Distributed Scheduling Agents for Disaster Response

Laura Barbulescu[1], **Zachary B. Rubinstein**[1], **Stephen F. Smith**[1], **David E. Wilkins**[2], and **Terry L. Zimmerman**[1]

[1]The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15024
{laurabar,zbr,sfs,wizim}@cs.cmu.edu

[2]SRI International
Artificial Intelligence Center
333 Ravenswood Ave.
Menlo Park, CA 94025
wilkins@ai.sri.com

Abstract

In this paper, we describe the application of a multi-agent framework for collaborative scheduling to a disaster response coordination problem. The target problem is a field exercise mockup of a natural disaster, where a team of human agents must rely on their respective automated scheduling agents to coordinate and accomplish various infra-structure repair and casualty transport tasks. Following the ground rules of the program sponsoring the experiment, our starting point is a collaborative scheduling framework developed under the strong assumption that no single agent has a global view of the overall problem. This peer-to-peer approach to multi-agent scheduling and coordination gives rise to a complex distributed search problem, and effective performance in the field exercise is found to depend heavily on the ability to provide the multi-agent system with strong initial strategic guidance. We describe the mechanisms developed for *steering* the multi-agent scheduling system to address specific disaster response scenarios and report performance results that were obtained in the field exercise test cases.

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 response horizon. Furthermore, communications may have latency and periods of blackouts.

In this paper, we consider the application of a multi-agent framework for collaborative scheduling to such disaster response problems. We focus specifically on the field exercise problem defined by the DARPA Coordinators program. In this setting, a team of human agents must rely on personal scheduling agents (provided on portable devices that the human agents carry) to coordinate their respective actions to achieve maximum effect within a stated execution window. Points are accrued for restoring failed infrastructure services and for successfully transporting casualties to operational medical facilities and the objective is to maximize accrued points. The programmatic goal of the Coordinators field exercise was to beat a second team of human agents equipped with radios and exploiting a centralized commander.

Following the design constraints of the Coordinators program, our approach to addressing the field exercise problem builds on a distributed scheduling framework where no single agent has a global view of the problem and solution (Smith et al. 2007; Barbulescu et al. 2010). Instead, each agent builds and maintains a task schedule based on its local view, and coordinates with other agents that share task dependencies to improve the utility of their joint actions. Such an approach has pros and cons with respect to the target field exercise domain. On one hand, it provides inherent robustness against unexpected events; if an agent becomes unable to fulfill a task that it has taken on (called a "self-injury" in the field test), the newly unscheduled task will immediately be noticed and picked up by another capable agent. On the other hand, the scale and complexity of the distributed search problem can quickly lead agents to sub-optimal coordination decisions.

One approach to dealing with the scale and complexity of the distributed coordination problem is to exploit human strategic guidance. In the disaster response domain of interest, for example, a range of information about the scenario (e.g., the likelihood of problems and casualties at different sites, the locations of hospital and clinic sites) is provided in advance of execution and can be used to globally struc-

ture the initial plan that is distributed to the agent team, to physically disperse the team of agents in anticipation of discoveries, and to set specific task priorities and preferences for specific agents. By providing basic capabilities for *steering* the agent team, it is possible to effectively manage the distributed coordination problem while retaining the robustness to respond to unanticipated events.

The remainder of the paper is organized as follows. We start by describing the disaster response problem defined for the Coordinators program field exercise. We then summarize the underlying multi-agent framework for collaborative scheduling that is taken as the starting point for addressing this problem. The mechanisms developed for injecting strategic guidance into the agent system to manage the complexity of distributed search in specific disaster response scenarios are described next, and performance results obtained on the field exercise test cases are then presented to illustrate the effectiveness of these basic steering mechanisms. We conclude with a short discussion of current research directions.

Problem Description

The Coordinators field test exercise was 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 18 physical sites in a suburban park in the Washington DC area (Figure 1). Sites that are potential damage sites include the two clinics (initially non-operational) and all numbered sites except the warehouses (which contain infra-structure repair kits) and the FRC, which contains an operational hospital and is the mission start site for the team. Three of the sites contain a service main (gas, power, or water) which can only be surveyed and restored by the single dedicated specialist for that service. Before a specific service local to a site can be restored its associated main must be surveyed, repaired as needed, and restored.

