Heterogeneous Multirobot Coordination with Spatial and Temporal Constraints

Mary Koes, Illah Nourbakhsh, and Katia Sycara
{mberna, illah, katia}@cs.cmu.edu
Robotics Institute
Carnegie Mellon University

Abstract

Although recent work in multirobot teaming targets challenging domains such as disaster response or planetary exploration, the lack of formal problem descriptions and benchmarks make it difficult to evaluate various coordination approaches. We formally describe the problem of scheduling a team of robots with heterogeneous capabilities to accomplish a set of joint tasks that are spatially distributed. Finding the optimal plan in these domains requires robots to integrate path planning and task allocation with scheduling, which existing approaches to multirobot coordination separate. We have developed a declarative framework for modeling the problem as a mixed integer linear programming (MILP) problem and a centralized anytime algorithm with error bounds. The anytime algorithm can outperform standard MILP solving techniques, greedy heuristics, and a market based approach which separates scheduling and task allocation. Generating improved plans is insufficient if the schedules cannot be repaired to accomodate new observations or hardware failures as the robots traverse the environment. We present a framework for schedule repair that leverages the expressive nature of the MILP representation and minimizes the plan disruption.

Introduction

Challenging environments such as disaster response, reconnaissance, and space construction require the coordination of teams of robots. Robot teams in these domains will necessarily be heterogeneous as cost limitations, power consumption and size constraints require tradeoffs between mobility and capabilities. Task execution in these domains may require subteams of robots to work together on a joint task if no single robot possesses the necessary capabilities. Even when one robot does possess sufficient capabilities, in time sensitive domains such as search and rescue, the formation and cooperation of subteams of robots may improve team efficiency saving time, energy, or even lives. Coordination of multiple robots on a joint task requires robots to consider scheduling concurrently with task allocation. The presence of spatial constraints requires robots to consider travel costs and path planning during scheduling. Therefore, coordinating a team of robots to accomplish a set of joint tasks at different locations in the environment requires simultaneously solving the task assignment, scheduling, and path planning problems.

We are interested in the problem where robots start with some initial information about the environment and goals. For example, in the search and rescue domain, robots may have an initial blueprint with probable victim locations. We assume that goals have time-varying rewards. Robots may be heterogeneous both in their ability to traverse the environment and in their sensors and manipulators. Tasks may require several capabilities. An example from the search and rescue domain is the task of evaluating the state of a victim which may require a microphone, vision system, and spotlight. If no single robot is able to provide all the required capabilities, multiple robots need to cooperate and all work on the goal at the same time.

The problem can be viewed as a constraint optimization problem with the objective of maximizing rewards from various goals. If we assume that rewards vary linearly with time, the problem can be formulated as a mixed integer linear programming (MILP) problem incorporating scheduling, task allocation, and path planning constraints. An MILP problem formulation allows the flexibility to combine multiple objective functions and to put ordering constraints on the tasks. The declarative nature of this representation allows intelligent replanning and reasoning about commitments to teammates and the effects of a change in plans. It also facilitates communication between robot agents and humans who may monitor execution and wish to change the plans.

Incorporating spatial constraints constitutes a significant challenge, however. A naive MILP problem formulation is to 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 representation is computationally infeasible. Solving MILPs is NP-hard, and linear programs are inefficient at solving systems with long sequential plans as in path planning (Bererton 2004). Likewise, discretizing time causes problem complexity to increase inversely with the time step size and directly with the amount of time robots consider during scheduling. Discrete time models may also be suboptimal if the time step size is not sufficiently small. Instead we have developed an MILP formulation that is independent of the environment size and uses a continuous time model.

Inspired by human emergency responder coordination, we

assume that the environment can be broken down into independent areas to be searched by manageably sized teams of robots so that each team has on the order of 10 robots. We present an anytime algorithm that converges to the optimal solution while providing error bounds by combining standard ILP solution techniques with domain specific heuristics. Although a centralized approach would not scale to systems with hundreds of robots, our approach can outperform distributed approaches since it leverages the relaxed solution to the MILP problem. It has been able to find schedules for up to 20 robots within seconds, a necessary quality for a system to be implemented on real robots.

