2)} ShortestPath(g_1, g_2, R_n) \leq R_n O_i\_travel$$

- Initial condition for time: (δ = starting time, $\delta \geq 1$).

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}} : R_n O_1\_startDo = R_n O_1\_travel + \delta$$

- General time constraint based on timestep definition

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, 1 \leq i \leq \omega} : R_n O_i\_startDo = R_n O_{i-1}\_startDo + R_n O_{i-1}\_do + R_n O_i\_travel$$

- Robots with the appropriate capabilities must be present at a node in order for the capability to be covered.

  $$\forall_{m \text{ s.t. } G^m \in \mathcal{G}, k \text{ s.t. } C^{m_k} = 1} : \sum_{n \text{ s.t.} R^n \in \mathcal{R}, S^{n_k} = 1} R_n G_m \geq G_m$$

- A joint task cannot start until all allocated robots are present:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} : G_m\_start \geq R_n G_m\_startDo$$

  $$\forall_{m \text{ s.t. } G^m \in \mathcal{G}} : G_m \cdot T_{max} \geq G_m\_start$$

- All robots must work on the joint task for the entire duration:

  $$\forall_{m \text{ s.t. } G^m \in \mathcal{G}, n \text{ s.t. } R^n \in \mathcal{R}}$$

  $$G_m\_start - R_n G_m\_startDo + d^m \geq R_n G_m\_do$$

  $$R_n G_m\_do \leq R_n G_m \cdot T_{max}$$

  $$R_n G_m\_do \geq d^m R_n G_m$$

  $$\forall_{m \text{ s.t. } G^m \in \mathcal{G}} : \sum_{n \text{ s.t. } R^n \in \mathcal{R}} R_n G_m\_do \geq d^m \sum_n R_n G_m$$

- A robot may not work on a joint task before the scheduled start time:

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} : G_m\_start \leq R_n G_m\_startDo^*$$

- All times must be less than T_{max}. All times must be non-negative as well, but this constraint is frequently assumed by linear programming solvers.

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} : R_n G_m\_start + R_n G_m\_do \leq T_{max}$$

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, 1 \leq i \leq \omega} : R_n O_i\_startDo + R_n O_i\_do \leq T_{max}$$

  $$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}, 1 \leq i \leq \omega} : R_n G_m O_i\_startDo + R_n G_m O_i\_do \leq T_{max}$$

  $$\forall_{m \text{ s.t. } G^m \in \mathcal{G}} : G_m\_start + d^m \leq T_{max}$$

- Linking constraints relate goal variables to their antivariables and goal variables to schedule variables through linking variables. Linking constraints are necessary in order for the resulting solution to be legal.

  Relate $R_n G_m$ to $R_n G_m O_i$

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} R_n G_m = \sum_{1 \leq i \leq \omega} R_n G_m O_i$

Force $R_n G_m O_i\_startDo$ to 0 if $R_n G_m O_i$ is false.

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}, 1 \leq i \leq \omega} : R_n G_m O_i \cdot T_{max} \geq R_n G_m O_i\_startDo$

Force $R_n G_m O_i\_startDo = R_n O_i\_startDo$ if $R_n G_m O_i$ is true.

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, 1 \leq i \leq \omega} : \sum_{m \text{ s.t. } G^m \in \mathcal{G}} R_n G_m O_i\_startDo = R_n O_i\_startDo$

Relate $R_n G_m\_start$ to $R_n G_m O_i\_startDo$ (taking advantage of the property that $R_n G_m O_i\_startDo$ is non-zero at most once).

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} : R_n G_m\_start = \sum_{1 \leq i \leq \omega} R_n G_m O_i\_startDo$

Set $R_n G_m O_i\_do = 0$ if $R_n G_m O_i$ is false.

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}, 1 \leq i \leq \omega} : R_n G_m O_i\_do \leq R_n G_m O_i \cdot T_{max}$

Set $R_n G_m\_do$ to the corresponding value of $R_n G_m O_i\_do$ taking advantage of the property that at most one value of $R_n G_m O_i\_do$ is non-zero.

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, m \text{ s.t. } G^m \in \mathcal{G}} : \sum_{1 \leq i \leq \omega} R_n G_m O_i\_do = R_n G_m\_do$

Relate $R_n O_i\_do$ to $R_n G_m O_i\_do$

$\forall_{n \text{ s.t. } R^n \in \mathcal{R}, 1 \leq i \leq \omega} : R_n O_i\_do = \sum_{m \text{ s.t. } G^m \in \mathcal{G}} R_n G_m O_i\_do$

Force $G_m\_start^*$ to T_{max} if G_m is false and to G_m_start otherwise.

$\forall_{m \text{ s.t. } G^m \in \mathcal{G}} : G_m\_start^* \geq (1 - G_m) \cdot T_{max}$

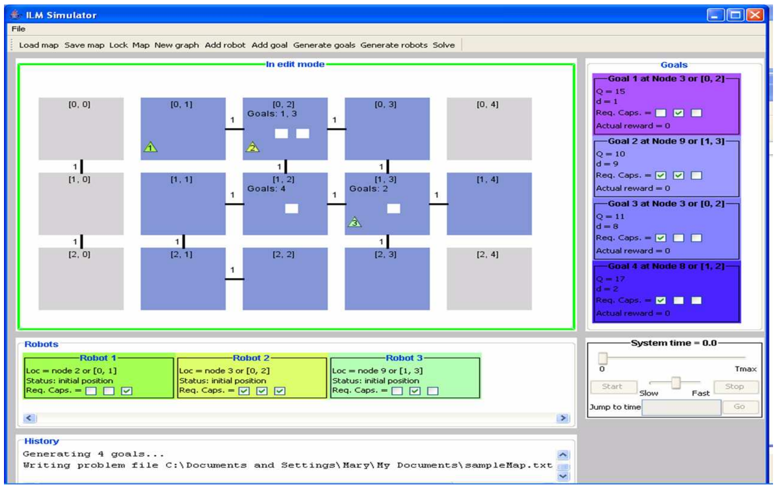
$\forall_{m \text{ s.t. } G^m \in \mathcal{G}} : G_m\_start^* \geq G_m\_start$

3.3 Abstract Simulator

Before we describe how we solve these problems, we will describe our process for generating, representing, and simulating the problems. We have developed an abstract simulator, for want of a better name, COCoASim. COCoASim is implemented in Java and consists of over 11,000 lines of code including algorithm implementation, simulator implementation, and debugging utilities. We plan on making the source code publicly available in the near future.

We represent physical environments as collections of nodes and weighted edges where the weight represents the time cost for a robot to travel along that edge. For simplicity, we use undirected edges though an extension to directed edges is trivial. Nodes in the environment are laid out in a grid with four point connectivity (again for simplicity and to avoid the challenging graph layout problem). We generate new environments by creating a grid of *rowsxcols* nodes, randomly generating edges between adjacent nodes (using the *nextBoolean* method of Java's Random class to decide if an edge exists and the *nextInt* method bounded by $max(rows, cols)$ to choose the edge cost), and selecting the largest connected subgraph. Unconnected nodes are discarded. When the capabilities and locations of robots or goals are not specified by the class of problem as described in Section 3.4.2, these too are chosen by random. Capabilities possessed by robots and required by goals are generated with the *nextBoolean* function for each capability in the problem. If a robot with no capabilities or a goal with no requirements is generated, a capability is randomly selected so that every robot has at least one capability and each goal has at least one requirement. If the environment has x connected nodes, robot and goal locations are chosen by randomly drawing an integer from x which corresponds to the index of the connected node. A simple example with a 3x5 base environment, three robots, and four goals is shown in figure 4. The unconnected nodes are drawn in gray while the 10 nodes that make up the largest connected subgraph are shown in blue.

In addition to generating problems, COCoASim can find a team plan using the algorithms we will describe in Section 3.4.1 and then display the plan. There is currently no uncertainty or implicit simulation of dynamic events in COCoASim. Robots always take the same amount of time to traverse an edge or complete a goal. There is no spatial interaction between robots. This was a design decision made to eliminate the need for robots to discover dynamic events. However, COCoASim does support certain types of explicit dynamic events. These dynamic events or replanning catalysts are discussed in detail in Section 3.6 and can be specified by the user or randomly generated and are detected one or more robots at some time (e.g. 15 minutes into executing the team plan, robot 2 discovers that a boulder is blocking a door and the edge from [0,2] to [1,2] no longer exists and so must find another path).



Goal 1 at Node 3 or [0, 2]
Q = 15
d = 1
Req. Caps. = ☐ ✓ ☐
Actual reward = 0

Goal 2 at Node 9 or [1, 3]
Q = 10
d = 9
Req. Caps. = ✓ ✓ ☐
Actual reward = 0

Goal 3 at Node 3 or [0, 2]
Q = 11
d = 8
Req. Caps. = ✓ ☐ ☐
Actual reward = 0

Goal 4 at Node 8 or [1, 2]
Q = 17
d = 2
Req. Caps. = ✓ ☐ ☐
Actual reward = 0

Robot 1
Loc = node 2 or [0, 1]
Status: initial position
Req. Caps. = ☐ ☐ ✓

Robot 2
Loc = node 3 or [0, 2]
Status: initial position
Req. Caps. = ✓ ✓ ✓

Robot 3
Loc = node 9 or [1, 3]
Status: initial position
Req. Caps. = ☐ ✓ ☐

```
Generating 4 goals...
Writing problem file C:\Documents and Settings\Mary\My Documents\sampleMap.txt
```

(a) COCoASim screenshot

```
Maxtime 1000

Capabilities 3

#Robot ID, Capabilities
Robot 1 [ 0 0 1]
Robot 2 [ 1 1 1]
Robot 3 [ 0 1 0]

#Goal ID, Location, Requirements...
#...Reward, Duration, Include
Goal 1 Node 3 [0 1 0] 15 1 1
Goal 2 Node 9 [1 1 0] 10 9 1
Goal 3 Node 3 [1 0 0] 11 8 1
Goal 4 Node 8 [1 0 0] 17 2 1


Mapsize 10
#environment 3 x 5
Node 2 Node 3 1.0
Node 3 Node 4 1.0
Node 3 Node 8 1.0
Node 4 Node 9 1.0
Node 7 Node 8 1.0
Node 7 Node 12 1.0
Node 8 Node 9 1.0
Node 9 Node 10 1.0
Node 9 Node 14 1.0
Node 12 Node 13 1.0

#Robot starting locations
Start Robot 1 Node 2
Start Robot 2 Node 3
Start Robot 3 Node 9
```

(b) Text file problem representation

Figure 4: Randomly generated small (3x5) world in in COCoASim with included nodes shown in blue and excluded nodes shown in gray (a). Robots and goals were also randomly generated. This environment can be saved as a text file (b) which is used to generate the MILP. Node IDs are by row major order and start at 1.

3.4 Solution Techniques

By modeling the problem as an MILP, we can leverage a vast amount of work in solving these problems including the commercially available solver, CPLEX [40]. The standard approach for solving an MILP is to relax the constraint that certain variables be integer and solve the resulting linear program (LP) using LP algorithms such as the simplex or dual simplex algorithm. This noninteger solution is known as the

Algorithm 1 MILP Anytime Algorithm

```
1: $oldSoln = \{\}$
2: for $\omega = 1$ to # Goals do
3:     $milp \Leftarrow$ CreateMILP($\omega$)
4:     $start \Leftarrow$ HeuristicScheduler($oldSoln$, $\omega$)
5:     $optimalSolnForPH \Leftarrow$ Optimize($milp$, $start$)
6:     $oldSoln = optimalSolnForPH$
7: end for
8: return $optimalSolnForPH$
```

relaxed solution and provides an upper bound on system performance. In the case that the relaxed solution is integral, this optimal solution is returned.

Since the problem is $\mathcal{NP}$-hard, generally the system must employ the branch and bound algorithm to search for the optimal integer solution. Without additional information, the search for the integer solution begins near the relaxed solution. Depending on the structure of the search space, finding a feasible solution can take a long time (hours to find a feasible solution to a large search and rescue problem and days or weeks to find the optimal solution). This is unacceptable for the search and rescue domain since planning delays the onset of execution. Furthermore, as robots execute their plans, handle robot failures, and discover inconsistencies in their models, they will need to replan online which requires real time algorithms.

3.4.1 Real time algorithms

Particularly for problems with few or no constraints, it is possible to use simple heuristics to find solutions in polynomial time. In fact, the search and rescue problem has elements, specifically rewards that decrease over time, that suggest that greedy heuristics would perform reasonably well. Conveniently, we can combine heuristics and MILP solution techniques resulting in an algorithm superior to either individual approach.

As described in the previous section, solving an MILP involves branch and bound search. By default, this search begins around the relaxed solution. It is possible to generate an initial solution using a heuristic and use this as the starting point for the search. This enables the system to find a feasible solution quickly and then improve the solution with available time. Furthermore, the bounds in the branch and bound algorithm provide error bounds on a given solution so that it is possible to stop when the solution is within some bounds of the optimal or estimate distance of an arbitrary solution from optimal. We exploit this symbiosis between heuristic and optimal solver in our MILP anytime algorithm (Algorithm 1).

MILP Anytime Algorithm

The key idea in the MILP anytime algorithm is to make the interaction between the optimization algorithm (CPLEX) and heuristic (any number of heuristics can be used–we describe two below) an iterative process by starting with a small subproblem and expanding this until it includes the entire problem. At each iteration, a heuristic is used to generate a solution for the subproblem. This heuristic solution is used as a starting point for search in the optimization step. Since the resulting optimized solution to this subproblem is at least as good as the heuristic solution to the same subproblem, the heuristic uses this optimized solution as its starting point and only finds a solution for the expanded portion of the problem.