The disaster rescue team consists of 8 human agents with heterogeneous capabilities (e.g., gas specialist, survey specialist, ambulance, etc.), charged with surveying sites, transporting discovered casualties to operational medical facilities and fixing discovered infra-structure problems. Points are accrued for successfully resolving any discovery (i.e., casualty rescues, restoration of services).

Figure 2 indicates the various constraints associated with processing a site and suggests the scope of the coordination problems. Briefly, gas and power must be isolated before any surveys at the site can be conducted, and must be carried out by an agent possessing the requisite capability. A casualty survey can turn up some number of serious and critical casualties worth varying numbers of points if they can be transported to a medical facility before their associated expiration dates. Infrastructure surveys can turn up service problems worth varying numbers of points, and the repairs may involve up to two agents with appropriate capabilities, and might require a repair kit, obtainable from a warehouse.

Once all problems of a given service have been successfully completed, the service can be restored and repair points are accumulated. In special cases, restoration of power at a site requires synchronized action with another agent located

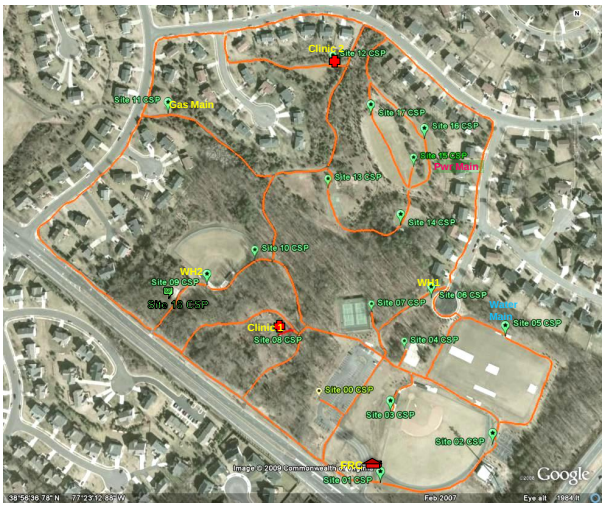

Figure 1: Site and Facilities Layout for Example Scenario.

at the remote power main site. Unlike the hospital, a clinic is initially non-operational until all services at its host site are surveyed and any damages repaired. Casualties evacuated to clinics cannot be treated until the clinic is operational.

Unexpected events that can be discovered at a site include a self injury that requires an immediate visit to an operational medical facility and a broken vehicle which pins the agent at that site for a designated period of time. During movement between sites, "road blocks" can be encountered which require cooperation between 2 agents to clear.

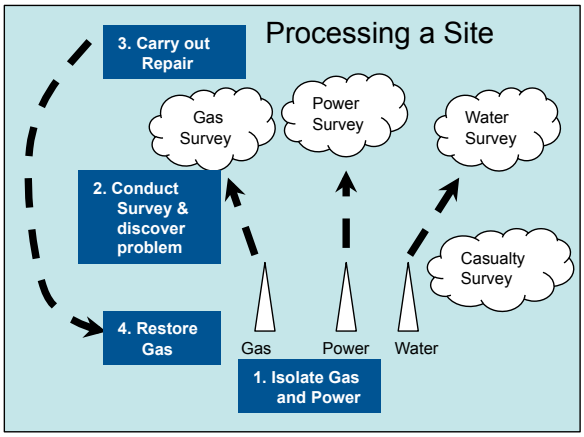

Figure 2: Processing a site in the field test.

Technical Approach

Architecture

Our approach to maintaining advance schedules is based on the cMatrix agent architecture originally described in (Smith et al. 2007). The field exercise configuration is depicted schematically in Figure 3. An agent is com-

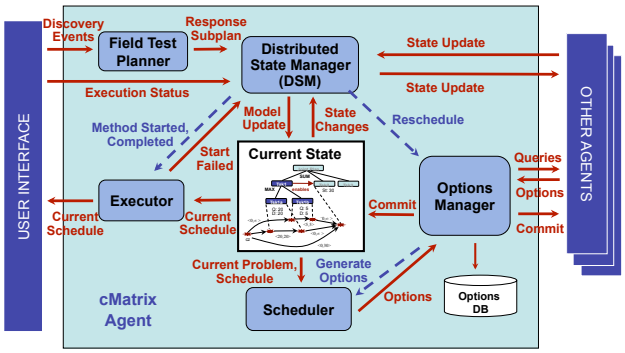

Figure 3: Agent Architecture.