Due to the uncertain nature of these environments the initial information is likely to be incorrect and will need to be refined online as the robots execute their plans. Robots may become disabled or lose capabilities while new teammates or sensors may be added to the team. Although it is always possible to naively replan after these events, this strategy is not condusive to team stability. Since robots may occasionally lose contact with the centralize planner, sudden and frequent disruptions should be avoided. For systems where humans interact with the robots such as in the search and rescue domain, human factors research (Adamczyk & Bailey 2004) suggests that interruptions reduce user effectiveness and should be carefully managed.

The rest of the paper is organized as follows. We discuss existing techniques for multirobot coordination and explain why they fail to solve the class of problems involving spatial and temporal constraints. Next we formally describe the problem and instantiate it in the search and rescue domain. We show that the problem can be formulated as a MILP problem and present our anytime algorithm that provides error bounds and converges to the optimal solution. We introduce a framework for replanning while considering disruption. We present the results of some experiments and conclude with future work.

Related Work

Jones *et al.* (2004) propose a taxonomy of multirobot task allocation problems depending 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). While this taxonomy ignores heterogeneity, it is useful for comparing our work, which falls into the ST-MR-TA category, with existing systems. Market-based planners such as MURDOCH (Gerkey & Matarić 2003) and TraderBots (Dias *et al.* 2004) do not permit subteams to work on a task and so fall into the ST-SR-IA and ST-SR-TA categories respectively. Neither of these systems provide error bounds. When a single robot is allocated to each task, interleaving task allocation and scheduling is sufficient and the problems do not need to be considered simultaneously. While recent work by (Schneider *et al.* 2005) extends the TraderBots system to include time varying rewards and heterogeneous robots, it still lacks the ability to solve the problem we propose requiring multiple robots to coordinate on a task.

The field of coalition formation in the multiagent systems domain is related to the ST/MT-MR-IA class of problems. Shehory and Kraus (1998) present distributed anytime algorithms with error bounds in the multiagent domain but do not address reasoning about spatial constraints or time varying reward. However, 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 (Li *et al.* 2003). This is a different paradigm than the declarative MILP framework in our work which has no model of individual robot utility but explicitly maximizes team utility.

A final body of related work includes distributed constraint optimization (DCOP) such as ADOPT (Modi *et al.* 2003). However, distributed algorithms cannot leverage the relaxed solution as a centralized solution. Even for the much simpler DiMES problem presented in (Maheswaran *et al.* 2004) which does not include spatial constraints, the ADOPT algorithm required minutes or hours so solve problems that a centralized planner could solve in a fraction of a second and so is ill-suited for robotics applications.

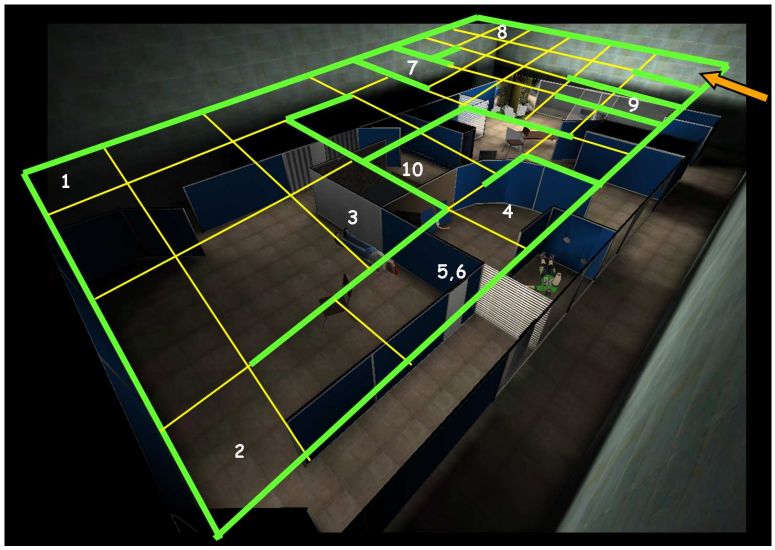
LA-DCOP (Scerri *et al.* 2005) is a task allocation algorithm for large scale teams that attempts to maximize the team's expected utility given probabilistic information by computing a minimum capability threshold for each task. LA-DCOP then interleaves scheduling and task allocation. It does not guarantee the optimal solution or provide error bounds. LA-DCOP 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.