Once the problem has been formulated as a MILP, standard linear programming techniques can be applied to find the optimal relaxed solution (without integer constraints) which can be computed in polynomial time [20]. In our experiments, computing the optimal relaxed solution was very fast, requiring well under a second for problems such as the example in figure 1(a-b) and less than 10 seconds even for a large environment with 900 nodes, 20 robots, and 20 tasks. This relaxed solution provides an upper bound on team utility. Finding the optimal integer solution however is $\mathcal{NP}$ hard [20] and the standard algorithm used to search the space of integer solutions, branch and bound, has worst case exponential time complexity. The integrality gap between the relaxed and integer solutions frequently means that finding a feasible solution can take minutes or even hours.

In this respect, domain specific heuristics can easily outperform an ILP solver as they can compute feasible

Algorithm 2 Myopic heuristic

```
1: $\omega = 1$
2: $unsatGoals = Goals$
3: $currSchedule = \{\}$
4: while ($\omega \leq$ # Goals) && !$isEmpty(unsatGoals)$ do
5:     $prob \Leftarrow$ CreateMILP(1, $unsatGoals$)
6:     $partialSoln \Leftarrow$ Optimize($prob$)
7:     $currSchedule \Leftarrow$ Schedule($partialSoln$)
8:     $unsatGoals \Leftarrow unsatGoals -$satGoals($currSchedule$)
9:     $\omega = \omega + 1$
10: end while
11: return $currSchedule$
```

if suboptimal solutions very quickly. Furthermore, the time varying property of the original problem suggests a greedy approach is likely to perform well. Most ILP solvers including the package we use (CPLEX) have the ability to take an existing solution as a starting point for the search. Recall from the problem formulation that the planning horizon, ω, is defined as the number of tasks (including the idle task) scheduled for a given robot. We combine the benefits of domain specific heuristics with the optimality guarantees of an ILP solver by interleaving the two approaches over an ever increasing planning horizon (Algorithm 1). The branch and bound algorithm stores the best solution so the algorithm may be interrupted at any time and will simply return the best solution thus far. When the planning horizon equals the number of tasks, the solution is guaranteed to be optimal since it grants every robot the possibility of accomplishing every task. In practice however, no single robot performs all tasks so the optimal solution may be found at a lower planning horizon.

Heuristics

Since goal rewards decrease over time, it is reasonable to expect greedy heuristics to perform well. Two greedy heuristics arise naturally from the problem variables–robots and goals.

In the first heuristic, which we call the *myopic* heuristic, robots use the MILP problem formulation to find the optimal allocation of goals for a single planning horizon. The goals allocated are fixed in the schedule and the process is iterated on the remaining goals until all goals are allocated or unable to be allocated as shown in algorithm 2. Empirically, this is very fast–well under a second–even for large problems. This is because problem size grows exponentially with the length of the planning horizon so this subproblem is much smaller than the original problem.

The second heuristic, or *greedy goal* heuristic, is inspired by market based task allocation algorithms which have been shown to work well in a variety of domains [24]. Since the goals are known in advance, they are sorted and auctioned off in decreasing order of reward. For each required capability of each goal, robots bid their costs to provide that capability based on their current schedules. The lowest bidder is assigned to that goal capability. The greedy goal heuristic is less sophisticated than many market based systems as it does not allow robots to reauction their goals.

The convergence of the MILP anytime algorithm is illustrated in figure 5. We ran the MILP anytime algorithm with the myopic heuristic on the sample problem from figure 1(a-b) with a time limit of 1000 time units. Recall that the sample problem consisted of 5 robots entering an arena from a common location and seeking to accomplish 10 goals spread out through the environment. The discrete jumps correspond to the heuristic scheduler step of Algorithm 1 (line 4) since the planning horizon is extended allowing additional tasks to be accomplished and slight improvements arise from search performed by CPLEX. The MILP anytime algorithm with the myopic heuristic reached a maximum utility of 124.5 after about 45 seconds. We used CPU time as the metric for this experiment and since CPLEX was set to search deterministically, these results are deterministic to within the accuracy of the system timer. For this example, the myopic heuristic by itself terminated with a maximum utility of 124.35 in less than 0.1 seconds. The greedy goal heuristic alone resulted in a utility of 123.3. Without the MILP anytime algorithm, CPLEX required 8 minutes to find a feasible solution and still had not converged to the optimal solution when stopped after an hour with a utility of 121.4. These results confirms our hypothesis that heuristics can significantly improve performance

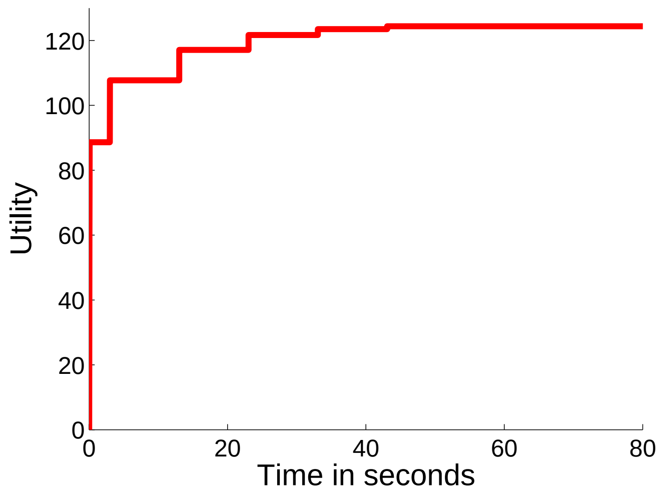


Figure 5: MILP Anytime Algorithm with myopic heuristic: results over time on example from figure 1(a-b). All times are CPLEX CPU times on a 2.8GHz Pentium 4 machine with 1GB RAM.

over standard MILP solution techniques alone. The gains for using the MILP anytime algorithm versus a heuristic appear relatively small in this example. This is partially because the value for T_{max} was large so small time savings realized by more efficient allocations have little effect on the overall utility. Another reason for the small gains over the heuristics is that the heuristics work relatively well on this problem as we hoped. However, each of these heuristics has its limitations and can lead to inefficient behavior if not combined with an optimization algorithm. We first demonstrate the strengths and weaknesses of the heuristics analytically and then empirically.

Myopic heuristic limitations

The myopic heuristic performs poorly if it is best for one robot to assume more goals than its teammates even though other robots also have the capability to accomplish the goals. For example, if there are two goals with the same requirements in the same room and two identical robots, one already in the room and one in a different building, the myopic heuristic will assign one goal to each robot although the optimal solution is clearly to assign both goals to the close robot as shown in figure 6(a).

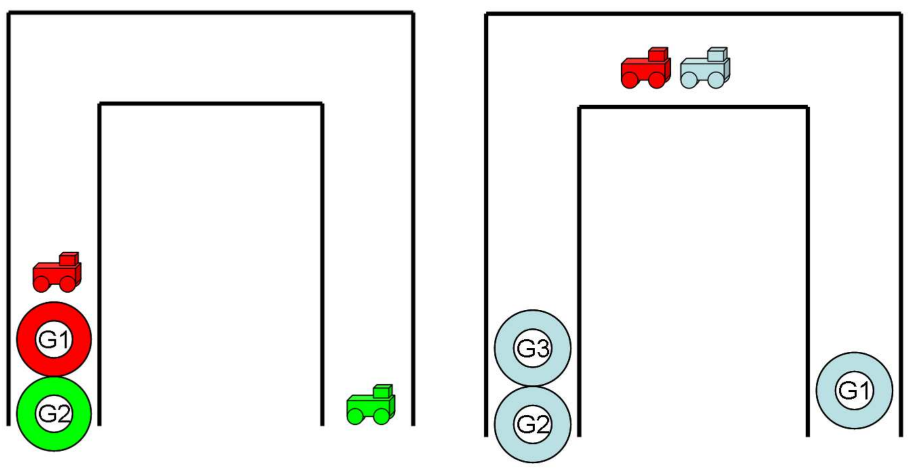
Greedy goal heuristic limitations

The greedy goal heuristic would correctly solve the example in figure 6(a), but may perform poorly if the problem requires careful resource management. Consider a disaster site with two flooded rooms and a dry room and a team of two robots. However, only robot 1 is waterproof and able to explore the flooded rooms. If robot 1 starts slightly closer to the dry room and the goal of exploring the dry room is allocated first, the greedy goal heuristic will assign all three goals to robot 1 although it would be more efficient to assign the dry room to robot 2 to explore. In the example in figure 6(b), only the blue robot has the necessary capabilities for goals 2 and 3 but either robot could do goal 1. However, if goal 1 is allocated first, the blue robot will win all three goals while the red robot remains idle.

In order to compare the performance of the various algorithms and heuristics in a more systematic manner, a large number of test cases that capture some of the challenges of the problem space are needed.

3.4.2 Benchmarks

The lack of benchmarks for heterogeneous multirobot coordination with joint goals is an obstacle to rigorously comparing algorithms and heuristics. We have developed a collection of benchmarks designed to illustrate some of the challenges of a complex domain such as search and rescue including the limitations of the



(a) Myopic heuristic limitations (b) Greedy heuristic limitations

Figure 6: (a) Both the red and green robots have the necessary capabilities for either task. The myopic heuristic allocates goal 1 to the red robot and goal 2 to the green robot in the first iteration. The myopic heuristic performs poorly when it is best for one robot to assume more goals than its teammates over some subproblem. (b) The red robot is only capable of performing goal 1 while the blue robot is capable of performing all three goals. If goal 1 is scheduled first, all three goals are allocated to the blue robot which is clearly suboptimal. The greedy heuristic leads to inefficiencies when capabilities must be careful managed.

heuristics as described. As a service to the research community, we have set up a repository to allow researchers to access our benchmarks and contribute their own.

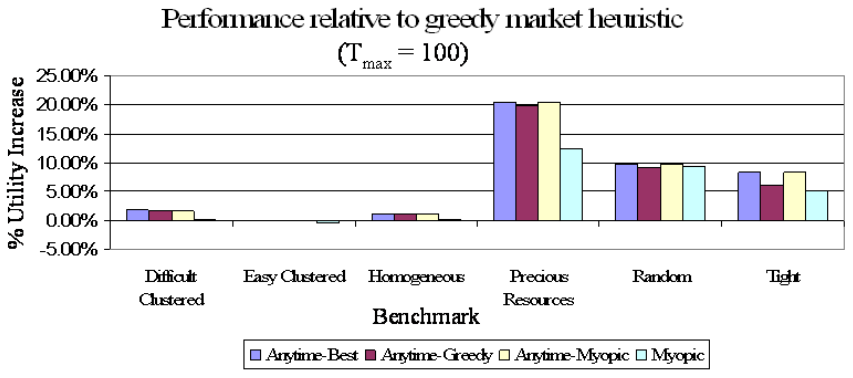
As described in Section 3.3, we created an abstract simulator, COCoASim, capable of generating random problems to certain specifications. We propose six interesting classes of benchmarks:

- Homogeneous: all robots and goals are homogeneous; all goals have a uniform duration, but varying rewards, and are distributed at random through the environment.
- Tight coordination: each robot has only a single capability while every task requires all three capabilities forcing three robots to work together on every task. The reward, duration, and location of each goal is chosen at random.
- Easy clustering: goals are clustered in one or two locations in the environment, but may be accomplished by any individual robot since each robot possesses all capabilities.
- Difficult clustering: goals are clustered in one or two locations in the environment, and require tight coordination as each robot only possesses a single capability.
- Precious resources: goals in this benchmark class are either easy or hard. The easy goals can be accomplished by any robot while only a single robot has the necessary capabilities for the hard goals forcing the system to use the "precious resources" (the robot with special capabilities) carefully.
- Random: everything, from robot capabilities to goal requirements, location, and duration, is randomly chosen. These benchmarks will exhibit some qualities of each of the other benchmark classes.

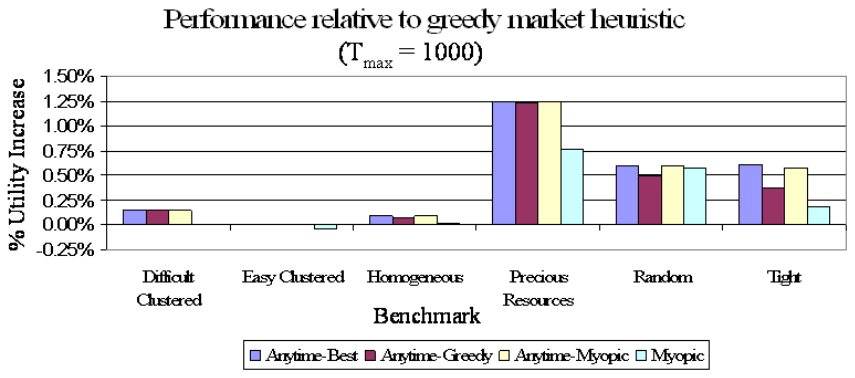
For each class of benchmark, we randomly generated 5 environments with 3 and 15 robots and 5 and 15 goals for a total of 20 problems in each benchmark class and 120 different problems in all. The environments were randomly generated as described in section 3.3 using 10 rows and 10 columns. All of the benchmarks generated use three relevant capabilities though the profiles of robot capabilities and goal requirements distinguish the various benchmarks as described above.

