Distributed Coordination of Mobile Agent Teams: The Advantage of Planning Ahead

Laura Barbulescu, Zachary B. Rubinstein, Stephen F. Smith, and Terry L. Zimmerman
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15024
{laurabar,zbr,sfs,wizim}@cs.cmu.edu

ABSTRACT

We consider the problem of coordinating a team of agents engaged in executing a set of inter-dependent, geographically dispersed tasks in an oversubscribed and uncertain environment. In such domains, where there are sequence-dependent setup activities (e.g., travel), we argue that there is inherent leverage to having agents maintain advance schedules. In the distributed problem solving setting we consider, each agent begins with a task itinerary, and, as execution unfolds and dynamics ensue (e.g., tasks fail, new tasks are discovered, etc.), agents must coordinate to extend and revise their plans accordingly. The team objective is to maximize the utility accrued from executed actions over a given time horizon. Our approach to solving this problem is based on distributed management of agent schedules. We describe an agent architecture that uses the synergy between intra-agent scheduling and inter-agent coordination to promote task allocation decisions that minimize travel time and maximize time available for utility-acrruing activities. Experimental results are presented that compare our agent's performance to that of an agent using an intelligent dispatching strategy previously shown to outperform our approach on synthetic, stateless, utility maximization problems. Across a range of problems involving a mix of situated and non-situated tasks our advance scheduling approach dominates this same dispatch strategy. Finally, we report performance results with an extension of the system on a limited set of field test experiments.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence - *Distributed Artificial Intelligence*

General Terms

Algorithms, Design

Keywords

Multi-agent scheduling, agent architectures

Cite as: Distributed Coordination of Mobile Agent Teams: The Advantage of Planning Ahead, Laura Barbulescu, Zachary B. Rubinstein, Stephen F. Smith, and Terry L. Zimmerman, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

1. INTRODUCTION

In many practical domains, a team of agents is faced with the problem of carrying out geographically dispersed tasks under evolving execution circumstances in a manner that maximizes global payoff. Consider the situation following a natural disaster such as an earthquake or hurricane in a remote region. A recovery team is charged with surveying various sites, rescuing injured, determining what damage there is to the infrastructure (power, gas, water) and effecting repairs. The tasks that must be performed entail various dependencies; some rescues cannot be safely completed until infrastructure is stabilized, certain repairs require parts that must be brought on site or prerequisite repairs to other services, and some services, such as clinics for injured, are not operational until utilities at the clinic sites are restored. There will likely be more for the team to do than can be achieved in the early stages of the response, necessitating judicious ordering of tasks for each agent and consideration of deadlines imposed by medical emergencies. Since sites are distributed geographically, task allocation must take into account locality to minimize the time agents spend traveling and maximize the time they spend performing disaster relief tasks. Complicating travel between sites is the possibility of discovering either roads that are blocked and require specialized resources to clear them, or impassable roads that cannot be repaired within the mission time. Furthermore, communications may have latency and periods of blackouts.

It is difficult to coordinate centrally in such environments, given the spatially distributed nature of unfolding events, communication limitations, and the need for each agent to act quickly as conditions evolve. While typically each team member has a globally constructed initial plan going into the mission, they quickly have to coordinate among themselves to accommodate discoveries that require additional or revised tasks.

In this paper, we consider this distributed coordination problem. Our general claim is that in domains where agents are both mobile and spatially distributed, and where sequence-dependent setup activities such as agent travel are required to perform target tasks, there is inherent leverage in maintaining advance schedules and using them to drive coordination. Projection of future actions provides an explicit basis for reasoning about the costs and benefits of different task allocation and task ordering decisions that might be taken by different agents, and for maximizing the time spent by the team of agents on utility accruing tasks. We focus specifically on the Phase 3 version of this problem defined within the DARPA Coordinators program, where each agent is given an initial schedule of tasks to perform, together with knowledge of the inter-dependencies with other agents' schedules and a specification of local (fall-back) task substitution options. The overall set of target tasks is a mixture of located and non-located tasks. The objective is to maximize the cumulative utility accrued from all executed tasks as execution unfolds in an environment where some tasks take more or less time than expected, others fail, and additional tasks are introduced.

Our advance distributed scheduling approach builds on the framework initially described in [10]. An incremental, Simple Temporal Network (STN)-based agent scheduler designed to favor high utility tasks is augmented with both an intra-agent mechanism that heuristically selects slots on the agent's timeline to minimize introduced travel and an inter-agent mechanism that arbitrates which agent executes a new task based on minimizing the team's tour. An additional auction-based coordination mechanism is incorporated for synchronizing inter-dependent tasks, also in a manner that minimizes travel.