prised of four core 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 (encoded as a CTAEMS task structure (Decker 1996; Boddy et al. 2007)) to an underlying Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991). 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 communicates their eligibility to be executed to the user interface as soon as all relevant 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 either execution results are received back from the user or changes to assumed external constraints are received from other agents, the agent's model of current state is updated. One stream of user feedback indicates simple execution status: when a given activity is started, when it is completed, and whether or not it was successfully completed (e.g.,whether an attempted infrastructure repair succeeded). The user also communicates any discovery events that occur as a result of executing a survey activity (e.g., casualties to be transported, infrastructure problems, broken vehicle). In these cases, a fifth component of the agent, the Field Test Planner (FTP) is invoked to generate a CTAEMS task structure for responding to each event. The FTP uses a pre-computed library of plan schemata that covers the set of possible discovery events/problems that might occur together with user communicated event parameters (e.g., casualty delivery deadline, type of repair problem) to instantiate sub-plans for the subset of agents capable of responding to it. These subplans are then communicated to the relevant agents (including the discovering agent if appropriate) as model updates [1]

An incremental propagator based on (Cesta and Oddi 1996) is used to infer the consequences of model updates 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 (Gallagher and Smith 2008)). 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. In the field exercise setting, this mechanism provides a basis for coordinating activities (e.g., repairs) that must start simultaneously (see below).

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 hierarchy of activities that the agent could perform and collects a set of activities that (if scheduled) would maximize the quality of the agent's local problem (i.e., lead to accumulation of the most points). The activities are collected without consideration of the agent's available capacity ; in essence, the quality propagator optimally solves a relaxed problem where the agent is capable of performing multiple activities at once. The allocator then selects activities from this list in decreasing order of expected quality gain and attempts to insert each

[1] The FTP is also used offline to generate initial survey plans for a given field exercise scenario.

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 like those in the field exercise, 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.

More specifically, when adding a new activity A to the agent's schedule, a sub-interval on the agent's timeline (called a slot) is sought that (1) is within A's specified time window, (2) is large enough to allow A to be feasibly executed and (3) minimizes the additional travel time that is incurred by the agent, referred to as the *travel delta* ($\triangle tr$). For each pair of consecutively scheduled located activities $A1$ and $A2$ falling within A's time window, any intervening travel activities are temporarily removed and an attempt is made to post sequencing constraints from $A1$ to A and from A to $A2$ with minimum separation times corresponding to travel time from $A1$ to A and A to $A2$ respectively (see Figure 4). If these constraints can be successfully posted (indicating that the slot is feasible), then $\triangle tr(A1, A2)$ is computed and the original travel between $A1$ and $A2$ is restored. If at least one feasible slot is found in this traversal, the slot with the minimum $\triangle tr$ is chosen.

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 4. 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 $\triangle tr$.

In addition to reasoning about locations and travel, the scheduler also tracks the agent's *carrying state* to enforce its capacity constraint (only one item - casualty or repair kit - can be carried at a time). When inserting a new pair of pickup/drop-off activities into the timeline, additional checks are performed to ensure that no two pickup activities are consecutively scheduled without an intervening drop-off activity.

One final aspect of state that is managed are inventories of repair kits at various locations. During execution, agents can strategically place kits at locations where they might be needed. A designated agent called the *resource manager* interacts with the rest of the agent team to maintain a record of the types and numbers of kits available at various locations. When an agent schedules a pickup of a repair kit at a given site, it issues a reservation request to the resource manager. This reservation request is approved if a kit of the desired type remains available at the designated site; and is denied if it has already been reserved by another agent. The Executor ensures that a kit pick-up can only be executed after a reservation approval has been issued by the resource manager.