3.4.3 Results

We compared five solution techniques on these benchmarks: the myopic heuristic (Algorithm 2), the greedy goal heuristic, the MILP anytime algorithm (Algorithm 1) with the myopic heuristic as the heuristic solver on line 4 of Algorithm 1, the MILP anytime algorithm with the greedy goals heuristic as the heuristic solver, and the MILP anytime algorithm with the "best" heuristic in which the heuristic solver compares the myopic and greedy heuristics and returns the solution with the highest utility. The optimization step (line 5 of Algorithm 1) was limited to 90 seconds. We compared the results for three different levels of time pressure, T_{max} = 100, 1000, and 2000. Recall that rewards decrease linearly from their starting value to 0 at time T_{max}. The five algorithms tested on 120 test cases with three different values for T_{max} resulted in a total of 1800 runs. The greedy goal and myopic heuristics were relatively fast but even limiting the optimization to 90 seconds per run, the tests took over 30 hours to complete. The results for $T_{max} = 100$ and $T_{max} = 1000$ are shown in figure 7.



(a) $T_{max} = 100$



(b) $T_{max} = 1000$

Figure 7: Benchmark results using greedy goal heuristic as baseline averaged over 3 and 15 robots and 5 and 15 goals.

In general, we observed that the greedy goal heuristic performed well on the homogeneous and clustered benchmarks. This matches what we know about these benchmarks and the workings of the greedy goal heuristic. The homogeneous and easy clustered benchmarks have no joint goals so the problem reduces to the optimal assignment problem on which greedy market task allocation has been demonstrated to perform well [24]. The greedy goal heuristic performs the worst on the precious resource benchmarks which matches our earlier analysis of the limitations of the greedy goal heuristic.

The myopic heuristic performed at least as well as the greedy goal heuristic on all benchmarks except the easy clustered benchmarks on which it performed slightly worse. This matches our analysis of the limitations of the myopic heuristic. The performance is only slightly worse because the tasks are clustered together so that even if the tasks are allocated suboptimally for a planning horizon of 1, this is outweighed by the goals accomplished over the remainder of the planning horizon. In this sense, the clustered goal benchmarks were less successful than we had hoped in illustrating the limitations of the myopic heuristic.

The MILP anytime algorithm was able to significantly improve performance on a number of benchmarks with a maximum improvement of over 50%. This indicates that the heuristics benefit significantly from the relatively small amount of time spent in optimization. Without using the heuristic starting solution, CPLEX is often unable to find a feasible solution in 90 seconds which illustrates the benefits of the heuristic starting solution to the MILP solver.

We found that the results from the other values of T_{max} are nearly identical to figure 7 (and so are not included here) except in scale. The utility gains when $T_{max} = 2000$ are half the utility gains for $T_{max} = 1000$. Since utility is defined as a linear function of T_{max}, we would expect a linear relationship between T_{max} and utility gain. However, the utility gains when $T_{max} = 100$ are 15 times the utility gains of $T_{max} = 1000$. If the average utility gain for $T_{max} = x$ is denoted as U_x, $U_{100} = 15U_{1000} = 30U_{2000}$. Since utility is defined as a linear function of T_{max}, we would expect a linear relationship between T_{max} and U_x which is true on the interval $T_{max} = [1000, 2000]$ but not on the interval $T_{max} = [100, 1000]$. The reason for this is that as T_{max} decreases, suboptimal solutions are not able to accomplish as many goals which magnifies differences in solution quality. This illustrates the importance of optimality in the search and rescue domain where there is often significant time pressure.

3.5 System Constraints

The expressiveness of the constraint language for the system constraints determines the scope of problems that can be solved. Goal, robot, and resource constraints form the building blocks for this constraint language.

There is the potential for confusion when discussing constraints in the MILP problem formulation. *System constraints* refer to semantically meaningful constraints. *Problem constraints* refer to constraints in the MILP problem formulation that implement system constraints or make the problem formulation legal.

3.5.1 Goal constraints

Temporal goal constraints enable more efficient modeling. For example, the complex goal of rescuing an identified victim may require a beam to be lifted, the victim to be helped out, and first aid to be administered (figure 8). It is more efficient to model the problem as three separate but related goals, G^{lift}, $G^{extract}$, and G^{aid} with the goal constraints $G^{extract}$ *finishes* G^{lift} and $G^{extract}$ *meets* G^{aid} where *finishes* and *meets* are some of Allen's 13 temporal relationships [2]. This model allows the robots with the first aid and extraction capabilities an extra hour to work on other goals while the lifting robot finishes 20 minutes sooner.

Although we have not conducted experiments to verify this, we believe that the temporal goal constraints will improve human-robot interaction since it is more intuitive to achieve certain behaviors using constraints (e.g. *look under the stairs and then in the elevator shaft*) than by specifying rewards.

The seven temporal goal constraints below and their inverses implement Allen's 13 temporal relationships [2]. With the exceptions noted below, goal constraints are defined to hold if and only if both goals are scheduled, allowing for a trivial solution of not scheduling one goal.

- G^x *before* G^y: Goal G^x must be completed before G^y begins. This is the only goal constraint defined such that G^x may be scheduled without scheduling G^y. We also allow G^y to be replaced with an absolute time for the goal constraint G^x *before 4:00.*

- G^x *equal* G^y: Goals G^x and G^y must start at the same time and finish at the same time.

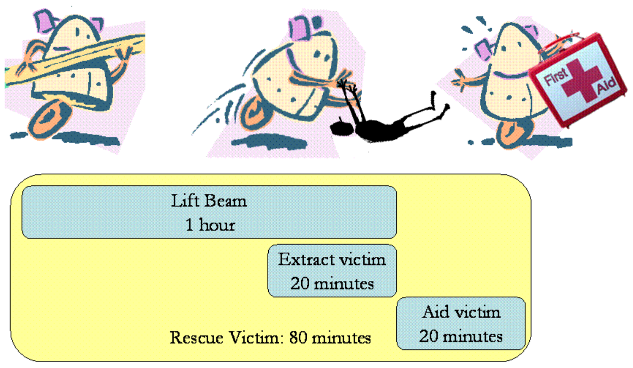


Figure 8: Three different robot capabilities are needed for the complex task of rescuing a victim. The goal can either be modeled as a single goal, G^{rescue} or as three separate goals G^{lift}, $G^{extract}$, and G^{aid} with the goal constraints $G^{extract}$ *finishes* G^{lift} and $G^{extract}$ *meets* G^{aid}.

- G^x *meets* G^y: The ending time of G^x equals the time at which G^y starts. We also allow G^y to be replaced with an absolute time; e.g. G^x *meets 4:00* would mean that G^x ends at 4:00 (and so starts at 4:00 - d^x).
- G^x *overlaps* G^y: Goal G^x starts before G^y starts; G^x ends before G^y ends; G^x ends after G^y starts.
- G^x *during* G^y: Goal G^x starts after G_y starts, and G_x ends before G_y ends. We also allow G^x to be replaced with an absolute time; e.g. *4:00 during* G^y would mean that goal y must be in progress at 4:00.
- G^x *starts* G^y: Goal G^x starts at the same time G_y starts but ends before G_y ends.
- G^x *finishes* G^y: Goal G^x ends at the same time G^y ends; G^x starts after G^y starts.

The problem formulation does not force any goal to be scheduled. There are many cases in the search and rescue domain where one or more goals must be met. One more goal constraint is therefore needed in addition to the temporal goal constraints, in order to specify that a goal must be accomplished: *do* G^m. The *do* constraint forces the goal to be scheduled at some time but does not restrict when it is scheduled.

3.5.2 Robot constraints

It is frequently convenient to specify additional constraints on the actions of individual robots. Rescue workers may wish to specify which robots should work on a goal or exclude a certain set of robots from high risk goals. This can all be effected with the *participantIn* constraint which operates on a robot and a goal. The robot constraint R^n *participantIn* G^m forces robot n to participate in goal m if the goal is scheduled.

Another important constraint is that a robot should return to base when finished with other goals. This is particularly important to consider at planning time if fuel is an issue. We model this with the *endAt* constraint: R^n *endAt* L means that robot n must return to location L at the end of its mission.

3.5.3 Resource constraints

Consumable resources are an important part of the search and rescue domain. Therefore, resource constraints are a necessary part of the problem formulation. We do not consider renewable resources for this domain. In general, we assume that resource constraints apply to some subteam of robots, $\mathcal{R}_s \subseteq \mathcal{R}$ and that the resource consumption for each robot is independent.

- $\mathcal{R}_s$ *limitFuelTo* F indicates that the robots in $\mathcal{R}_s$ may collectively travel no farther than F which is the amount of fuel that they possess. By setting $\mathcal{R}_s = R^n$, we can limit the amount of fuel an individual robot consumes.

- $\mathcal{R}_s$ *limitResourcesOf* $\langle f(R^n, G^m), \rho\rangle$ indicates that the robots in $\mathcal{R}_s$ may collectively consume no more than ρ of a particular resource. The $limitResourcesOf$ constraint applies to resources that are only consumed when a robot works on a goal. The function, $f(R^n, G^m)$, represents the amount of the resource robot, R^n, would consume working on goal, G^m. It must be defined for all goals in the problem, $G^m \in \mathcal{G}$, and for all robots in the subteam, $R^n \in \mathcal{R}_s$. Again, by setting the subteam to be an single robot, $\mathcal{R}_s = R^n$, we can limit the amount of resource that individual robot consumes. Thus we can have resource constraints such as *Robot 3 can only test for chemicals in one large or two small rooms since it only carries two indicator kits.*

3.5.4 Expressive constraint language

In order to cover the full range of possible system constraints needed for search and rescue, we must be able to combine goal, robot, and resource constraints. We establish a constraint language that implements first order logic for the system constraints in $\mathcal{C}$.

Definition **3.4.** A **system constraint** is an element in $\mathcal{C}$ and is denoted as $\mathbf{c}_i$. A system constraint may either be a simple constraint or a complex constraint.

Definition **3.5.** A **simple constraint** is either a goal constraint (*before, equal, meets, overlaps, during, starts, finishes, do*), robot constraint (*participantIn, endAt*), or resource constraint (*limitFuelTo, limitResourcesOf*).

Definition **3.6.** A **complex constraint** consists of the composition of simple and complex constraints according to the operators listed below.

Before defining the logical operators and quantifiers, we first need to explain how constraints, which we are generally defined to always be true, can be terms in boolean logic. At the top level, the general definition of a constraint still holds; the outer most clause of a complex constraint expression must always be true. However, each nested constraint is conditional upon the value of the clause above it. If this clause is true, the nested constraint must be satisfied. If this clause is false, the nested constraint must be false. The exceptions to this are resource constraints and the *endAt* constraint which we define such that if the clause is false, the nested constraint is not required to be true.

- Logical not: $\neg\mathbf{c}_1$

  Example: Robot 1 may not work on goal 2: $\neg(R^1 \textit{participantIn } G^2)$

- Conditional statement: $\mathbf{c}_1 \rightarrow \mathbf{c}_2$

  Example: Unless a room is checked for chemicals (G^{c_1}) before looking for victims in that room (G^{s_1}), robot 1 cannot participate in the search goal since it cannot be decontaminated:

  $\neg(G^{c_1} \textit{ before } G^{s_1}) \rightarrow \neg(R^1 \textit{participantIn } G^{s_1})$

- If and only if: $\mathbf{c}_1 \leftrightarrow \mathbf{c}_2$

- Or expression: $\mathbf{c}_1 \vee \mathbf{c}_2$

  Example: Exploring the roof (G^{roof}) can either be achieved by sending a robot up the stairs (G^{stairs}) or having a helicopter fly over (G^{fly}).

  $\textit{do } G^{roof} \leftrightarrow \textit{do } G^{stairs} \vee \textit{do } G^{fly}$

- And expression:$\mathbf{c}_1 \wedge \mathbf{c}_2$

  Example: Only explore the shaft (G^{es}) if communication is relayed from the entrance (G^{rc}) when the robot reaches the bottom (the end of goal G^{es}).

  $(\textit{do } G^{es}) \leftrightarrow (\textit{do } G^{rc}) \wedge (G^{rc} \textit{finishes } G^{es})$

- xor expression: $\mathbf{c}_1 \oplus \mathbf{c}_2$

  Example: Since exploring a shaft is dangerous and we want to avoid losing all of our capable robots, robot 1 may not both work on that goal.