Prior research in distributed coordination of mobile agents via advance planning has focused primarily on domains where goal tasks either are largely independent of one another and decompose nicely into local traveling salesman problems (e.g., [7]), or possess complex interdependencies but are not temporally constrained (e.g., [12, 11]). Other work in distributed scheduling approaches to coordination in dynamic, uncertain domains (e.g., [5]), including prior work within the Coordinators program [10], has focused on synthetic, stateless domains in which tasks can be executed without delay by qualified agents any time after specified enabling tasks have been executed. In fact, in this problem setting it was recently shown that a dynamic dispatching strategy, based on distributed estimation of the relative criticality of various pending tasks and use of simple policies like keeping an agent busy whenever possible, significantly outperformed an advance scheduling approach. [8]. However, as we will show in this paper, these comparative results between dynamic dispatch and advance scheduling are reversed when tasks are spatially dispersed in the world and the cost of travel must be considered.

The remainder of the paper is organized as follows. We first specify the distributed coordination setting that we consider more precisely. We then summarize the agent architecture and principal underlying scheduling and coordination mechanisms that we have developed to solve this class of problem. We next present the results of an extensive comparative performance evaluation of our approach and the above mentioned dispatch strategy across a range of problem scenarios requiring the execution of tasks at different specified locations. We then briefly describe extensions to allow application of our agent to a disaster response field test exercise, and report results obtained in this experiment. We conclude with a discussion of current research directions.

2. PROBLEM SETTING

As indicated above, we focus on solving the distributed coordination problem defined for Phase 3 of the DARPA Coordinators program. In this setting, it is assumed that no agent has a global view of the overall problem and solution. Instead each agent is given a local *subjective view*, which encapsulates the portion of the overall team plan that it is responsible for executing (including those tasks that are not in the agent's initial schedule but could potentially be executed by the agent) and the set of remote tasks (i.e., owned by other agents) that have inter-dependent relationships with local tasks. Any task can be constrained to occur at a specific location; this introduces the need to reason about agent travel. Agents accrue quality when a task is successfully executed, and the objective is to maximize the cumulative quality accumulated by all agents over the course of a fixed mission duration. An agent can execute only one activity at a time and can communicate at any time with any other agent. A multi-agent simulation system (called MASS) is used to provide the uncertain execution environment and an infrastructure for evaluation.[1]

Problem scenarios and initial agent plans are specified using an extended version of *C_TAEMS* [1], which is itself a variation of the original TAEMS (Task Analysis, Environment Modeling and Simulation) framework [4]. The *C_TAEMS* language is essentially a fully expanded, Hierarchical Task Network (HTN) representation of the overall team mission. Primitive tasks (also refered to as methods or activities) are tasks that are executable by a specific designated agent (owner), and include specification of probability distributions for duration, quality, and failure likelihood. Higher-level tasks aggregate more detailed processes and alternatives, and a rich set of quality accumulation functions (QAFs) is provided for specifying how successfully completed subtasks combine to contribute to the quality of parent tasks. Release and deadline constraints can be associated with any task in the hierarchy and inter-dependencies between tasks are specified as non-local effects (NLEs). NLEs can include hard constraints such as enabling and disabling precedence constraints between activities in the HTN and soft constraints that specify a facilitating or hindering relationship between two activities when they execute in a specific order. A more complete summary of these language features is presented in [10].

For Phase 3 problem definition, the *C_TAEMS* language was extended to provide constructs for specifying located activities and movement of agents. Locations are defined by associating a label with a unique pair of latitude and longitude values. An activity with a designated location carries the semantics that the executing agent must be located at this location for the activity's entire duration for the specified quality to be accrued. The route network is specified through special movement template forms that provide a basis for instantiating agent movement activities between two locations. Each movement template specifies the duration for a movement between the two designated locations. Movement activities accrue no quality when executed; they are inserted into an agent's schedule to relocate the agent from one declared domain location to another.

3. BASIC AGENT ARCHITECTURE

Our approach to maintaining advance schedules is based on the agent architecture originally described in [10], which is depicted schematically in Figure 1. In its most basic form, an agent comprises four principal components - an Executor, a Scheduler, a Distributed State Manager (DSM), and an Options Manager - all of which share a common model of the current problem task structure and solution state. This common model couples a domain-level representation of the agent's local (subjective) view of the overall team mission

[1]Coordinators Phase 3 also included an extended field test exercise, based on a disaster rescue scenario, that will be discussed in Section 6.

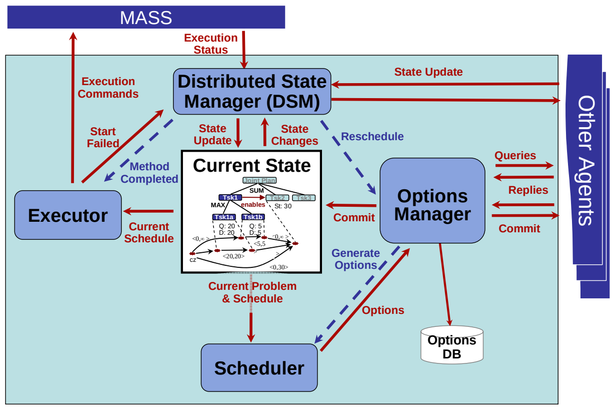


Figure 1: Agent Architecture.