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

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

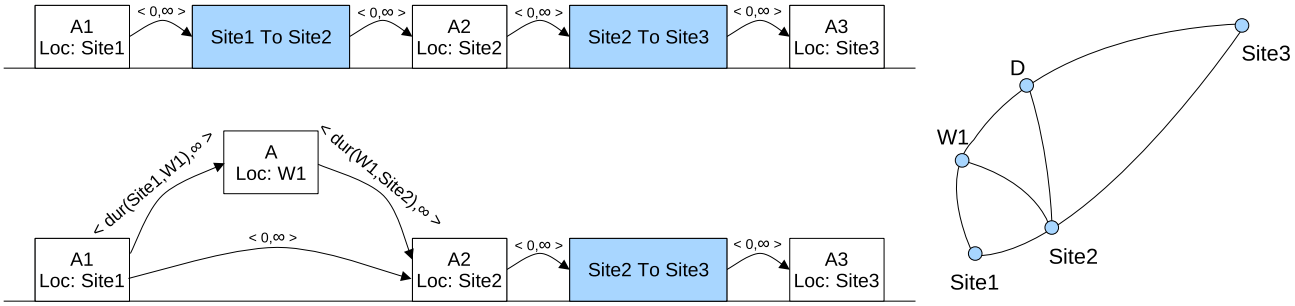

Figure 4: Slot selection: checking slot feasibility to insert activity A situated at $W1$ between $A1$ at $Site1$ and $A2$ at $Site2$. Delta travel added to the timeline if the slot is feasible is: $dur(Site1, W1) + dur(W1, Site2) - dur(Site1, Site2)$, where $dur(X, Y)$ represents the duration of travel between sites X and Y. The $< 0, \infty >$ arcs in the figure are precedence constraints. Right figure depicts the associated geographical layout.

breaking criteria used to guarantee a unique winner for any C-node competition. Figure 5 shows a C-Node Scheduling example where two agents can do the same repair task but only one is required.

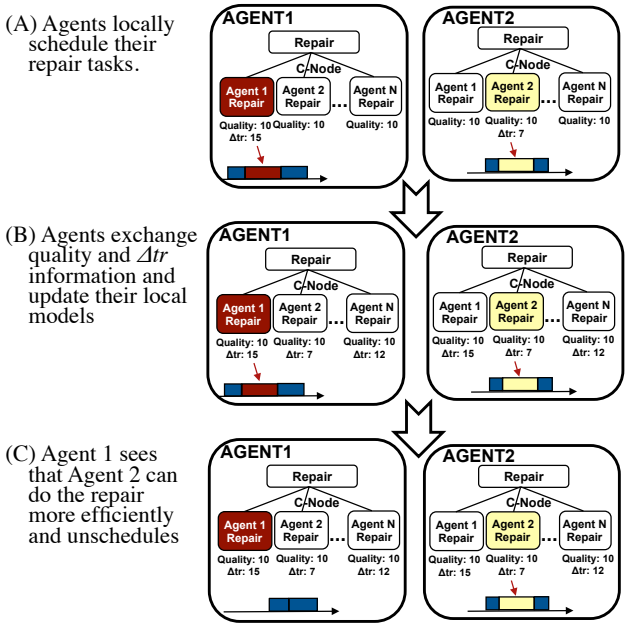

Figure 5: C-Node Scheduling Example

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.

Synchronizing Tasks of Multiple Agents

One particularly difficult coordination element is synchronizing agents to start interdependent activities within some acceptable time delta. For example, in our natural disaster domain a repair problem for power service at a site may require 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 (Smith 1980) 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 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 and no quality is earned for the synchronized tasks or for activities that were not 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 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.

Steering the Agent System

A significant challenge in these field scenarios is the uncertainty of what types of problems will be discovered and the time it will take to rectify them. Before execution begins, the sites, the topography, and the possible service problems are known, but unknowns include the specific problems, which repairs will be successful, what capabilities will be needed at a site,the number and location of casualties, and how long it will take to process the sites. In fact, the initial schedule will contain only the activities associated with gaining safe access to and surveying the sites. It is not until the FTP gets the survey results that it generates the model corresponding to those discoveries. The scheduler generates good solutions based on the model it is given without considering the likelihood of future events. As an example of this myopic decision-making, consider the allocation of two agents, the Power Specialist, who can perform both major and minor power repairs, and another agent with only minor repair capability. The scheduler may assign the Power Specialist to do a minor power repair at a site because it is slightly closer than the other capable agent. However, this decision ignores any likelihood that the Power Specialist may be needed to do a repair that requires a major power capability at another site that will soon be surveyed.