  $(R^2 \textit{participantIn } G^m) \oplus (R^1 \textit{participantIn } G^m)$

Goal Constraints	
G^x *before* G^y	$G_x\_start + d^x G_x < G_y\_start^*$ $G_x\_start^* \leq G_y\_start^*$
G^x *equal* G^y	$G_x\_start = G_y\_start$ $G_x d^x = G_y d^y$
G^x *meets* G^y	$G_x\_start + d^x G_x = G_y\_start$
G^x *overlaps* G^y	$G_x\_start < G_y\_start^*$ $G_x\_start + d^x G_x < G_y\_start^* + d^y G_y$ $G_x\_start^* + d^x > G_y\_start$ $G_x = G_y$
G^x *during* G^y	$G_x\_start^* > G_y\_start$ $G_x\_start + d^x G_x < G_y\_start^* + d^y G_y$ $G_x = G_y$
G^x *starts* G^y	$G_x\_start = G_y\_start$ $G_x\_start + d^x G_x < G_y\_start^* + d^y G_y$
G^x *finishes* G^y	$G_x\_start + d^x G_x = G_y\_start + d^y G_y$ $G_x\_start^* > G_y\_start$ $G_x = G_y$
do G^x	$G_x = 1$

Robot Constraints	
R^n *participantIn* G^m	$R_n G_m = G_m$
R^n *endAt* L	We create a new goal G_{end_L} $R_n G_{end_i} = 1$

Resource Constraints	
$\mathcal{R}_s$ *limitFuelTo* F	$\sum_{n \text{ s.t. } R^n \in \mathcal{R}_s} \sum_{i<\omega} R_n O_i\_travel \leq F$
$\mathcal{R}_s$ *limitResourcesOf* $\langle f(R^n, G^m), \rho \rangle$	$\sum_{R^n \in \mathcal{R}_s} \sum_{G^m \in \mathcal{G}} f(R^n, G^m) R_n G_m \leq \rho$

Table 1: COCoA implementation of simple constraints

- Forall quantifier: This quantifier operates with robots as variables, $\forall R^n \in \mathcal{R}, \mathbf{c}_1$, or goals as variables, $\forall G^m \in \mathcal{G}, \mathbf{c}_1$.
- Exists quantifier: This quantifier also operates with robots as variables, $\exists R^n \in \mathcal{R}_s, \mathbf{c}_1$, or with goals as variables, $\exists G^n \in \mathcal{G}_s, \mathbf{c}_1$.

3.5.5 Modeling constraints in COCoA

In general, goal constraints are defined to hold only if the goals are scheduled, allowing a trivial solution of not scheduling either goal. This is achieved by carefully combining goal variables and antivariables in the goal constraints. With the exception of the *begin* constraint, goal constraints are also defined so that scheduling one of the constrained goals forces the other constrained goal to be likewise scheduled. We assume that the duration, d^m, of goal G^m is greater than 0 and less than T_{max} for all goals. The simple constraints can be directly translated into problem shown in table 1.

While simple constraints can be modeled with variables in the problem formulation, complex constraints require the addition of new binary constraint variables, denoted C_x.

Complex constraints can be modeled by applying the following process beginning with the outer most expression and continuing in to the nested expressions.

1. If the constraint is a complex constraint introduce new binary constraint variables, denoted C_x, and constraints according to the rules for each complex expression listed below. The validity of each logical expression can be checked using truth tables:

Logical not $\neg \mathbf{c}_1$:

$C_{not} = 1 - C_1$

C_1	C_{not}
0	$C_{not} = \mathbf{1}$
1	$C_{not} = \mathbf{0}$

Conditional statement $\mathbf{c}_1 \rightarrow \mathbf{c}_2$:

$C_{if} \le C_2 - C_1 + 1$

$C_{if} \ge \frac{C_2 - C_1 + 1}{2}$

C_1	C_2	C_{if}
0	0	$C_{if} \le 1$ $C_{if} \ge \frac{1}{2}$ $C_{if} = \mathbf{1}$
0	1	$C_{if} \le 2$ $C_{if} \ge 1$ $C_{if} = \mathbf{1}$
1	0	$C_{if} \le 0$ $C_{if} \ge 0$ $C_{if} = \mathbf{0}$
1	1	$C_{if} \le 1$ $C_{if} \ge \frac{1}{2}$ $C_{if} = \mathbf{1}$

IF AND ONLY IF : $\mathbf{c}_1 \leftrightarrow \mathbf{c}_2$:

$C_{iff} \le C_2 - C_1 + 1$

$C_{iff} \le C_1 - C_2 + 1$

$C_{iff} \ge \frac{C_1 + C_2 - 1}{2}$

$C_{iff} \ge \frac{1 - C_1 - C_2}{2}$

C_1	C_2	C_{iff}
0	0	$C_{iff} \le 1$ $C_{iff} \le 1$ $C_{iff} \ge -\frac{1}{2}$ $C_{iff} \ge \frac{1}{2}$ $C_{iff} = \mathbf{1}$
0	1	$C_{iff} \le 2$ $C_{iff} \le 0$ $C_{iff} \ge 0$ $C_{iff} \ge 0$ $C_{iff} = \mathbf{0}$
1	0	$C_{iff} \le 0$ $C_{iff} \le 2$ $C_{iff} \ge 0$ $C_{iff} \ge 0$ $C_{iff} = \mathbf{0}$
1	1	$C_{iff} \le 1$ $C_{iff} \le 1$ $C_{iff} \ge \frac{1}{2}$ $C_{iff} \ge -\frac{1}{2}$ $C_{iff} = \mathbf{1}$

OR expression: $\mathbf{c}_1 \vee \mathbf{c}_2$

$C_{or} \le C_1 + C_2$

$C_{or} \ge C_1$

$C_{or} \ge C_2$

C_1	C_2	C_{or}
0	0	$C_{or} \leq 0$ $C_{or} \geq 0$ $C_{or} \geq 0$ $C_{or} = \mathbf{0}$
0	1	$C_{or} \leq 1$ $C_{or} \geq 0$ $C_{or} \geq 1$ $C_{or} = \mathbf{1}$
1	0	$C_{or} \leq 1$ $C_{or} \geq 1$ $C_{or} \geq 0$ $C_{or} = \mathbf{1}$
1	1	$C_{or} \leq 1$ $C_{or} \geq 1$ $C_{or} \geq 1$ $C_{or} = \mathbf{1}$

AND expression:$\mathbf{c}_1 \wedge \mathbf{c}_2$

$C_{and} \geq \frac{C_1+C_2-1}{2}$

$C_{and} \leq C_1$

$C_{and} \leq C_2$

C_1	C_2	C_{and}
0	0	$C_{and} \geq -\frac{1}{2}$ $C_{and} \leq 0$ $C_{and} \leq 0$ $C_{and} = \mathbf{0}$
0	1	$C_{and} \geq 0$ $C_{and} \leq 0$ $C_{and} \leq 1$ $C_{and} = \mathbf{0}$
1	0	$C_{and} \geq 0$ $C_{and} \leq 1$ $C_{and} \leq 0$ $C_{and} = \mathbf{0}$
1	1	$C_{and} \geq \frac{1}{2}$ $C_{and} \leq 1$ $C_{and} \leq 1$ $C_{and} = \mathbf{1}$

XOR expression: $\mathbf{c}_1 \oplus \mathbf{c}_2$

$C_{xor} \geq C_1 - C_2$

$C_{xor} \geq C_2 - C_1$

$C_{xor} \leq C_1 + C_2$

$C_{xor} \leq 2 - C_1 - C_2$

C_1	C_2	C_{xor}
0	0	$C_{xor} \geq 0$ $C_{xor} \geq 0$ $C_{xor} \leq 0$ $C_{xor} \leq 2$ $C_{xor} = \mathbf{0}$
0	1	$C_{xor} \geq -1$ $C_{xor} \geq 1$ $C_{xor} \leq 1$ $C_{xor} \leq 1$ $C_{xor} = \mathbf{1}$
0	1	$C_{xor} \geq 1$ $C_{xor} \geq -1$ $C_{xor} \leq 1$ $C_{xor} \leq 1$ $C_{xor} = \mathbf{1}$
1	1	$C_{xor} \geq 0$ $C_{xor} \geq 0$ $C_{xor} \leq 2$ $C_{xor} \leq 0$ $C_{xor} = \mathbf{0}$

Forall quantifier: This quantifier operates with robots or goals as variables.

We create a binary constraint variable, $C_{\forall_i}$, for each system constraint, $\mathbf{c}_{R^n}$ or $\mathbf{c}_{G^m}$. These system constraints must be formulated using this process like any other complex constraint. Additionally, we have the following problem constraints where n is the number of robots or goals:

$C_\forall \geq 1 - n + \sum_i C_{\forall_i}$

$C_\forall \leq \frac{\sum_i C_{\forall_i}}{n}$

$\sum_i C_{\forall_i}$	$C_\forall$
$[0, n-1]$	$C_\forall \geq 0$ $C_\forall < 1$ $C_\forall = \mathbf{0}$
n	$C_\forall \geq 1$ $C_\forall \leq 1$ $C_\forall = \mathbf{1}$

Exists quantifier: This quantifier also operates with robots as variables.

We create a binary constraint variable, $C_{\exists_i}$, for each system constraint, $\mathbf{c}_{R^n}$ or $\mathbf{c}_{G^m}$. These system constraints must be formulated using this process like any other complex constraint. Each of these system constraints also results in a problem constraint, $C_\exists \geq C_{\exists_i}$, so that the *exists* constraint evaluates as true if any of the individual system constraints are true. Additionally, we add the following problem constraints so that the *exists* constraint does not evaluate to true unless one or more of the individual constraints is true:

$C_\exists \leq \sum_i C_{\exists_i}$.

$\forall i : C_\exists \geq C_{\exists_i}$

$\sum_i C_{\exists_i}$	$C_\exists$
0	$C_\exists \leq 0$ $C_\exists \geq 0$ $C_\exists = \mathbf{0}$
[1,n]	$C_\exists \leq 1$ for some i: $C_\exists \geq 1$ $C_\exists = \mathbf{1}$

Algorithm 3 Generate problem constraints for $C_x \rightarrow \mathbf{c}_i$

```
Use the constraints from table 1 that correspond to $\mathbf{c}_i$.
Turn all equalities, $x = y$, into inequalities: $x \leq y$ and $x \geq y$.
Rearrange each inequality so that the RHS = 0
if LHS $\leq 0$ or LHS $< 0$ then
    RHS = $(1 - C_x)T_{max}$
else
    RHS = $(C_x - 1)T_{max}$
end if
```

Algorithm 4 Generate problem constraints for $\neg C_x \rightarrow \neg \mathbf{c}_i$

```
Use the constraints from table 1 that correspond to $\mathbf{c}_i$.
Turn all equalities, $x = y$, into inequalities: $x \leq y$ and $x \geq y$.
Rearrange each inequality so that the RHS = 0
for all resulting inequality problem constraints do
    Create a new binary variable $C_{x_i}$.
    if LHS $< 0$ then
        LHS $\geq (C_{x_i} - 1)T_{max}$
    else if LHS $\leq 0$ then
        LHS $> (C_{x_i} - 1)T_{max}$
    else if LHS $> 0$ then
        LHS $\leq (1 - C_{x_i})T_{max}$
    else
        LHS $< (1 - C_{x_i})T_{max}$
    end if
end for
Add problem constraint: $\sum_i C_{x_i} + \sum_{G_m \in \mathbf{c}_i} (1 - G_m) \geq (1 - C_x)$
```

2. For each constraint clause, $\mathbf{c}_i$, continue to apply the rules for modeling complex expressions.

3. Simple constraints that are nested inside a complex constraint cannot be directly incorporated into the MILP as non-nested simple constraints but must be formulated as dependent upon the constraint variable, C_x, of the expression in which they are nested. Algorithm 3 describes how to generate the problem constraints so that $C_x \rightarrow \mathbf{c}_i$. Note that LHS and RHS refer to the Left (or Right) Hand Side of the inequality.

4. Recall from the definition of a complex constraint that resource constraints and the *endAt* constraint are defined such that $\neg \mathbf{c}_i$ means that the constraint is not *required* while all other simple constraints are defined such that $\neg \mathbf{c}_i$ means that the constraint is not *permitted*. These other simple constraint need to be correctly modeled such that $\neg C_x \rightarrow \neg \mathbf{c}_i$. This is accomplished by applying DeMorgan's law ($\neg(A \wedge B) = \neg A \vee \neg B$) and following algorithm 4. The final problem constraint forces at least one of the constraints to be false if $C_x = 0$ unless one or more of the goals are not scheduled in which case $\mathbf{c}_i$ is trivially false.

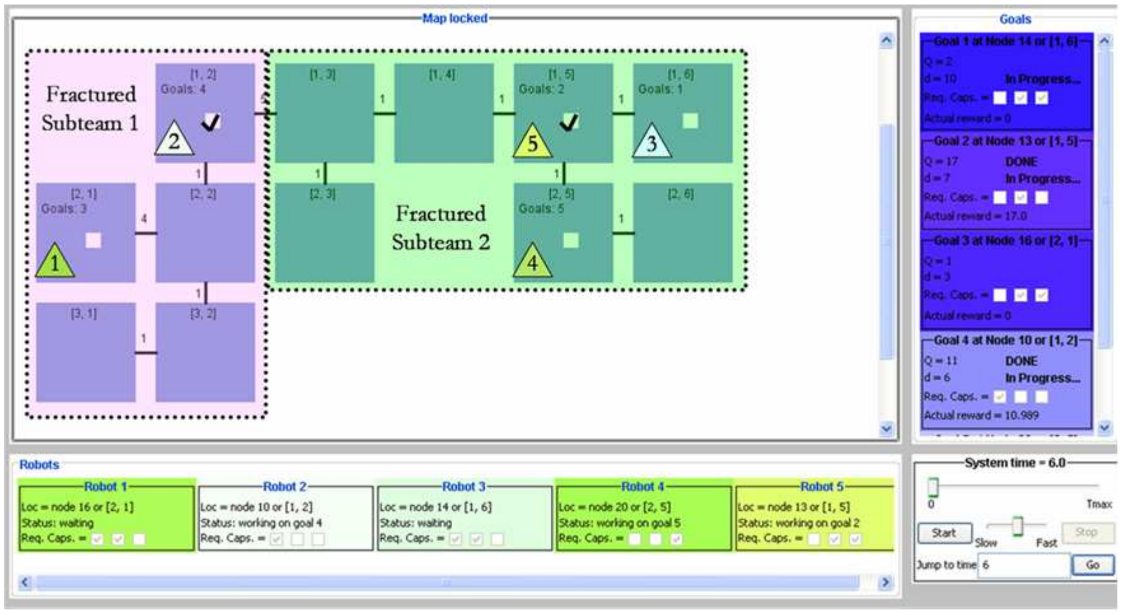
5. Set the top level constraint variable $C_x = 1$.