(encoded as a CTAEMS task structure) to an underlying Simple Temporal Network (STN) [3]. The STN includes time points corresponding to the start and end of each task in an agent's task hierarchy, along with special time points representing the time origin and the current time. Various temporal constraints corresponding to task durations, release times, deadlines and causal dependencies are encoded in the STN as distance constraints. Scheduling decisions are implemented by posting additional precedence constraints in the STN between the set of activities currently selected for execution. These precedence constraints specify the agent's current schedule (or timeline).

At any point during operation, the currently installed schedule dictates the timing and sequence of domain-level activities that will be initiated by the agent, and the execution flexibility provided by the STN serves as a basic hedge against durational uncertainty. The Executor, running in its own thread, continually monitors the enabling conditions of various pending activities and activates the next pending activity as soon as all of its causal and temporal constraints are satisfied. The other three components run on a separate thread in a blackboard-based control regime and have responsibility for coordinating with other agents and managing the current schedule over time.

When execution results are received back from the environment (shown in Figure 1 as the MASS simulator) and/or changes to assumed external constraints are received from other agents, the agent's model of current state is updated. An incremental propagator based on [2] is used to infer consequences within the STN. If an update leads to inconsistency in the STN or it is otherwise recognized that the local schedule might now be improved, the Scheduler is invoked to revise the current solution and install a new schedule. To manage detected STN conflicts that are due simply to the asynchrony and potential latency of inter-agent communication, a domain-level interpretation of each conflict is used to determine which constraints to retract or temporarily suspend to allow the agent scheduler to continue (for details, see [6]). Whenever the agent's local schedule changes, either in response to a current state update or through manipulation by the Scheduler, the DSM is invoked to communicate these changes to those agents that share dependencies and have overlapping subjective views. After responding locally to a given state update and communicating consequences, the agent will use any remaining computation time to explore possibilities for improvement through joint schedule change or extension. The Options Manager utilizes the Scheduler (in this case in hypothetical mode) to generate one or more non-local options, i.e., identifying changes or extensions to the schedule of one or more other agents that will enable the local agent to raise the quality of its schedule. These options are formulated and communicated as queries to the appropriate remote agents, who in turn hypothetically evaluate the impact of proposed changes from their local perspective. In those cases where global improvement is verified, the agents commit to the joint changes.

4. PLANNING WITH LOCATED ACTIVITIES

In this section, we describe the mechanics of the scheduling and coordination components developed to cope with the execution of geographically dispersed tasks. Since tasks are physically situated and it takes time to travel between task locations, the agent's scheduling and coordination mechanisms are designed to minimize travel time where possible, and hence maximize the time spent on value accruing tasks.

4.1 Managing Local Schedules

The Scheduler consists of two basic components: a quality propagator and an activity allocator that work in a tightly integrated loop. The quality propagator analyzes the task hierarchy and collects a set of activities that (if scheduled) would maximize the quality of the agent's local problem. The activities are collected without regard for resource contention; in essence, the quality propagator optimally solves a relaxed problem where agents are capable of performing multiple activities at once. The allocator selects activities from this list in decreasing order of expected quality gain and attempts to insert each activity together with any additional enabling activities into the agent's schedule (also referred to as scheduling the activity). If the allocator fails in this attempt, the quality propagator is reinvoked with the problematic activity excluded.

When scheduling situated activities, travel between different activity locations must also be inserted into the agent's timeline. Since no quality is accrued for travel activities, the activity allocator is designed to minimize the time spent traveling in the interest of leaving more time available for quality accruing activities. This section describes the mechanisms used to determine where on the agent's timeline to insert a new located activity to meet this objective.

In general, the agent's schedule (or timeline) can contain both located activities and non-located activities (i.e., those that can be executed at any location). Between any two located activities $A1$ and $A2$ on the timeline, any number of non-located activities might be scheduled. The travel activities scheduled for moving from $A1$ to $A2$ do not necessarily have to appear adjacent to $A1$ or $A2$; when non-located activities are scheduled between $A1$ and $A2$, travel can be scheduled before or after any of these non-located activities.

When scheduling an activity A, a sub-interval on the agent's timeline (called a slot) that is within A's specified time window and large enough to be feasible is sought. Scheduling a non-located activity requires no changes to the travel activities that are already scheduled and on the timeline. In contrast, scheduling a located activity A will always require some deletion and insertion of existing travel activities, unless the slot found for A is situated between two located activities that have the same location as A.