Heterogeneous Task Scheduling with Spatiotemporal Constraints

Given a map of the environment with tasks to be accomplished and a set of robots with heterogeneous capabilities, we need an approach to obtain the optimal solution. Figure 1 shows a sample environment and an abstraction for the environment that the robots can use for planning. The desired output, a *team schedule* in absolute time that can be executed by the robots, is shown in figure 2.

We present a general problem formulation that captures the important characteristics of problems with heterogeneous agents and joint activities for domains that require spatial as well as temporal coordination. Suppose we have a set of N resources operating in an environment with K relevant capabilities and a set of M tasks to be accomplished. Each task has an associated reward, location, and duration. We then begin with a resource set $\mathcal{R} := \{R^1, R^2, ..., R^N\}$ and a set of tasks or goals $\mathcal{G} := \{G^1, G^2, ..., G^M\}$. The object is to create a team plan for all the resources to maximize the team reward over the time available.

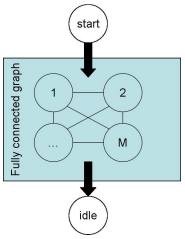
We assume that the environment is discretized and represented as a weighted undirected graph of nodes and edges, $(\mathcal{N}, \mathcal{E})$. Each resource R^n has specific capabilities, S^n



(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



(c) Goal map

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. We use an intermediate goal map representation (c) for the MILP problem formulation.

where S^n is a vector whose length is equal to the cardinality, K, of the set of capabilities relevant to this environment.

Each task has an associated location, duration, reward, and necessary set of capabilities. 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 resource set, $\mathcal{R}$, 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.

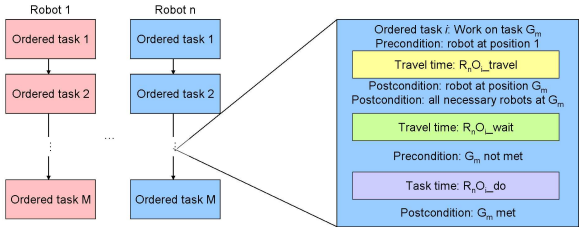


Figure 2: The output of our algorithm is a *team schedule*, a set of goal oriented schedule each of which can be executed by an average robot with a 3T architecture.

Search and Rescue

The dangerous nature of a disaster area makes it an ideal application for robotic exploration. Yet the unstructured and uncertain nature of the environment, the necessity for speed and heterogeneous capabilities, and the communication constraints all make it a challenging application for multirobot coordination. Disaster response encompasses a large area of problems. This work focuses on the problem of exploring a semistructured disaster site such as the RoboCup Rescue competition (Jacoff, Messina, & Evans 2002) and the accompanying simulator (Wang, Lewis, & Gennari 2003) (Figure 1a). The team receives points for mapping the environment, locating victims, determining whether the victim is "alive" (by analyzing heat, motion, or sound signatures), and reading small identification tags the victim may be wearing. We consider three relevant capabilities based on real robot capabilities: mapping (requires a laser range finder), victim state identification (requires heat sensor), and victim ID tag recognition (requires light, camera system, and good mobility). Since the simulator interacts with the environment at a very low level, the robot architecture must provide a suitable layer of abstraction for planning (Nourbakhsh *et al.* 2005).

MILP Problem Formulation

Since MILP solvers are typically not adept at solving long sequential plans and can be much slower at path planning than A* or other search methods, we attempt to reduce path planning complexity by transforming the original environment $(\mathcal{N}, \mathcal{E})$ into a goal map $(\mathcal{N}_G, \mathcal{E}_G)$ where each goal $G^m \in \mathcal{G}$ is mapped to a node in $\mathcal{N}_G$ (Figure 1c). The transformation from the original environment into the goal map requires that path planning be precomputed. We use

Floyd's algorithm which is $O(n^3)$ (Cormen, Leiserson, & Rivest 1990). In general, the edge costs may differ for different classes of robots. In addition to the task nodes, we also have a start node and an idle node. The start node has directed edges to each task node in the graph as well as the idle node and represents the starting positions of the robots. Every node has an edge with zero cost to the idle node. The idle node has no transitions out since a robot would never schedule a timeslot as idle unless there were no more possible tasks for it to work on.