We can quickly verify that the process for establishing $C_x \rightarrow \mathbf{c}_i$ is valid (Algorithm 3): if $C_x = 0$, we have the left hand side of the expression, LHS $\{>, \geq\} - T_{max}$ or LHS $\{<, \leq\} T_{max}$. Since all times are bounded by T_{max}, this is a trivial constraint. On the other hand, if $C_x = 1$, the right hand side is 0 which is simply a restatement of the original constraint.

Similarly, we can validate $\neg C_x \rightarrow \neg \mathbf{c}_i$ (Algorithm 4). If $C_x = 1$, the final problem constraint is trivially satisfied since the addition of binary variables will always be greater than or equal to 0 and there are no constraints on any of C_{x_i} or G_m. We have defined C_{x_i} to represent whether constraint clause i is false. If

$C_x = 0$, one or more constraint clauses must be false ($C_{x_i} = 1$ for some i) causing the entire constraint to be false by DeMorgan's law (or as we have defined our constraints, one or more of the goals in the constraint may be unscheduled). We can verify that if $C_{x_i} = 1$ then the constraint is false. Each true constraint is put into the form, LHS $\{<, \leq, >, \geq\}0$. This means that, in order for the constraint to be false, LHS $\{\geq, >, \leq, <\}0$. If $C_{x_i} = 1$, these conditions hold. However, if $C_{x_i} = 0$, the constraints stand as LHS $\{\geq, >\} - T_{max}$ or LHS $\{\leq, <\}T_{max}$ which is trivially true. Therefore, $C_{x_i} = 1$ indicates that one element of the constraint is false which indicates by DeMorgan's law that the entire constraint is false so $\neg C_x \rightarrow \neg \mathbf{c}_i$.

3.6 Distributed Replanning



Goal 1 at Node 14 or [1, 6]
Q = 2
d = 10 In Progress...
Req. Caps. = ☐ ✓ ✓
Actual reward = 0

Goal 2 at Node 13 or [1, 5]
Q = 17 DONE
d = 7 In Progress...
Req. Caps. = ☐ ✓ ☐
Actual reward = 17.0

Goal 3 at Node 16 or [2, 1]
Q = 1
d = 3
Req. Caps. = ☐ ✓ ✓
Actual reward = 0

Goal 4 at Node 10 or [1, 2]
Q = 11 DONE
d = 6 In Progress...
Req. Caps. = ✓ ☐ ☐
Actual reward = 10.989

Robot 1
Loc = node 16 or [2, 1]
Status: waiting
Req. Caps. = ✓ ✓ ☐

Robot 2
Loc = node 10 or [1, 2]
Status: working on goal 4
Req. Caps. = ✓ ☐ ☐

Robot 3
Loc = node 14 or [1, 6]
Status: waiting
Req. Caps. = ✓ ✓ ☐

Robot 4
Loc = node 20 or [2, 5]
Status: working on goal 5
Req. Caps. = ☐ ☐ ✓

Robot 5
Loc = node 13 or [1, 5]
Status: working on goal 2
Req. Caps. = ☐ ✓ ✓

System time = 6.0

Jump to time 6

Figure 9: Abstract simulator with small example problem. Robots are illustrated as triangles and fractured subteams are illustrated by bounding boxes.

The necessity of graceful degradation of performance in the face of communication failures is acknowledged by the multirobot community [24]. However, many coordination systems assume perfect communication between team members. Those systems that do analyze communication failure assume that an arbitrary message will fail with a uniform probability. This model fails to capture the true nature of communication breakdowns in multirobot teams.

Robots generally depend on peer to peer or ad hoc wireless RF networks for communication. Work in the networking community [3], [63] indicates that the signal strength between two robots varies according to $\frac{1}{d^n}$ where d is the distance between the robots and n is the path loss exponent that is dependent on the environment. Therefore, the ability of two robots to communicate varies depending on their proximity and environmental factors and so is systematic. Since robot communication is frequently based on TCP which is a reliable protocol, dropped packets are relatively unimportant. If robots are sufficiently close, they will be able to communicate reliably and otherwise they will be unable to communicate at all.

The networking community has extensively analyzed communication through message relays in connected and disconnected ad hoc networks [51], [63]. However, these models have not yet been applied to communication in the multirobot coordination domain. We introduce the formalism of *fractured subteams* which captures the elements important for multirobot planning while abstracting away the mechanics of the ad hoc network:

DEFINITION **3.7.** A **fractured subteam** consists of a set of robots $\mathcal{R}_F \subseteq \mathcal{R}$ that

1. Jointly believe in a common problem,

$\rho_F = \langle \mathcal{R}, \mathcal{G}, E, \mathcal{C}, T_{max} \rangle$

2. Share a team plan $\mathcal{T}_F = \{\mathcal{S}_{R^1}, \mathcal{S}_{R^2}, ...\mathcal{S}_{R^n}\}$
3. Share a *belief evolution model*, $\mathcal{B}$, defined in definition 4.2
4. Are able to communicate with each other through transitive closure
5. Are unable to communicate with any robot not in $\mathcal{R}_{\mathcal{F}}$

DEFINITION **3.8.** A **belief evolution model**, denoted $\mathcal{B}$, consists of the set of tuples each containing a time critical tight coordination team planning problem, a team plan, and the set of robots known to have shared this model: $\mathcal{B} = \{\langle \rho_0, \mathcal{T}_0, \mathcal{R} \rangle, \langle \rho_1, \mathcal{T}_2, \mathcal{R}_{F_i} \rangle, ...\langle \rho_n, \mathcal{T}_n, \mathcal{R}_{F_j} \rangle\}$ Since we assume that all robots start with a common problem and team plan, all robots' belief evolution models contain a common element.

It follows from this definition that a robot always belongs to exactly one fractured subteam. A simple example of how a team may be partitioned into fractured subteams based on the topology of the environment is shown in figure 9 where robots 1 and 2 belong to one fractured subteam and robots 3, 4 and 5 belong to a second fractured subteam. If robot 2 crossed into node $[1, 3]$, it would become part of the second fractured subteam and would have to merge its problem and team plan with robots 3, 4 and 5. The beliefs and plan of robot 1 would remain unchanged. Fractured subteams are themselves another variable in the system which must be considered while replanning. At the same time, fractured subteams force distributed replanning.

3.6.1 Distributed coordination across fractured subteams

As robots seek to execute their team plan in dangerous and uncertain environments, they may discover that the problem and environment which they encounter does not match the problem for which they planned. If this occurs, they must refine their model of the problem and replan. There are two main challenges in replanning. First, since robots are operating in time sensitive domains, the replanning must be performed quickly since planning itself delays execution. The second problem arises when failures in communication prevent the refined problem and solution from being disseminated to all team members resulting in inconsistencies. Before addressing our solution to these challenges, we identify the set of replanning catalysts or discoveries that necessitate replanning based as shown in table 2.

The discovery of replanning catalysts is beyond the scope of this paper but is covered in part in other work [24].

COCoA is capable of efficiently generating optimal or near optimal team plans for a given time critical tight coordination team planning problem. We extend COCoA from a purely centralized solver to a hybrid architecture. This hybrid architecture employs opportunistic centralization but is fundamentally distributed in that each fractured subteam is unable to communicate with other fractured subteams requiring distributed replanning. Each fractured subteam maintains its own knowledge from all previous subteams encountered which it uses for this replanning.

This extension to COCoA has three main components. First, we have extended the architecture itself beyond the master-slave model embedded in the centralized architecture. Next, we provide mechanisms for distributed fractured subteams with independent beliefs and plans to split and merge. The final key to our successful hybrid architecture is that we explicitly reason about communication failure and fractured subteam composition during the opportunistically centralized replanning.

3.6.2 Robustness through redundancy

Since COCoA was designed to be a centralized planner, only a single robot was required to possess the planning capabilities and complete problem and team plan. Other than this centralized planning robot, robots required only their individual tightly coupled schedule. In the distributed version of COCoA, each robot possesses full planning capabilities and a copy of the complete problem and team plan. The increase in communication cost is marginal if information is broadcast. When a replanning catalyst is discovered, the architecture leverages the nature of a fractured subteam for opportunistic centralization. A robot from that fractured subteam is elected to replan and broadcast the results to the subteam.

$\mathcal{R}$: Robot	added removed incapacitated capability added capability removed
$\mathcal{G}$: Goal	added removed requirement added requirement removed reward increased reward decreased duration increased duration decreased
E: Environment path cost	shortened lengthened
$\mathcal{C}$: System constraint	added removed
T_{max}: Time limit	shortened extended

Table 2: Replanning catalysts are divided into categories based on on our definition of a time critical tight coordination team planning problem (Definition 3.1)

3.6.3 Dynamic fractured subteam formation

Although the replanning is locally centralized, COCoA is distributed with respect to the individual fractured subteams. Each fractured subteam maintains its own problem and team plan. Each robot carries with it the complete knowledge of the fractured subteam to which it belongs. When robots from two fractured subteams are merged into a single fractured subteam, their problems and team plans must likewise be merged. Merging different beliefs is a deep research problem that we intend to address in more detail in future work. Presently, we use a simple approach for merging two fractured subteams, F_1 and F_2 described below:

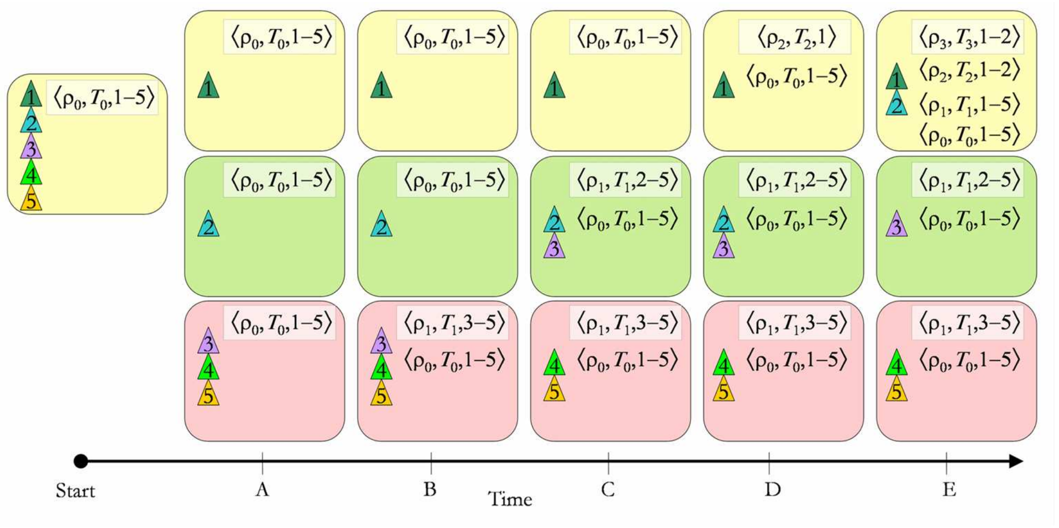
1. If the currently held problem and team plan of F_1 match the currently held problem and team plan of F_2, no changes need to be made. The membership of the new fractured subteam is set to the union of the membership of F_1 and F_2.

2. If the currently held problem and team plan of F_1 match a state in the belief evolution model of F_2 and the team plan of F_2 is valid given the current status of the members of F_1 and F_2, all members of the merged fractured subteam adopt the problem, team plan, and belief evolution model of F_2 (timestep C in figure 10).

3. By symmetry, the previous rule applies if the currently held problem and team plan of F_2 match a state in the belief evolution model of F_1.

4. If none of the above hold, create a new problem by merging the last common state in the belief evolution models of F_1 and F_2 with the differences between the common state and the problem of F_1, the differences between the common state and the problem of F_2, and the current status of the members of the merged fractured subteam. Generate a new team plan to the resulting problem. Update the belief evolution model of the merged fractured subteam to include the union of the belief evolution models for F_1 and F_2.

The merge process is illustrated on a sample problem in figure 10. The process of splitting a single fractured subteam into multiple fractured subteams is likewise illustrated in figure 10 and is much simpler than the merge process. Each new fractured subteam maintains its problem, team plan, and belief evolution model.



A:	Communication failures result in 3 fractured subteams
B:	Robot 4 in fractured subteam 3 is disabled resulting in replanning.
C:	Robot 3 moves from fractured subteam 3 to 2 resulting in a merge and the dissemination of robot 4's failure.
D:	Fractured subteam 1 is informed of a new goal and replans.
E:	Robot 2 moves from fractured subteam 2 to 1; the merge requires replanning.

Figure 10: Example splitting and merging of fractured subteams. Rounded boxes indicate fractured subteams while triangles indicate robots. The belief evolution models for each subteam are shown with the robots for that model indicated as a range (i.e. $\langle \rho_0, \mathcal{T}_0, 1-5 \rangle$ indicates robots 1-5 (R^1, R^2, R^3, R^4, R^5) shared the initial problem ρ_0 and team plan $\mathcal{T}_0$).