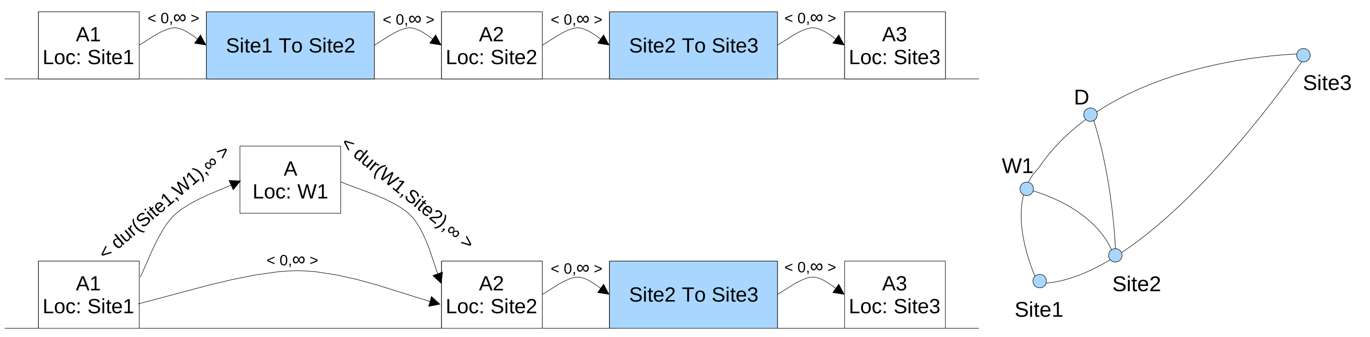


Figure 2: Slot selection: checking slot feasibility to insert activity A situated at $W1$ between $A1$ at $Site1$ and $A2$ at $Site2$. Delta travel added to the timeline if the slot is feasible is: $dur(Site1, W1) + dur(W1, Site2) - dur(Site1, Site2)$, where $dur(X, Y)$ represents the duration of travel between sites X and Y. The $< 0, \infty >$ arcs in the figure are precedence constraints. Right figure depicts the associated geographical layout.

The search for candidate slots for a located activity A proceeds forward in time over the agent's timeline. On each iteration the subsequence bounded by the next pair of consecutive located activities, $A1$ and $A2$, is examined. Specifically, the travel activities within this subsequence are temporarily unscheduled, and checks are performed to determine if A can be feasibly inserted into any slot between $A1$ and $A2$ (if there are no non-located activities scheduled between $A1$ and $A2$, there is only one possibility). If a feasible slot is found between $A1$ and $A2$, the additional travel time that would be incurred if A were inserted in this slot is recorded. This quantity, referred to as the *travel delta* ($\triangle tr$), reflects the difference between the duration of the new travel to be inserted on the timeline and the duration of the travel activities that were temporarily unscheduled. After checking all possible slots between $A1$ and $A2$, the original travel between these two activities is restored and the search continues with the next pair of located activities. After considering all subsequences that contain possible slots within A's time window, the slot with the smallest $\triangle tr$ is chosen for the insertion.

The check for feasibility of a given slot between two located activities $A1$ and $A2$ is complicated by the need to account for travel: there must be sufficient time available not only to perform A but also to travel from $A1$'s location to A's and then on to $A2$'s. To confirm feasibility, an attempt is made to post two new sequencing constraints in the underlying STN, reflecting the required travel duration associated with each travel leg. If both constraints can be successfully posted, the slot in question is feasible. In the simplest case where there is only a single slot between $A1$ and $A2$, the constraints are linked directly to these two activities (see Figure 2). However, determination of the predecessor and successor activities for these constraints is more complex when there are intervening, non-located activities. In this case, a search is conducted through consecutive activity pairs from $A1$ to A and from A to $A2$ (with A inserted in the slot being evaluated). The search is terminated either when both constraints are successfully posted (in which case feasibility has been confirmed) or there are no more activity pairs to be checked (in which case the slot is not viable). Once determination has been made, any temporarily posted constraints are retracted.

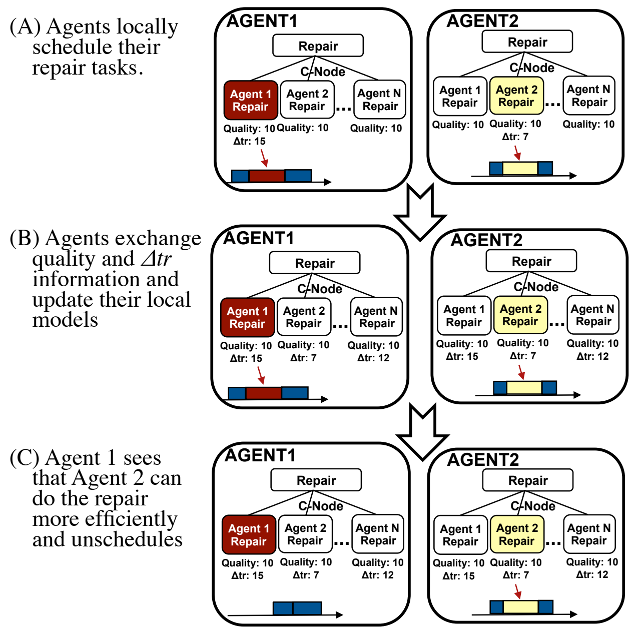
When unscheduling a located activity A, the timeline is modified in an inverse fashion. Travel activities to and from A are unscheduled and direct travel between newly adjacent located activities is substituted.