The objective is to find a schedule for each robot that maximizes team utility. Each schedule consists of a set of subplans representing a path through the goal map that can be mapped back onto a path through the original environment. The number of subplans scheduled for any given robot is defined as the planning horizon (ω) for that robot. Each subplan consists of the time to travel to the location to perform the task, the time spent waiting for teammates to arrive, and the time actually performing the task (Figure 2).

We build a 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:

- 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 (figure 2) 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 must work on exactly one task at a time:

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

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

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

- Network constraints: 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:

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

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

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

  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).

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

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

$$\forall_{n \text{ s.t. } R^n \in \mathcal{R}} : R_nO_1\_startDo = R_nO_1\_travel + \delta$$

- General time constraint based on timestep definition (Figure 2)

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

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

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

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

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

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

$$\forall_{m \text{ s.t. } G^m \in \mathcal{G}, n \text{ s.t. } R^n \in \mathcal{R}} : G_m\_start - R_nG_m\_startDo + d^m \geq R_nG_m\_do$$
$$R_nG_m\_do \leq R_nG_m \cdot T_{max}$$
$$R_nG_m\_do \geq d^m R_nG_m$$
$$\forall_{m \text{ s.t. } G^m \in \mathcal{G}} : \sum_{n \text{ s.t. } R^n \in \mathcal{R}} R_nG_m\_do \geq d^m \sum_n R_nG_m$$

- A robot may not work on a joint task before the scheduled start time: (necessary to force all robots to work on the task concurrently):

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

- All times must be less than T_{max}.
- 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. For example, we have linking constraints relating $R_n O_i\_startDo$ to $R_n G_m O_i\_startDo$ and $R_n G_m O_i\_startDo$ to $R_n G_m\_startDo$ so that the constraints on the robot's time are transferred to the tasks allocated to that robot. Linking constraints are necessary in order for the resulting solution to be legal. These constraints are not included here but can be viewed at *http://www.cs.cmu.edu/~mberna/research/aaai_sup.pdf*.

Anytime Scheduling Algorithm

Algorithm 1 Anytime Scheduling using MILP Solver

1: $planHoriz$ = # Tasks
2: $prob$ = CreateMILP($planHoriz$)
3: $upperBound$ = RelaxedSolution($prob$)
4: $oldSoln = \{\}$
5: **for** $planHoriz = 1$ to # Tasks **do**
6: $prob$ = CreateMILP($planHoriz$)
7: $start$ = HeuristicScheduler($oldSoln$, $planHoriz$)
8: $optimalSolnForPH$ = Optimize($prob$, $start$)
9: $oldSoln = optimalSolnForPH$
10: **end for**
11: return $optimalSolnForPH$

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 (Cormen, Leiserson, & Rivest 1990). 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 NP hard (Cormen, Leiserson, & Rivest 1990) 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 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.

Framework for Schedule Repair

As robots execute their plans, they will discover discrepancies between the real world and their representation. We have defined a set of replanning catalysts:

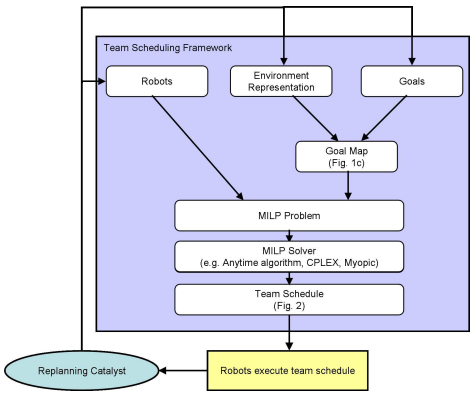
– Robot added
– Robot removed/incapacitated
– Robot capability added
– Robot capability removed/incapacitated
– Goal added
– Goal removed
– Goal capability requirement added
– Goal capability requirement removed
– Path cost shortened
– Path cost lengthened
– Goal duration increased
– Goal duration decreased
– Goal reward changed

The discovery of any of these events necessitates replanning. We have identified three methods for replanning.