3.6.4 Replanning with Selective Disruption Minimization

While full replanning works well in perfect communication, it tends to perform poorly with fractured subteams. To illustrate this problem, consider a simple example with three robots. Robots 1 and 2 form one fractured subteam while robot 3 is in a second fractured subteam. In the original plan, robot 2 was to perform a certain goal. Before robot 2 was able to complete the goal, it suffers a breakdown. Robot 1 observes this and replans. If robot 3 is closer to the goal than robot 1, the goal will be allocated to robot 3 while robot 1 remains idle. The problem arises because robot 3 is out of communication and unaware of its new commitment. If robot 1 had considered the communication failure while replanning, system performance would have been significantly better even though allocating robot 1 to the goal is suboptimal by our original objective function.

We explicitly reason about communication failure and fractured subteam composition during replanning and prefer solutions that only change the commitments of robots in the fractured subteam. The changes in coordination commitments are known as *disruption* [6]. Our approach is inspired by Bartold and Durfee's work for disruption minimization for a specific class of multiagent coordination problems based on weighting changes to blocking and synchronization commitments. Since the commitments in the time critical tight coordination team planning problem are significantly different than the multiagent coordination problem in [6], we have developed our own metrics for measuring disruption. Our work also differs from previous work in that only the disruption of robots external to the replanning fractured subteam is minimized.

Before quantifying change, we must first establish coordination commitments. In general, every goal assigned to a robot represents a commitment to the team. However, not all changes to commitments are undesirable. If a robot is in the fractured subteam and aware of the replanning catalyst, some changes in the robot's schedule are advantageous as they increase system performance. Since the problem is tightly coupled, distinguishing between commitments that should be preserved and those that may be modified is challenging. We use *commitment graphs* described in Algorithm 5 and shown in figure 11 to determine where disruption should be minimized. Note that at this time we only consider disruption of temporal goal

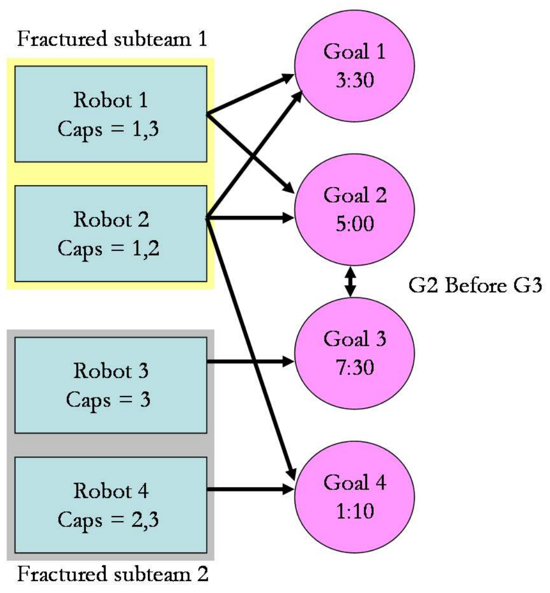


Figure 11: Sample commitment graph. Robot coordination commitments are omitted on the edges for clarity.

constraints (based on Allen's 13 temporal relationships [2]) and do not model disruption of resource or robot constraints.

One of the advantages of the MILP problem formulation in COCoA is the relative ease in combining multiple objective functions. We selectively minimize disruption by adding a cost function. There are several possible cost functions. We penalize near term changes more than long term changes since there is a higher probability that changes in the plan will be discovered in time for the robots to respond if the change is sufficiently far in the future. We then define our disruption minimization objective function as follows:

$$Minimize: \sum_{m \text{ s.t. } G^m \in \mathcal{G}} (Cost(G^m) + \sum_{n \text{ s.t. } R^n \in \mathcal{R}} Cost(R^n, G^m))$$

The cost function for each goal $Cost(G^m)$ is dependent upon the robots allocated to the goal represented by the commitment graph, the original time at which the goal was scheduled, *oldTime*, the new time at which the goal is to be scheduled (a variable in the MILP), *newTime*, and the system time when replanning, *replanTime*. The cost function for the goal can be constructed based on the following rules:

Algorithm 5 Build commitment graph (figure 11)

```
1: for all Goals $G^m \in \mathcal{G}$ such that $isScheduled(G^m)$ do
2:     Add goal node to graph
3: end for
4: for all Robots $R^n \in \mathcal{R}$ do
5:     Add robot node to graph
6:     $\mathcal{S}_{R^n}$ = last known schedule for $R^n$ in belief evolution model
7:     for all Steps in this robot's schedule, $S_i \in \mathcal{S}_{R^n}$ do
8:         Add edge with robot coordination commitments, $\mathcal{D}$, of $S_i$ from robot node to goal $G^i$
9:     end for
10: end for
11: for all Temporal goal constraints $C_j \in \mathcal{C}$ do
12:     Add edge $C_j$ between constrained goal nodes
13: end for
```

1. If G^m contains no edges between it and robots not in the subteam and the set of goal nodes reachable from goal node G^m contain no edges between them and robots not in the subteam, the cost is 0 (Goal 1 in figure 11).

2. If G^m is not scheduled and so has no node in the commitment graph, the cost is 0.

3. If G^m contains no edges between it and robots not in the subteam but is reachable from goal nodes with these links, the cost must be formulated to match the temporal goal constraints. We leave the description of these cost functions for future work. Intuitively, however, the cost must match the nature of the constraint. In the example in figure 11, scheduling goal 2 earlier than 5:00 would have no penalty while scheduling it later would incur a penalty of

$$\alpha \frac{G_2\_start - 5:00}{T_{max}}$$

4. If G^m has edges linking it to nodes of robots not in the fractured subteam (goals 3 and 4 in figure 11) then if the old start time equals the new start time, the cost is 0. If the goal is cancelled, the cost is a constant:

$$\alpha_{cancel}(1 - \frac{oldTime - replanTime}{T_{max} - replanTime})$$

   Otherwise the cost equals

$$\alpha_{insert}(1 - \frac{newTime - replanTime}{T_{max} - replanTime})$$

The cost function for each robot $Cost(R^n, G^m)$ can similarly be constructed by following simple rules:

1. If R^n is a member of the fractured subteam the cost is 0.

2. If R^n is not a member of the fractured subteam and is not currently allocated to G^m (independent of whether or not G^m is allocated), the cost is depends on whether the task is scheduled in the near term or long term:

$$\beta_{insert}(1 - \frac{newTime - replanTime}{T_{max} - replanTime})$$

3. If R^n is not a member of the fractured subteam but is currently allocated to G^m, the cost is 0 if the new time is the same as the previous time and R^n is still assigned to the goal. If R^n is not assigned to G^m in the new plan, the cost is a constant:

$$\beta_{cancel}(1 - \frac{oldTime - replanTime}{T_{max} - replanTime})$$

   Otherwise the cost is determined by:

$$\begin{array}{c}\beta_{cancel}(1 - \frac{oldTime - replanTime}{T_{max} - replanTime}) + \\ \beta_{insert}(1 - \frac{newTime - replanTime}{T_{max} - replanTime})\end{array}$$

The constants α and β may be tuned depending on the desired emphasis.
The new objective function can be combined with the existing objective function:

$$\begin{array}{c}Maximize: \sum_{m \text{ s.t. } G^m \in \mathcal{G}} \frac{T_{max} - G_m\_start^*}{T_{max}} Q^m - \\ \sum_{m \text{ s.t. } G^m \in \mathcal{G}} (Cost(G^m) + \sum_{n \text{ s.t. } R^n \in \mathcal{R}} Cost(R^n, G^m))\end{array}$$

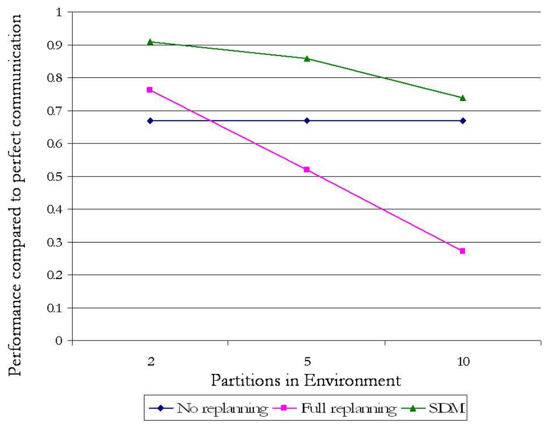


Figure 12: Comparison of no replanning, full replanning, and replanning with selective disruption minimization on a sample environment with 5 robots and 10 goals. One robot was disabled at an early randomly selected time. Strategies for replanning include *no replanning*, *full replanning*, and *selective disruption minimization* (SDM).

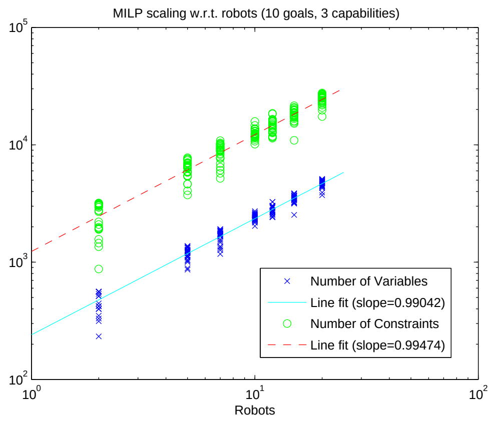
3.6.5 Evaluation

We conducted experiments to analyze the effect of fractured subteams on system performance. We used an abstract simulator (figure 9) with a randomly generated environment, 5 robots, and 10 goals. The environments are randomly partitioned into various blackout zones which impose the communication limitations, resulting in a set of fractured subteams. The number of fractured subteams is dependent on the topology of the environment and the composition of these subteams changes as robots traverse the environment.As an starting point for understanding the effect of communication breakdown on performance, we conducted a simple set of experiments in which one robot in the team was randomly disabled at some random time early in the plan.

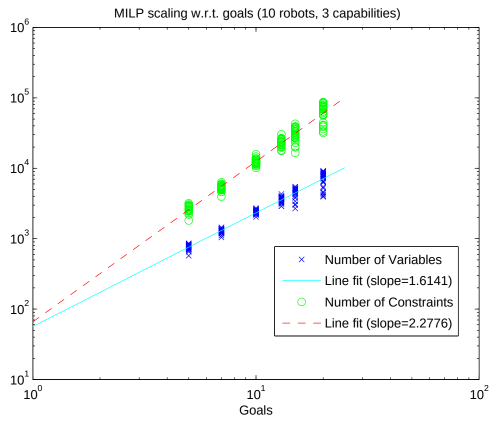
We compared three different replanning strategies. In the first strategy, *no replanning*, robots attempt to execute their original tightly coupled schedules despite knowledge that the problem is out of date. A second possible strategy is *full replanning* in which robots replan without consideration of communication failure or fractured subteam composition. Inefficiencies may arise since some robots on the team are unaware of the new team plan. The final strategy we consider is *replanning with selective disruption minimization* as described in Section 3.6.4. The results vary depending on the topology of the environment. The results for a single typical environment are shown in figure 12. As expected, no replanning performs relatively poorly since the tasks assigned to the disabled robot are never accomplished. If the communication failures are minimal, full replanning outperforms the no replanning strategy, but as violations of the implicit assumption of full communication in the full replanning strategy increase, full replanning actually decreases team performance. Finally, by applying selective disruption minimization, we see that the degradation of performance with communication failure is significantly improved.

3.7 Limitations

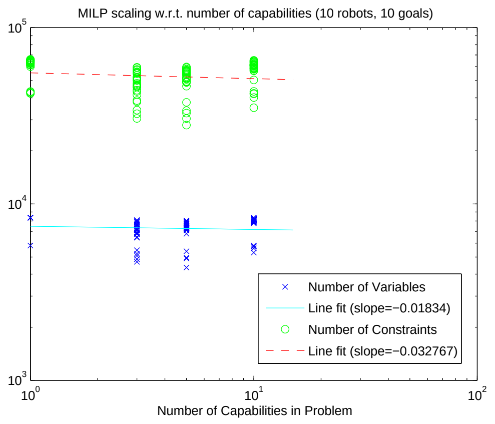
Our approach of using a mathematical programming model for coordination has a number of limitations. One of the strengths of modeling the problem as a MILP is that we can leverage a vast amount of work in solving these problems including the commercially available solver, CPLEX [40]. This strength is also a weakness since CPLEX is expensive. The cost of putting it on a number of robots may be prohibitive. We experimented with some free ILP solvers including GLPK (GNU Linear Programming Kit) [36] and lp_solve [52] though neither package performed nearly as well as CPLEX on mixed integer problems. The dependency on a linear programming package as well as the increased computational complexity make COCoA inappropriate for domains without optimality demands or without joint goals allowing task allocation and scheduling to be



(a) Effect of number of robots on MILP model size



(b) Effect of number of goals on MILP model size



(c) Effect of number of capabilities on MILP model size

Figure 13: To determine how the MILP problem formulation scales we randomly generated over 10,000 environments. Problem size is characterized by the number of variables and constraints in the MILP model. We fit a line to each set of data points in log-log space using least squares best fit. (a) Model size grows linearly with the number of robots in the problem. (b) Model size grows superlinearly with the number of goals in the problem. (c) Model size is independent of the number of capabilities relevant to the domain.

more easily decomposed.

This thesis proposal only addresses the team planning problem. We do not model agents as having individual rewards. This means that COCoA is not applicable to adversarial domains or even multiagent domains with self interested agents. Our assumption that agents do not have hidden preferences allows us to ignore the issue of privacy.