A special heuristic is utilized to handle the scheduling of located activities with enabling interdependencies, where the enabled activity can only be started after the enabling activity has been successfully completed. For such pairs of activities, selection of slots that minimize $\triangle tr$ independently (as described above) is myopic and can result in suboptimal solutions. Consider again the sequence of scheduled activities from Figure 2. Suppose that A enables an activity to be executed at location D (e.g., the agent needs to pick up a patient at location $W1$ and drop her at the hospital at location D). Also, suppose that $\triangle tr$ when traveling from $Site1$ to $W1$ and then to $Site2$ is shorter than the $\triangle tr$ when going from $Site2$ to $W1$ and then to $Site3$. Minimizing $\triangle tr$ to insert A independently misses the fact that the latter sequence of travel (from $Site2$ to $W1$ and then to $Site3$) becomes advantageous if the agent needs to execute another activity (drop the patient at site D) on its way from $W1$ to $Site3$. Accordingly, in the case of located activities with enabling interdependencies, the heuristic searches simultaneously over all sets of feasible slot assignments and minimizes the joint $\triangle tr$.

4.2 Arbitrating Task Assignments

While the above scheduling mechanism serves to minimize travel on an agent's timeline, it does not address the coordination issue of deciding which agent or agents should execute a task in the case where a task may be done by different subsets of agents. For example, in the natural disaster domain, suppose that three different agents can repair a power failure at a site. The challenge is in deciding which of these agents should actually attempt the repair. Excluding the need for redundancy, the goal in this case is to select the single agent that is the least constrained by scheduling the task and yields the most quality. The *C-Node Scheduling* coordination mechanism solves this problem by having all agents use the shared policy of first locally scheduling themselves for the task and then collectively selecting the one that maximizes a domain-specific objective function.

In C-Node Scheduling, a task that can be done by different subsets of agents is called a C-Node and is represented as a supertask with its children primitive tasks being each agent's copy of that task. When a C-Node warrants scheduling, i.e., is brought up by the quality propagator, each agent that owns a child of the C-Node, i.e., can execute it, schedules its best option, where best is determined by a given objective function. Those scheduling decisions along with their corresponding metrics necessary for computing the objective function value of those decisions are shared among the agents through the DSM. Each agent then computes the best overall child and unschedules its own child if it is not the winner.[2] In the case of non-located activities, the objective function is simply the quality earned, meaning the agent that schedules the highest quality activity is selected. For located activities, the objective function has to balance $\triangle tr$ and quality. For example, if executing the highest quality child requires substantial travel, it might be better to forgo that one in favor of one with slightly less quality but significantly reduced travel, i.e., leaving more room on the timelines for other quality-accruing activities. The objective function for located activities combines $\triangle tr$ and quality by using domain-specific equivalency classes among quality ranges, such that qualities that differ within a certain amount are considered equal and $\triangle tr$ becomes the differentiator. Included in these objective functions are a series of tie breaking criteria used to guarantee a unique winner for any C-node competition. Figure 3 shows a C-Node Scheduling example where two agents can do the same repair task but only one is required.



(A) Agents locally schedule their repair tasks.
(B) Agents exchange quality and Δtr information and update their local models
(C) Agent 1 sees that Agent 2 can do the repair more efficiently and unschedules
AGENT1
AGENT2
Repair
C-Node
Agent 1 Repair
Agent 2 Repair
Agent N Repair
Quality: 10
Δtr: 15
Δtr: 7
Δtr: 12

Figure 3: C-Node Scheduling Example

Safeguards are introduced to prevent a couple of deleterious effects that can result from C-Node Scheduling being repeatedly invoked through multiple reschedulings as execution unfolds. The first effect occurs when an agent begins traveling towards a located activity. It is possible that, while that agent is traveling to the location of the activity, it might lose the C-Node and then travel to the location for no purpose. To avoid this spurious travel, C-Node Scheduling immediately locks the winner of the C-Node once an agent commences its travel to the location of the C-Node.

The second detrimental effect occurs when, due to particular states on the timelines of different agents, cycling behavior can occur in C-Node Scheduling. This cycling happens, in part, due to the use of an STN. Since reservations are not locked on the timeline, it is possible for a scheduled activity to move later on the timeline when another activity is scheduled as long as no hard constraints are violated. If the activity slides later, the agent may lose the C-Node due to one of the tie breakers being the earliest start time of the scheduled task. But, unscheduling the activity frees up room on the timeline, allowing the task to once again be scheduled at the earlier time. This cycle will continue until one of the agents actually starts executing either the travel to the activity or the activity itself. To suppress this effect, we imposed a limit on how often an agent can lose to another agent for a specific C-Node before it no longer competes. Since new opportunities may arise over execution, periodically the limits are cleared, so that the C-Node Scheduling can search the full space, again.

4.3 Synchronizing Tasks of Multiple Agents

One particularly difficult coordination element is synchronizing agents to start interdependent activities within some acceptable time delta. For example, consider again the natural disaster domain in which a repair for a power utility at a site requires two agents working in unison. If the two agents do not work together, then quality cannot be accrued for the overall synchronized task.

The *Sync Activator* mechanism achieves this coordination using a contract-net protocol approach [9] for scheduling the synchronization for these activities. When, in local scheduling, a synchronized task is determined to be worthwhile, the agent that is assigned the responsibility for the task, i.e., the auctioneer, makes an announcement to all the relevant agents soliciting bids from each agent that can do part of the synchronized task. A bid summarizes the cost in terms of quality and $\triangle tr$ for an agent to schedule its part of the synchronized task within a given window of time. Each agent generates a bid by hypothetically scheduling with the constraints stated in the announcement and calculating the costs from the resulting schedule. The auctioneer compares the bids and selects the best set of bids that satisfy the constraints of the synchronized task. Like C-Node scheduling, "best" is determined by a domain-specific objective function. Once a set of bids is accepted, then the involved agents schedule their corresponding activities, constraining them such that they cannot move on the timelines beyond the acceptable start-time delta.