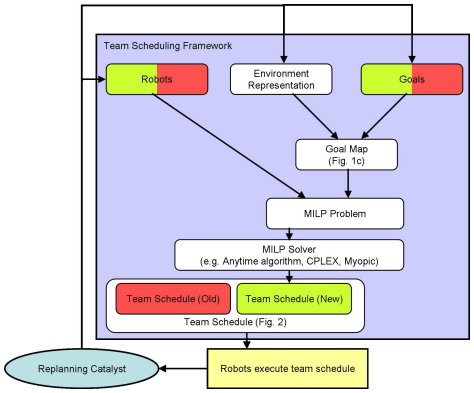
Full Replanning Given the current state at the time of the interruption and the changes to the beliefs caused by the interruption, a new MILP can be formulated and solved using the anytime algorithm previously described. This approach (illustrated in figure 3a) will converge to the optimal solution but ignores the cost of processing an interruption. If a robot is out of communication and does not receive the new plan, team performance may suffer. Humans monitoring and interacting with the robots may have difficulty context switching from one task to another. Human factors research suggests that interruptions should be kept to a minimum and carefully handled in order to reduce cognitive load and receive optimal performance (Adamczyk & Bailey 2004).

Constrained Replanning One option to guard against disruption is to artifically constrain the problem (Figure 3b). For example, robots out of communication range may be constrained to keep the original plan and ignored for replanning purposes or robots interacting with humans may not change the next two planned tasks. This reduces the size of the MILP and provides for certain guarantees on system stability but may result in suboptimal behavior.

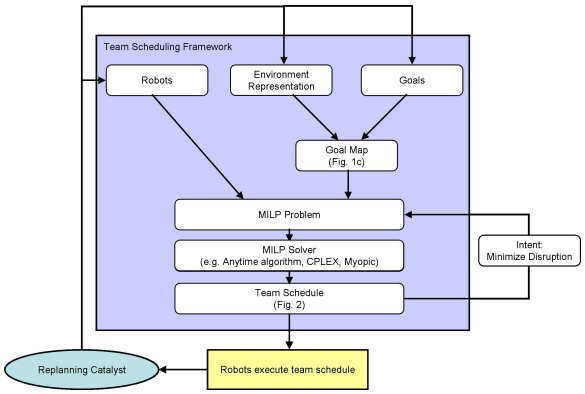
Aware Replanning Rather than using hard constraints as in constrained replanning, it is possible to include pref-



(a) Full replanning



(b) Constrained replanning



(c) Aware replanning

Figure 3: The schedule repair framework permits three types of replanning. Full replanning (a) converges to the optimal solution within the restrictions of the solver. Constrained replanning (b) holds some of the original plan fixed (red) and solves the remaining reduced problem (green). Aware replanning incorporates the desire to reduce disruption into the objective function during replanning.

erences to avoid changes to the current plan in the objective function. Rather than simply maximizing the utility as described in the MILP Problem Formulation, the system seeks to maximize the utility minus penalties incurred for changing variable values (Figure 3c). The resulting schedule is then optimal to the extent that these penalties reflect the true costs of changing the plan. Hybrid approaches combining constrained and aware replanning are also possible.

In order to avoid miscommunications and reduce the cognitive load of humans observing and interacting with the robots, we want to minimize the amount that the plan changes, particularly in the near term. The quantification of the difference between two plans is defined as disruption. We investigated several possible metrics for measuring disruption including a dynamic programming metric based on biological sequence analysis (Durbin *et al.* 1998). However, in order to include disruption in the objective function for aware replanning, it must be described as a linear equation.

We use the following metric for disruption:

$$\sum_{m \text{ s.t. } G^m \in \mathcal{G}, n \text{ s.t. } R^n \in \mathcal{R}} \frac{|R_n G_m\_start^{*'} - R_n G_m\_start^{*}|}{T_{max}} \gamma_n$$

where $R_n G_m\_start^{*'}$ is the value of $R_n G_m\_start^{*}$ in the original schedule and $R_n G_m\_start^{*}$ corresponds to the the time at which robot n is scheduled to begin task m or T_{max} if robot n is not scheduled to work on task m as explained in the MILP Problem Formulation. Each robot has a cost associated with changes in the plan, γ_n. Our metric for disruption is the weighted change in time at which the task will start. Taking on a task or dropping a tasks has a large cost while small changes are relatively inexpensive. Other possible metrics include penalizing changes in earlier tasks more strongly than changes in later tasks, penalties based only on the number of robots with changed schedules, or time independent penalties based on changes in the set of goals. The metric chosen should be appropriate to the domain and the cost of context switching within that domain.