The goal oriented problem representation allows us to preprocess path planning since each robot trajectory is abstracted to a single variable, the amount of time allocated to travel to a goal. This abstraction allows us to precompute path planning but limits the system's ability to reason about or maintain spatial relationships between robots as in the Hoplites framework [45].

There are a number of limitations to the problem description that do not represent fundamental limitations of the mathematical programming model but rather design decisions. For example, we assume that goal rewards decrease linearly with time. Instead, rewards could be modeled as piecewise linear or quadratic functions. We chose our representation of goal rewards to speed up the MIP solver. Similarly, we assume that all robot capabilities are binary. This is a reasonable assumption for robot teams where robots either have a sensor or not but would not accurately model humans, for example, where some people are more skilled at accomplishing a task that others. Presumably as robotic technology advances, this discretization of robot capability will become less appropriate. Fortunately, it is possible to incorporate real valued capabilities into the objective function so that reward depends on the level of capability of the robot working on the goal. The reward for each goal would then be

$$\frac{T_{max} - G_m\_start^*}{T_{max}} Q^m S_{bot}$$

where S_{bot} is the skill of the robot performing the task. We decided not to include this in our model since the resulting problem is a quadratic rather than a linear programming problem. The same applies to goal duration; currently we require the requirements for the goal be satisfied for the entire duration. It is possible to weight reward according to the amount of time or energy dedicated to the goal.

In order to guarantee optimality, COCoA plans over all team members and all goals. While the problem formulation is invariant to changes in the environment size and the time limit, T_{max}, we want to characterize the dependence on the number of robots, goals, and capabilities in the system. It is challenging to analyze the complexity of a MILP problem since performance is very dependent on the formulation. Adding additional constraints can sometimes improve system performance. In general, however, problems with more variables and constraints are more challenging to solve. Therefore, we analyze the problem space in terms of the number of variables and constraints in the MILP problem. Analytically, we would expect the problem size to grow linearly with the number of robots in the problem since the problem formulation has a constant number of variables per robot given some fixed number of goals and capabilities. We expect the problem size to grow cubically (worst case) with the number of goals in the system since our problem definition includes a binary variable $R_nO_iP_{(g_1,g_2)}$ for every plan step (which is limited by the number of goals in the system) and for every possible pairing of goals. Since there are no variables relating to the capabilities in the problem, we would expect the problem size to be independent of the number of relevant capabilities.

We randomly generated over 10,000 benchmarks with variable numbers of robots, goals, and capabilities and tallied the number of variables and constraints. Since we expected the results to be polynomial, we created log-log plots and used the least squares metric to find the best fit of a line to the data as shown in figure 13. The slope of the line in log-log space represents the degree of the polynomial. The data in figure 13(a) confirms our hypothesis that problem size grows linearly with the number of robots in the problem since the slope of the lines is approximately 1. Figure 13(b) indicates that although problem size doe not grow as rapidly with the number of goals in the problem as feared, the relationship is still superlinear. Adding more goals to the problem has a greater effect on problem complexity that adding more goals. The reason that the number of variables and constraints do not grow cubically with the number of goals in the problem is that we prune combinations of robots and goals that would never be allocated which also frequently allows us to lower the maximum number of plan steps for any robot. Finally, figure 13(c) matches our analysis of the effect of the number of relevant capabilities in the problem. The number of variables and constraints in the problem remain essentially constant as the number of capabilities increases (there is actually a slight decrease in both the number of constraints and variables due to more aggressive pruning of infeasible allocations between robots and goals).

4 Proposed Work

4.1 Intelligent plan merging

As described in section 3.6, one component of a distributed implementation of COCoA is merging fractured subteams. We described a simple approach for merging fractured subteams in section 3.6.1. This naive approach may result in inefficiencies as frequent replanning is necessary. It may be possible to resolve the plans without this full replanning (step 4) by intelligently merging the beliefs together. The old plans may be useful in generating the new plan when full replanning is necessary. In either case, efficient plan merging is a necessary component of a fully functional coordination architecture and will be addressed by the proposed thesis.

4.2 Temporal constraint reasoning

While the myopic and greedy goals heuristics are useful in the absence of the system constraints described in section 3.5, these heuristics are inadequate for tightly constrained systems. If the heuristics are unable to provide a feasible solution, the coordination architecture must rely entirely on domain independent integer linear programming solution techniques. In order to improve performance on problems with goal, resource, and robot constraints, we plan to develop heuristics based on temporal constraint reasoning to efficiently generate feasible solutions.

4.3 Robust plan selection

Thus far, this work has assumed that the environment is static for planning purposes. It handles dynamic events by responding to the replanning catalyst. We postulate that system performance may be improved if certain replanning catalysts are anticipated. For example, if goals are arriving randomly in some environment, we would expect performance to improve if robots that are currently unemployed move to some central location where they can more easily be deployed when a task arrives. As a second example, if robots in a system frequently fail, performance may be improved if extra robots are allocated to an important goal so that even when one robot fails, the goal can still be accomplished.

Reasoning about dynamic events to improve performance has two components. The system must obtain models of these events. We propose to address this trivially by providing the system with these models. Models of robot failure, goal generation, and other replanning catalysts could be learned by the system over many runs of simulation. This topic, however, is not central to the thesis and so is left for future work.

Given these models, the system must somehow incorporate them into the planning. We propose to do this by defining a second objective function for plan robustness. Plans with extra travel time or that allocate extra robots to a goal would be robust to certain types of failure. Based on the models, this new objective function would be combined with the original objective function. This approach allows the plans to be incorporated in a principled manner. The system can explicitly tradeoff between more optimal and safer plans.

4.4 Communication

Communication is a necessary yet presently unaddressed component to our coordination architecture. The ad hoc approaches to communication favored by the robotics community with specially defined protocols are generally not easily readable by humans nor are they standard which limits interoperability between teams. Instead, we adopt a standard agent communication language (ACL). KQML [30] and FIPA-ACL [31] are two of the most widely used ACLs in multi-agent systems. We propose adopting the RETSINA communicator [72] which uses KQML for communication in our system. By adopting a standard language, we facilitate interoperability and interconnection between robot teams. A standard language, however, is insufficient if the agents do not have a shared vocabulary or ontology. The proposed thesis will provide an ontology for coordination within a constraint optimization framework.

While agent communication languages are well suited to exchange of high level information relating to goal allocation, replanning catalysts, or subteam composition, for example, a multirobot system interacting with a dynamic physical environment must also send low level information (e.g. telemetry or video data). These low level communications often occur at high frequencies, which can clog the main line of communication. We have extended the RETSINA architecture to allow multiple lines of communication between agents [46]. The additional lines, or backchannels, are for the transfer of low level information. The proposed thesis will also provide a set of backchannels designed for the search and rescue domain but applicable to a range of other multirobot applications.

Finally, the proposed thesis will address the questions of *what*, *when*, and *with whom* to communicate. We will analyze the effect of eager versus hesitant communication policies where robots vary the amount of information they share with their teammates.

4.5 Evaluation in USARsim

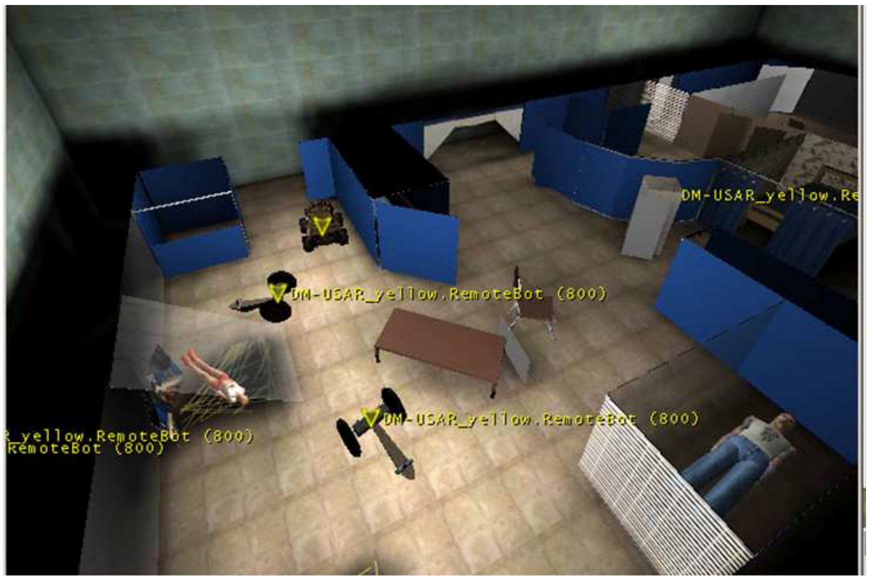

DM-USAR_yellow.Re
DM-USAR_yellow.RemoteBot (800)
DM-USAR_yellow.RemoteBot (800)
_yellow.RemoteBot (800)
RemoteBot (800)

Figure 14: A team of heterogeneous robots explore a high fidelity model of the yellow NIST disaster arena in the USARsim simulator.

Thus far, all experiments have been carried out in COCoASim. While sufficient to illustrate the concepts and algorithms described in this thesis proposal, an abstract simulator may inadvertently hide issues that prevent the coordination architecture from being applicable to real systems. Since our research is motivated by the urban search and rescue domain, it is appropriate to demonstrate our coordination architecture in this domain.

While the ultimate goal of providing multirobot teams capable of interacting with humans at various levels of autonomy to search a disaster environment is still years away, NIST has developed the RoboCup Rescue competition to advance research in this area [41] [58].

However, the scarcity and expense of robots capable of autonomously navigating a complex environment limit coordination research on real robots. A high fidelity, extensible simulator, USARsim, has been developed to facilitate this research [81] [82] [16]. A screenshot of the simulator is shown in figure 14. Based on Epic Games' UnrealEngine2 [78], USARsim includes models of rescue robots used by teams competing in the RoboCup Rescue competition including the Pioneer P2AT and P2DX, the iRobot ATRV-Jr, the Personal Exploration Rover, a robot built by our RoboCup Rescue team called Corky, and a generic four-wheeled car. The models were constructed using the Karma physics engine, which simulates rigid body dynamics in real time [82]. Not only does USARsim capture physical dynamics such as slip causing robots to veer as they attempt to drive straight, for example, but it also captures noise in sensor data. The uncertainty in sensor and motion models have recently been validated in USARsim for the arenas used in the RoboCup Rescue competition [82] [16]. Recent work has also been done to develop a communication model for USARsim that accurately propagates wireless Ethernet signals according to the structure of the environment. USARsim has been so successful that a new competition based on it (Virtual Robots competition) will be added to the RoboCup competition beginning in 2006. We propose to use USARsim to validate the coordination architecture described.

This validation will have the following components:

Agent architecture We will incorporate COCoA into the planning layer of the robot-agent architecture [58].

Evaluation We propose a two phase evaluation of our approach in USARsim. First, we will evaluate COCoA on benchmarks similar to those designed for COCoASim. We will select 5-10 different physical environments, create models of these environments that are consistent with the level of abstraction used in COCoA, and populate these models with a number of "goals", virtual victims. We will vary the number of known and unknown victims (we expect to be able to accommodate up to 15 known and unknown victims, depending on the size of the virtual environment), the number of robots (from two robots to teams of 15-20 robots), and the capabilities of the robots (from very capable to very limited in the sensors they carry though all robots will be chosen from the set of publicly available USARsim robots). Additionally, we will impose certain system constraints on some portion of these benchmarks. Uncertainty will come partly from USARsim since the amount of time to traverse the environment is no longer deterministic and partly from the accuracy of the abstract model provided to the robots. For example, we can simulate a landslide blocking a corridor by providing robots an initial map showing a free corridor when there is actually a wall there. Thus the environment dynamics we will consider include adding a robot, disabling a robot, adding a goal, removing a goal, disabling a capability on a robot, inaccuracies in the estimate of the amount of time required to accomplish a goal or travel from point to point, and changes in the environment. We will not consider making modifications to the environment (e.g. clear the "landslide") since this goes against the rules of the RoboCup Rescue competition and so is not easily simulated in USARsim. We will use the communication model integrated in USARsim to model fractured subteams and will look at the effect of the range of communication on performance. Since the simulator is now nondeterministic, multiple runs will be required for each algorithm or heuristic on each test case. We anticipate running thousands of simulations.

As a special case of these tests, we will evaluate our system in the rules of the RoboCup Rescue Competition where the robots will start with no information on the environment. We will look at the objective function used in that competition which does not discount rewards with time. While the previous phase of testing involves creating new benchmarks for others, this phase of testing will compare our approach to other approaches on existing benchmarks in a controlled manner.

Virtual Robots competition Although the Virtual Robots competition does not explicitly deal with the time critical tight coordination team planning problem addressed by this thesis proposal, we plan to enter the coordination architecture in the Virtual Robots competition to illustrate the flexibility of this approach and to increase awareness of the problem and solution techniques.

Illustrating COCoA in USARsim provides a number of the advantages of applying the architecture to real robots. USARsim has a lower cost than physical robots which is not only desirable for our own research but useful for the dissemination of the algorithms to the wider research community.

5 Schedule

This section provides a rough outline of proposed work for the completion of this thesis.

January-February 2006 Implement robust plan selection in COCoA. Design and run experiments.