One issue that arises with scheduling these synchronized activities is that constraining them on the timeline introduces rigidity to the STN. In highly dynamic and uncertain environments, that rigidity can affect scheduling downstream activities in that they have to be placed around the synchronized activities. If execution realities differ from scheduling assumptions, e.g., durations for activities are longer, then the synchronized starts can fail, meaning no quality is earned for the synchronized tasks nor for activities that do not get scheduled due to the scheduled synchronized activities. We addressed this issue in the following two ways. First, we limit the synchronized scheduling to only schedule activities within a limited horizon. If the synchronized activities cannot be scheduled within that horizon, then they are tried again in a subsequent scheduling session. Second, we

[2] We refer to the agent that owns the child that is selected to be scheduled as the winner of the C-Node.

allow synchronized activities to be removed from the timeline if execution realities make the synchronization impossible, i.e., the hard constraints of the activities are violated. The synchronization coordination can be tried again in a subsequent scheduling should that task still be considered important.

5. PERFORMANCE EVALUATION

5.1 Experimental Design

A meaningful multi-agent evaluation depends heavily on the creation of a test suite unbiased by a particular agent design. We report here on evaluations based on a large suite of real-time simulation problems generated and run by an impartial third-party evaluation group. These problems, which reflect a progression in problem scope over the lifetime of the Coordinators program, define large agent team scenarios that feature both located and non-located tasks, incorporate network level communication latency and reflect a range of dynamics and inter-agent dependencies.

The need to run a statistically significant suite of tests for each team and problem feature of interest required adoption of short activity durations (5 - 15 sec) to keep overall problem duration manageable. As a consequence, for those problems with high levels of dynamics (e.g. modeling missions whose goals evolve significantly) mission change events may occur at intervals on the order of a few seconds.

All evaluation problems were randomly generated using a scenario generator that can be configured to produce problems reflecting an assortment of user-specified characteristics. Evaluations were conducted in an external lab with each agent and the MASS simulator running on dedicated machines. Before execution begins MASS provides agents with their local, subjective view of the problem and an initial schedule of activities that was generated by a centralized solver with a global view of the mission. As the scenario unfolds the agents send commands to the simulator specifying the activities they want to initiate and the simulator samples duration and outcome quality (0 for failed activities) from the distributions in the objective *C_TAEMS* model. The simulator will also send agents their subjective view of changes to the mission at intervals specified in the objective view but hidden from the agent teams.

The phase 3 Coordinators final evaluation was composed of two test suites: 1) Problems designed to be small enough to be solvable by an optimal, centralized solver and 2) A range of much larger problems on which teams were compared against each other. The evaluation team solved the "optimal problems" using an MDP-based solver. Although the 'optimal' test suite was restricted to small problems, this assessment provided an upper bound to test the agent systems against. In addition to the much larger scale of the principal test suite problems, they also featured communications latency simulation and dynamics such as mission change events that were precluded from the optimal problems. Since optimal scores are infeasible for these problems, our agent is compared against the dispatching approach developed by the other phase 3 team, referred to as criticality-sensitive coordination[8]. Rather than plan ahead, measures of the criticality of various tasks eligible for execution are instead computed from information that is dynamically and continuously communicated between agents, and used to decide what activity to execute next whenever the agent is idle.

The test suite problems were created in four or five classes,

Problem Class	**Description**	**Scheduling Agent/Opt.**
OptSynch (32 probs)	sync tasks, no NLE chains	98.5%
OptMovement (32 probs)	located tasks emphasize travel	98.1%
OptContingency (32 probs)	NLE targets depend on source outcome	100.0%
OptBlend (32 probs)	mix of NLE, movement, sync	98.7%
Overall		98.8%

Table 1: Scheduling agent performance on optimals.

where each class was designed to reveal agent performance on certain features of interest in the problem space. We describe these classes in presenting evaluation results in the next section.

5.2 Results

The four problem classes of the optimal test suite are briefly described in Table 1. Careful reasoning about movement to achieve the optimal score is mainly required in the OptMovement class. OptSynch emphasizes simple high-level tasks under which multiple agents must start their activities simultaneously for the task to accrue quality. OptContingency emphasizes "contingency" structures; enables NLEs having multiple target tasks, only one of which is enabled based on outcome of the source-side task. OptBlend problems mix a variety of problem space features, including some movement aspects. Our agent scored an average of 99% of the optimal value across the 128 problems in the four categories of the test suite, easily meeting the Coordinators program goal of 90% of optimal.