Results

We have tested our formulation and algorithm extensively in an abstract simulator. The simulator randomly generates an initial blueprint, represented as a grid world. We tested environments between 10 and 1000 nodes and found (as expected) that planning complexity is independent of environment size in our formulation. The simulator also randomly generates a set of tasks with rewards and capability requirements. Robots are randomly given capabilities. The value of T_{max} was always sufficiently high to permit the team to accomplish all the tasks (subject to constraints on team capabilities).

We considered four different algorithms. A myopic scheduling algorithm recursively optimizes the MILP for a planning horizon of 1, that is scheduling 1 task for each robot at each iteration. The standard ILP solver (CPLEX) optimizes the MILP for a fixed planning horizon. The anytime algorithm combines these approaches as described in Algorithm 1. We also compared our results to a greedy

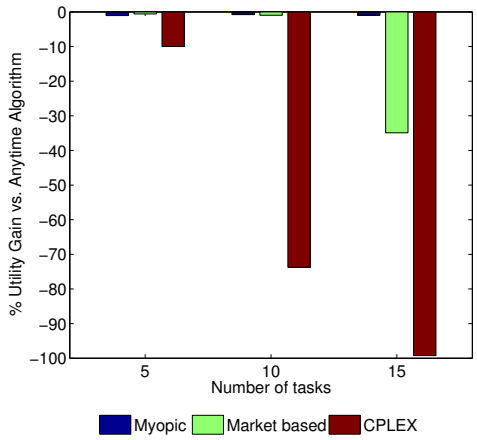


Figure 4: The anytime algorithm consistently resulted in higher utility than the other approaches. Results shown are averaged over trials with 3, 9, and 15 robots.

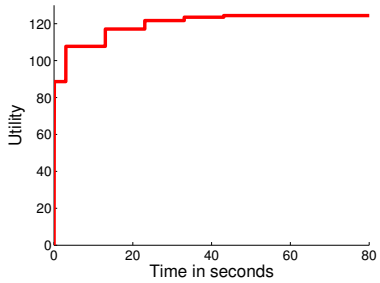


Figure 5: Anytime Algorithm results over time on example from figure 1(a-b). All times are CPLEX CPU times on a 2.8GHz Pentium 4 machine with 1GB RAM.

market based algorithm that sequentially holds one auction for each capability requirement of each task to allocate the tasks and then optimizes the schedule based on that allocation. Though other improvements are possible, including reauctioning tasks, several market-based systems (e.g. MURDOCH (Gerkey & Matarić 2003)) use this greedy allocation algorithm.

Figure 4 shows the results of the four algorithms on randomly generated test cases, varying the number of robots (3, 9, and 15) and the number of tasks (5, 10, and 15). The ILP solver, CPLEX, performed poorly due to the time limit of 60 seconds we imposed during which time it was unable to find an integer solution. Note that the anytime algorithm always performed as well as or better than the myopic scheduler with a mean utility gain of 0.6% and a maximum utility gain of approximately 13% even under tight time constraints and improves further over time. The anytime algorithm performed significantly better than the market based algorithm with separated task allocation and scheduling with an average utility gain of 10%. The performance of both the market based system and CPLEX decreased as problem complexity (number of tasks) increased. However, while CPLEX typically performed slightly better with fewer robots, the market based system performed significantly worse with 3 robots because the robot interactions and increased number of tasks per robot resulted in schedule inefficiencies.