February 21: AAAI paper deadline

March 2006 Design communication protocol for agents in COCoA; Incorporate communicator and constraint optimization algorithms into planning layer of the robot agent architecture; *Allocate some time for U.S. Open RoboCup preparation*

April 15-19: U.S. Open RoboCup competition

April-July 2006 Get familiar with USARsim; Port code from abstract simulator to USARsim

July 13-16: RoboCup 2006

August 2006 Incorporate lessons learned from Virtual Robots competition; Design benchmarks in USARsim; Publish code and benchmarks

September 2006 Improve performance through temporal constraint reasoning

October 2006 Improve plan merging algorithms

November 2006-May 2007 Run additional experiments; Write thesis

June 2007 Defend

6 Contributions

The proposed thesis would provide the following contributions to the robotics community:

Multirobot coordination through constraint optimization The mathematical programming model of the coordination problem described in this thesis proposal is a novel approach to multirobot coordination. We also describe an algorithm for combining traditional constraint optimization solution techniques with domain specific approaches such as market based coordination.

Time critical tight coordination team planning problem This thesis proposal formalizes and analyzes the time critical tight coordination team planning problem. We present a set of benchmarks that illustrate the complexities of the problem space. We empirically compare various algorithms on these benchmarks.

System constraints The first order logic constraint language described in this thesis proposal captures the important aspects of many multirobot coordination domains. We provide an implementation for this constraint language. The proposed thesis will also analyze the effect of various system constraints on team performance.

Communication failure In this thesis proposal we describe a new model for communication failure in spatially distributed robot teams. These communication failures result in *fractured subteams* and necessitate a distributed approach to coordination. We describe a hybrid system with distributed replanning with opportunistic centralization. We present an empirical analysis of communication failures on team performance.

Selective disruption minimization This thesis proposal describes selective disruption minimization as a method for improving team performance by attempting to avoid changes to schedules of certain robots on the team in order to reduce the cognitive load of humans interacting with those robots or to avoid inconsistent plans due to communication failures. We show that selective disruption minimization can be incorporated in the mathematical programming model of the team planning problem in a principled manner.

Robust plan selection The proposed thesis will describe a method for selecting plans that are robust to certain expected types of failure by, for example, allowing robots extra time to travel to goals or allocating surplus robots to a goal in case one robot fails. We will analyze the effect of robust plan selection on team performance both when the actual failures match the model of expected failures and when the expected failures do not occur.

Acknowledgements

This work is supported by National Science Foundation Award IIS-0205526. I would also like to thank the ILOG corporation for their generous support.

References

[1] Heimo H. Adelsberger and Wolfram Conen. Economic coordination mechanisms for holonic multi agent systems. In *DEXA Workshop*, pages 236–240, 2000.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[3] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, 2000.

[4] T. Balch. Taxonomies of multirobot task and reward, 1998.

[5] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. In *IEEE Transactions on Robotics and Automation*, 1999.

[6] Thomas Bartold and Edmund Durfee. Limiting disruption in multiagent replanning. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 49–56. ACM Press, 2003.

[7] Curt Bererton. *Multi-Robot Coordination and Competition Using Mixed Integer and Linear Programs.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2004.

[8] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 32–37, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[9] Silvia S. C. Botelho and Rachid Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[10] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, pages 478–485, 1999.

[11] M. E. Bratman. Shared cooperative activity. *The Philosophical Review*, 101(2):327–341, 1992.

[12] Michael Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, 1987.

[13] R. G. Brown and J.S. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *International Conference on Intelligent Robots and Systems (IROS 1995)*, August 1995.

[14] P. Caloud, Wonyun Choi, Jean Claude Latombe, C. Le Pape, and M. Yim. Indoor automation with many mobile robots. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, July 1990.

[15] E. Camponogara. Altruistic agents in dynamic games. Proc. 16th Brazilian Symposium on Artificial Intelligence, Lecture Notes in Artificial Intelligence, November 2002.

[16] Stephano Carpin, Jijun Wang, Michael Lewis, Andreas Birk, and Adam Jacoff. High fidelity tools for rescue robotics: results and perspectives.

[17] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1023–1028, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[18] L. Chaimowicz, T. Sugar, V. Kumar, and M. Campos. An architecture for tightly coupled multi-robot cooperation. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 2292–2297, 2001.

[19] P. R. Cohen and H. J. Levesque. Persistence, intention, and commitment. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 33–69. MIT Press, Cambridge, MA, 1990.

[20] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[21] Jeffrey S. Cox, Edmund H. Durfee, and Thomas Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 821–827, New York, NY, USA, 2005. ACM Press.

[22] M. Dias and A. Stentz. Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. CMU-RI-TR-03-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2003.

[23] M Bernardine Dias, Marc B Zinck, Robert Michael Zlot, and Anthony (Tony) Stentz. Robust multirobot coordination in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[24] M Bernardine Dias, Robert Michael Zlot, Nidhi Kalra, and Anthony (Tony) Stentz. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2005.

[25] M Bernardine Dias, Robert Michael Zlot, Marc B Zinck, Juan Pablo Gonzalez, and Anthony (Tony) Stentz. A versatile implementation of the traderbots approach for multirobot coordination. In *International Conference on Intelligent Autonomous Systems (IAS-8)*, March 2004.

[26] Frank Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, 1999.

[27] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3):309–314, 1997.

[28] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3:375–397, 1996.

[29] Edmund H. Durfee and Victor R. Lesser. Negotiating task decomposition and allocation using partial global planning. *Distributed Artificial Intelligence (Vol. 2)*, pages 229–243, 1989.

[30] T. Finin, R. Fritzson, and R. McEntire. KQML as an agent communication language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, November 1994.

[31] Foundation for intelligent physical agents, 1997.

[32] Brian Gerkey, Sebastian Thrun, and Geoff Gordon. Parallel stochastic hill-climbing with small teams. In *Multi-Robot Systems.* Kluwer, 2005.

[33] Brian P. Gerkey and Maja J. Mataric. Murdoch: Publish/subscribe task allocation for heterogeneous agents. In Carles Sierra, Maria Gini, and Jeffrey S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 203–204, Barcelona, Catalonia, Spain, 2000. ACM Press.

[34] Brian P. Gerkey and Maja J Matarić. Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, Taipei, Taiwan, May 2003.

[35] Brian P. Gerkey and Maja J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[36] GNU linear programming kit, http://www.gnu.org/software/glpk/glpk.html, accessed october 25, 2005.

[37] H.J. Greenberg. *Mathematical Programming Glossary.* World Wide Web, `http://www.cudenver.edu/~hgreenbe/glossary/`, 1996–2005.

[38] B. Grosz. Collaborative systems. AAAI-94 Presidential Address, reprinted, 1996.

[39] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming*, pages 222–236, 1997.

[40] ILOG. *ILOG CPLEX 9.0 User's Manual*, 2004.

[41] A. Jacoff, E. Messina, and J. Evans. Experiences in deploying test arenas for autonomous mobile robots. In *Proc. 2001 Performance Metrics for Intelligent Systems Workshop (PerMIS01)*, National Inst. of Standards and Technologies, 2002.

[42] N. Jennings. *Foundations of Distributed Artificial Intelligence*, chapter Coordination Techniques for Distributed Artificial Intelligence, pages 187 – 210. Wiley, 1996.

[43] N. R. Jennings and J. R. Campos. Towards a social level characterisation of socially responsible agents. *IEE Proceedings on Software Engineering*, 144(1):11–25, 1997.

[44] N. R. Jennings, E. H. Mamdani, I. Laresgoiti, J. Perez, and J. Corera. GRATE: A general framework for cooperative problem solving. *IEE-BCS Journal of Intelligent Systems Engineering*, 1(2):102–114, 1992.

[45] Nidhi Kalra, David Ferguson, and Anthony (Tony) Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the International Conference on Robotics and Automation*, April 2005.

[46] Mary Koes, Illah Nourbakhsh, and Katia Sycara. Communication efficiency in multi-agent systems. In *Proceedings of ICRA 2004*, May 2004.

[47] K. Lerman and A. Galstyan. A general methodology for mathematical analysis of multi-agent systems. Technical report ISI-TR-529, USC Information Sciences, 2003.

[48] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X.Q. Zhang. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

[49] C. Li and K. Sycara. Algorithm for combinatorial coalition formation and payoff division in an electronic marketplace, 2002.

[50] Cuihong Li, Shuchi Chawla, Uday Rajan, and Katia Sycara. Mechanisms for coalition formation and cost sharing in an electronic marketplace. Technical Report CMU-RI-TR-03-10, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2003.

[51] Qun Li and Daniela Rus. Communication in disconnected ad hoc networks using message relay. *J. Parallel Distrib. Comput.*, 63(1):75–86, 2003.

[52] lp_solve linear programming code developed by michel berkelaar, http://www.cs.sunysb.edu/ algorith/implement/lpsolve/implement.shtml, accessed november 3, 2005.

[53] Roger Mailler and Victor Lesser. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 438–445. IEEE Computer Society, 2004.

[54] M. Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *In Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, volume 2, pages 432–441, 1992.

[55] P. J. Modi and M. Veloso. Multiagent meeting scheduling with rescheduling. In *Distributed Constraint Reasoning, (DCR) 2004*, 2004.

[56] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proc. of the second international joint conference on Autonomous agents and multiagent systems*, pages 161–168, 2003.

[57] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artif. Intell.*, 161(1-2):149–180, 2005.

[58] Illah Nourbakhsh, Katia Sycara, Mary Koes, Mark Yong, Michael Lewis, and Steve Burion. Human-robot teaming for search and rescue. *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, pages 72–78, January 2005.

[59] Lynn Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14:504–516, April 1998.

[60] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie P. Kaelbling. Learning to cooperate via policy search. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–314, San Francisco, CA, 2000. Morgan Kaufmann.

[61] Adrian Petcu and Boi Faltings. An efficient constraint optimization method for large multiagent systems. In *AAMAS 05 - LSMAS workshop (Large Scale Multi-Agent Systems)*, Utrecht, the Netherlands, July 2005.

[62] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[63] V. Rodoplu and T.H. Meng. Minimum energy mobile wireless networks. *IEEE J. Select. Areas Commun.*, 17(8):1333 – 1344, August 1999.

[64] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[65] T. Sandholm. Making markets and democracy work: A story of incentives and computing, 2003.

[66] Paul Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proc. of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005.

[67] Jeff Schneider, David Apfelbaum, Drew Bagnell, and Reid Simmons. Learning Opportunity Costs in Multi-Robot Market Based Planners. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2005.

[68] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, May 1998.

[69] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), December 1980.

[70] Socially Situated Intelligence: a workshop held at SAB'98. *Social Intelligence as Norm Adaptation*, August 1998.

[71] D. G. Sullivan, A. Glass, B. J. Grosz, and S. Kraus. Intention reconciliation in the context of teamwork: An initial empirical investigation. *Lecture Notes in Computer Science*, 1652:149–??, 1999.

[72] K. Sycara, M. Paolucci, M. van Velsen, and J. Giampapa. The RETSINA MAS infrastructure. To appear in the special joint issue of Autonomous Agents and MAS, July 2003.

[73] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83 – 124, 1997.

[74] M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. Maheswaran, J. Marecki, P. J. Modi, R. Nair, S. Okamoto, J. P. Pearce, P. Paruchuri, D. Pynadath, P. Scerri, N. Schurr, and P. Varakantham. Conflicts in teamwork: hybrids to the rescue. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 3–10. ACM Press, 2005.

[75] F. Tang and L.E. Parker. Coalescent multi-robot teaming through asymtre: A formal analysis. In *Proceedings of the International Conference on Robotics and Automation*, April 2005.

[76] F. Tang and L.E. Parker. Distributed multi-robot coalitions through asymtre-d. In *Proc. of the Workshop on Networked Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005.

[77] A. Tews and G. Wyeth. Multi-robot coordination in the robot soccer environment, 1999.

[78] Unreal engine, http://www.epicgames.com, accessed november 1, 2005.

[79] Douglas Vail and Manuela Veloso. Dynamic multi-robot coordination. In *Multi-Robot Systems*. Kluwer, 2003.

[80] Douglas Vail and Maria Manuela Veloso. Multi-robot dynamic role assignment and coordination through shared potential fields. In A. Schultz, L. Parker, and F. Schneider, editors, *Multi-Robot Systems*. Kluwer, 2003.

[81] J. Wang, M. Lewis, and J. Gennari. USAR: A game based simulation for teleoperation. In *Proc. 47th Ann. Meeting Human Factors and Ergonomics Soc.*, July 2003.

[82] Jijun Wang, Michael Lewis, Stephen Hughes, Mary Koes, and Stephano Carpin. Validating usarsim for use in hri research. In *Proceedings of the 49th Annual Meeting of the Human Factors and Ergonomics Society*, September 2005.

[83] Barry Brian Werger and Maja J. Mataric. Broadcast of local eligibility: behavior-based control for strongly cooperative robot teams. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*, pages 21–22, New York, NY, USA, 2000. ACM Press.

[84] H. P. Williams. *Model Building in Mathematical Programming.* Wiley, fourth edition edition, 2003.

[85] Yang Xu, Paul Scerri, Bin Yu, Steven Okamoto, Michael Lewis, and Katia Sycara. An integrated token-based algorithm for scalable coordination. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 407–414, New York, NY, USA, 2005. ACM Press.

[86] Junichi Yamamoto and Katia Sycara. A stable and efficient buyer coalition formation scheme for e-marketplaces. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 576–583, New York, NY, USA, 2001. ACM Press.

[87] Holly A. Yanco and Jill L. Drury. Classifying human-robot interaction: an updated taxonomy. In *SMC (3)*, pages 2841–2846, 2004.

[88] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.

[89] R. Zlot and A. Stentz. Multirobot control using task abstraction in a market framework. In *Collaborative Technology Alliances Conference*, 2003.