Turning now to the large scale test suite scenarios, the 158 trials fall into 5 classes as outlined in Table 2. All five categories include problem instances that pose a range of both movement complexity challenges and induced communications latency. SmallTests features simple versions of specialized task structures for which early decisions in a task chain can greatly reduce the quality accrued much later. The Contingency class features much more complex versions of the NLE outcome structures described above for the optimals. RandomDynamic scenarios have a high concentration of unexpected events and changes in mission and task constraints. Mixture30AG scenarios contain a blend of many different features from the problem space, including a fair amount of dynamics. Synchronization focuses on more complex versions of the OptSynch features in which a high-level task requires agents to synchronize the start of activities for multiple subtasks in order to accrue task quality.

As Table 2 suggests, in an environment where agents are both mobile and spatially distributed our advance planning pays off in maximizing reward by minimizing travel. In all but one problem class the scheduling approach dominates the criticality-sensitive dispatching by an overall average of 20%. The single category for which the dispatching approach outperforms, SmallTests, is not surprising in that the role of locations is minimal for these problems.

Recalling that problem classes include problem instances featuring a range of both movement challenges and communications latencies, it is informative to compare performance across these variables. Figure 4 plots the relative performance of the two approaches as both the travel demands

Problem Class	Description	Criticality Sensitive Coord.	STN-Based Scheduling
SmallTests - 33 trials - 22 agents	-specialized constructs minimal movement	100.0	87.7
Contingency - 33 trials - 30 agents	Many contingent NLEs, fewer tasks	84.3	94.7
Random Dynamic - 33 trials - 20 agents	Many mission changes, fewer NLEs	76.7	100.0
Mixture30AG - 27 trials - 30 agents	Blend of features	52.9	74.6
Synchronization - 32 trials - 30 agents	Multiple sync tasks Many tasks & NLEs	43.1	100.0
Overall		71.4	91.4

Table 2: Comparative performance in Coordinators Phase 3 lab evaluation. Score of approach a over n trials is computed by $100 \times \sum_{t=1}^{n} \frac{Quality(a,t)}{MaxQuality(t)}/n$

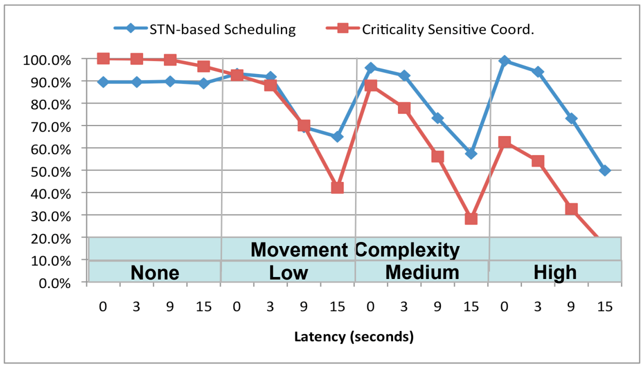


Figure 4: Impact of communication latency.

increase due to task dispersal and communications degrade. The problems are categorized here by the evaluation team into one of four movement complexity classes and four different latency classes, ranging from no latency to 15 seconds. Not surprisingly, the performance of both multi-agent teams generally degrades as latency increases, though by minor amounts for scenarios in which reasoning over travel is not critical. The more interesting performance differential surfaces as the necessity to reason over travel escalates. Here, not only does the scheduling approach dominate, but the benefit of planning ahead is seen to increase as problems feature a more challenging geographic task dispersal.

6. EXTENDING TO FIELD EXERCISES

A second major challenge established for Phase 3 of the Coordinators program was to evaluate the agent in a field test exercise, designed to resemble the disaster rescue scenario outlined at the outset of this paper. Three mock disaster rescue scenarios were designed over a layout of 17 physical sites in a suburban park in the Washington DC area. Particular sites were designated as potential damage sites, a hospital, a clinic (initially not operational) and warehouses (containing infra-structure repair kits). In each scenario run, a team of 8 human agents with heterogeneous capabilities (e.g., gas specialist, survey specialist, ambulance, etc.),

Figure 5: The field test agent.

were charged with surveying sites, transporting discovered casualties to operational medical facilities and fixing discovered infra-structure problems. Points were accrued for successfully resolving any discovery (i.e., casualty rescues, restoration of services). A 90 minute time limit was imposed on each run, creating an oversubscribed problem and forcing the agent teams to make choices about which tasks to perform. Each scenario was run twice, once with a team equipped with our "Coordinator Agents" and once with a second team equipped only with radios. The radio team was allowed to coordinate through normal voice communication over the radios and to centralize planning through a human commander, while the "Coordinators" team was, essentially, effectors, only performing tasks prescribed by our agents. The goal was to score higher than the radio team.

Significant extensions to our agent were required to address this field test problem. The agent's ability to reason with locations was augmented to handle additional aspects of state (e.g. agent carrying state, service operating status at sites), to make travel-efficient kit sourcing decisions, and to track locations of repair kits. A dynamic plan instantiation capability was developed for generating casualty transport and service repair task structures in response to discovered problems. A user interface was designed and implemented for conveying current tasks to the agent's human user, and for accepting user observed results of executed actions. Extensive efficiency and robustness improvements were made to the agent's communication infra-structure to enable operation with cell modems. Figure 5 shows a snapshot of the final field test agent.