We also examined the performance of the various algorithms over time on the example problem from Figure 1(a-b). Figure 5 shows that the anytime algorithm has discrete jumps corresponding to the heuristic scheduler step of Algorithm 1 (line 7) and slight improvements as the ILP searches the space near the heuristic solution, reaching the maximum utility of 124.5 after about 45 seconds. The myopic heuristic terminated with a maximum utility of 124.35 in less than 0.1 seconds. The market based approach required 56 seconds to find a feasible schedule and 210 seconds to find the optimal schedule (given the initial allocation) with a utility of 123.3. 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 show that the anytime algorithm significantly improves performance over standard ILP solution techniques, though if time is an issue, the myopic heuristic may be more suitable.

Conclusion and Future Work

The problem of task allocation for heterogeneous robot teams with support for joint tasks has different challenges than either task allocation without joint tasks or multiagent coalition formation without spatial constraints. Finding the optimal solution requires simultaneously solving the path planning, scheduling, and task allocation problems. A declarative framework enables robots to reason about their commitments to their teammates, explain their actions to humans, and provides sufficient flexibility. We presented a continuous time MILP problem formulation that minimizes the effect of path planning complexity. We also demonstrated an anytime algorithm with error bounds capable of both quickly finding a solution for midsized robot teams and of finding the optimal schedule, given enough time.

Due to inaccuracies in the representation and uncertainty in the environment, robots will need to refine their schedules as they execute their plans. We utilize the declarative framework provided by the MILP problem formulation for intelligent replanning, reasoning about interactions and schedule stability during replanning.

The algorithm and framework provided suffer from the standard drawbacks of a centralized system including vulnerability to point failure, communication failure, and lack of scalability. We are investigating methods for mitigating these drawbacks by designing and implementing a distributed version of this algorithm. One approach would be to have each robot contain a copy of the environment and perform the planning. This naturally introduces challenges in keeping the plans synchronized across robots, particularly during replanning.

Another area for future work includes augmenting the problem definition to include ordering constraints between tasks, partial rewards for partial satisfaction of goal requirements, and real valued capabilities. The MILP problem formulation is general enough to easily accommodate these

changes though it is unclear what the impact on solution complexity would be.

While work thus far has dealt primarily with creating plans and reacting to disturbances, system performance may be increased by reasoning about uncertainty. For example, if there exists a high prior probability that a robot will be incapacitated, it may be preferable to use another robot, or send two robots. Learning the distributions for interruptions online and planning accordingly is another interesting topic for future research.

Acknowledgements

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

References

Adamczyk, P., and Bailey, B. 2004. If not now, when?: The effects of interruption at different moments within task execution. In *Human Factors in Computing Systems: Proceedings of CHI'04*, 271–278. ACM Press.

Bererton, C. 2004. *Multi-Robot Coordination and Competition Using Mixed Integer and Linear Programs*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. MIT Press/McGraw-Hill.

Dias, M. B.; Zlot, R. M.; Zinck, M. B.; Gonzalez, J. P.; and Stentz, A. T. 2004. A versatile implementation of the traderbots approach for multirobot coordination. In *International Conference on Intelligent Autonomous Systems (IAS-8)*.

Durbin, R.; Eddy, S.; Krogh, A.; and Mitchison, G. 1998. *Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Gerkey, B. P., and Matarić, M. J. 2003. Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*.

Jacoff, A.; Messina, E.; and Evans, J. 2002. Experiences in deploying test arenas for autonomous mobile robots. In *Proc. 2001 Performance Metrics for Intelligent Systems Workshop (PerMIS01)*.

Jones, C.; Shell, D.; Matarić, M. J.; and Gerkey, B. 2004. Principled approaches to the design of multi-robot systems. In *Proc. of the Workshop on Networked Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*.

Li, C.; Chawla, S.; Rajan, U.; and Sycara, K. 2003. 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.

Maheswaran, R. T.; Tambe, M.; Bowring, E.; Pearce, J. P.; and Varakantham, P. 2004. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 310–317.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *Proc. of the second international joint conference on Autonomous agents and multiagent systems*, 161–168.

Nourbakhsh, I.; Sycara, K.; Koes, M.; Yong, M.; Lewis, M.; and Burion, S. 2005. Human-robot teaming for search and rescue. *IEEE Pervasive Computing: Mobile and Ubiquitous Systems* 72–78.

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

Schneider, J.; Apfelbaum, D.; Bagnell, D.; and Simmons, R. 2005. Learning Opportunity Costs in Multi-Robot Market Based Planners. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

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

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