During actual field trials with our extended agent, we were only partially successful, achieving a score that exceeded that of the radio team in only the first of three scenarios that were run. Key to the success of our initial run was our ability to supply the agent with a well designed initial plan, specified in terms of an initial itinerary for each agent and a complex set of agent preferences and priorities with respect to repair and casualty transport roles at various sites. Exercise rules allowed us to see the scenarios 72 hours in advance of the first run. However, due to the primitive nature of available tools for specifying team strategy, all of our effort in this 72 hour period went into the first scenario. After the field test, comparable plans were developed for the 2nd and 3rd scenarios, and used to run repeated simulations

Scenarios	**Radio Team**	**Agent Team**	**Description**
Exercise-3	6100	6150	At field test
Exercise-5	6925	7725	Post test simulations (ave. of 3 runs)
Exercise-6	4575	4650	Post test simulations (ave. of 3 runs)

Table 3: Field test scenario results

.

of each of these latter two scenarios (assuming a rather extreme 20% variance in task durations). In the end, we were able to achieve a level of performance comparable to that of the radio team (see Figure 3).

7. CONCLUSIONS AND FUTURE WORK

In this paper we have argued that, in domains where agents are both mobile and spatially distributed and where sequence-dependent setup activities such as agent travel are required to perform target tasks, there is inherent leverage in maintaining advance schedules and using them to drive coordination. We described an approach to distributed coordination of mobile agent teams based on this premise, which couples an incremental STN-based agent scheduler with inter-agent task allocation and synchronization mechanisms that promote minimization of overall travel time, and attempt to maximize the time that the agent team spends executing value accruing tasks. We presented experimental results that compared the performance of our agent with that of an agent operating with an intelligent dispatching strategy driven by task criticality metrics, which had previously been shown to significantly outperform an advance scheduling approach on synthetic, stateless, quality maximization problems. Across a broad range of problems where tasks are physically situated in the world and the cost of agent travel must be considered, the comparative results are reversed and our advance scheduling approach is shown to dominate. Finally, we reported results obtained in a mock disaster rescue field exercise to demonstrate the practical viability of the approach.

Following the recent experience with the field test exercise, our current research is focused in two directions. The first concerns development of frameworks and mechanisms for providing strategic guidance to collaborative agent teams. In highly dynamic environments where there is considerable latitude in potential task assignments to different agents, and agents must react quickly to keep pace with execution, the combinatorics of the distributed search problem can quickly lead the agent team toward subpar solutions if not provided with appropriate global structure and constraints. Our vision is a high-level interface through which a human can impart strategy and dynamically steer the agent team. A second direction of current research is investigation of greater use of organizational structure and decision centralization within our distributed schedule management framework. In some sense, the goal of the Coordinators program was to explore the limits of the extreme position where all agents are restricted to local subjective views. However, in many settings there is no reason not to exercise the advantage of more decision-making control.

Acknowledgements. This paper is based on work supported by the Department of Defense Advance Research Projects Agency (DARPA) under Contract # FA8750-05-C-0033. Any opinions findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

8. REFERENCES

[1] M. Boddy, B. Horling, J. Phelps, R. Goldman, R. Vincent, A. Long, and R. Kohout. C_taems language specification v. 2.04, April 2007.

[2] A. Cesta and A. Oddi. Gaining efficiency and flexibility in the simple temporal problem. In *Proc. 3rd Int. Workshop on Temporal Representation and Reasoning*, Key West FL, May 1996.

[3] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, May 1991.

[4] K. Decker. TÆMS: A framework for environment centered analysis & design of coordination mechanisms. In G. O'Hare and N. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 16, pages 429–448. Wiley Inter-Science, 1996.

[5] K. Decker and V. Lesser. Designing a family of coordination algorithms. In *Proceedings 1st International Conference on Multi-Agent Systems*, pages 73–80, San Francisco CA, 1995.

[6] A. Gallagher and S. Smith. Recovering from inconsistency in distributed simple temporal networks. In *Proceeding 21st International Conference of the Florida Artificial Intelligence Research Society*, Coconut Grove, FL, May 2008.

[7] D. Goldberg, V. Cicirello, M. Dias, R. Simmons, S. Smith, and A. Stentz. Market-based multi-robot planning in a distributed layered architecture. In *Multi-Robot Systems: From Swarms to Intelligent Automata- Volume 2*, pages 27–38. Kluwer Academic Publishers, 2003.

[8] R. Maheswaran and P. Szekely. Criticality metrics for distributed plan and schedule management. In *Proceedings 18th International Joint Conference on Automated Planning and Scheduling*, Sydney Australia, 2008.

[9] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, Dec. 1980.

[10] S. Smith, A. Gallagher, T. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed management of flexible times schedules. In *Proceedings 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Honolulu, 2007.

[11] A. Wehowsky, S. Block, and B. Williams. Robust distributed coordination of heterogeneous robots through temporal plan networks. In *ICAPS-05 Workshop on Multiagent Planning and Scheduling*, pages 67–72, 2005.

[12] R. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research*, 25(1):1–25, January 2006.