The piece not represented in the model is the strategy humans devise to address this kind of uncertainty. Human planners will decide how they want to do the exploratory search, often choosing particular patterns that hedge against the uncertainty while accepting a modest trade-off in optimality. For example, human planners in this domain must consider what sites the agents will initially survey, the assignment of survey agents to these sites such that sufficient capabilities will be nearby to resolve the types of problems that might arise, whether planning for infrastructure repairs is worthwhile, etc. This human-developed strategic guidance will have the greatest impact on the scheduler if it encompasses the entire mission, from building the initial schedule[3] through all the replanning episodes that arise during execution. To provide this bridge between the human strategy and the scheduler, we extended cMatrix to accept a set of human-generated tactics specified upfront and to generate corresponding constraints that will restrict the scheduler to search for solutions consistent with the overall strategy. The human planner currently uses a text-based protocol for expressing these tactics, a process that could clearly be greatly improved through a graphical interface accompanied by a hypothetical planning tool.

The remainder of this section describes the primary types of tactics that were found to be most effective in shaping scheduler performance in the disaster rescue domain as well as the scheduler mechanisms that enforce them. We then present a simple example of a set of tactics for restoring the gas main.

- **Ordering Priority (OP)** - A value indicating a temporal order that must be respected between an agent's scheduled activities; an activity cannot be inserted after an activity with a lower priority unless that activity has already executed. Ordering priorities are one tool used to ensure that some activities, if they are scheduled, are done earlier than others for a given agent. [Example: Human planners can specify that an agent should perform repairs that may be discovered at one site before moving on to survey another site.]
- **Agent Bias (AB)** - A value used for promoting or demoting an agent relative to other candidates that can perform a given task. Specifically, the agent bias value is used to adjust up or down the apparent quality of an agent's activity under the C-Node task. C-Node scheduling favors higher quality activities, scheduling the spatially closest highest value activity for which there is sufficient room on the timeline. [Example: Human planners can specify that Agent1 is preferred for a particular repair before another capable agent.]

The next two tactics, 'gates' and 'conditioned survey releases', are predicated on the agents sharing state (isolated, surveyed, repaired, restored) about the sites. In cMatrix, this state sharing is done via a bit vector that encodes the current state of each of the sites. When any agent changes the state of a site, it sends the updated vector to the agents, who then update their local vector through an XOR operation. If, when the state is updated, a gate is executing, the gate's condition is checked against the new state to see if it is satisfied. If it is, then it is automatically completed.

- **Gate** - A construct that, when set for an agent and a particular site, precludes it from moving beyond that site to other located activities on its timeline until a speci-

[3] A global scheduler, based on the same principles as the agent scheduler, is used offline to generate a joint initial schedule for distribution to the agents

fied condition is met. Gates are implemented as pseudo-activities on an agent's timeline that are both started and completed automatically. A key role for gates is in supporting a strategy aimed at keeping a sub-team of agents with complementary capabilities near to each other for efficiency in tackling repairs that may arise. [Example: Human planners can specify that a specific agent wait at a site until another site's service is restored.]

- **Conditioned Survey Release (CSR)** - A construct used to make the survey activities for a specified site invisible in an agent's model until *after* a specified condition occurs. When an agent updates its site state vector, if any conditioned survey release is satisfied by the new state then the corresponding activities are added to the model. The CSR gives the human planner some control over how much of a mission gets scheduled in advance, enabling a strategy of segmenting it into stages. Agent timelines can become tight and leave little room for activities arising from new discoveries if all the site surveys are scheduled at the outset. [Example: Human planners can specify that the survey activities for a site will not appear in the model until another site's services are fully restored.]
- **Earliest Start Time (EST)** - Used to prevent an agent from starting a specified activity before an absolute mission clock time. EST's provide the planner with a second tool (along with CSR's) for establishing mission stages. Human planners will often position agents to respond to key high-value discoveries in early/mid mission, but before these activities are discovered the scheduler for an idle agent -seeking whatever reward is available- could work against such strategies and send agent traveling after low value actvities. [Example: Planner can specify that the important Power Specialist not schedule low quality repairs that may be discovered at a remote site until the last 10 minutes of the mission, if at all.]
- **Kit Staging (KS)** - Kit staging is used to force the scheduler to allocate a given agent to pickup a repair kit of a given type at a warehouse and deposit it at a particular site before a specified deadline. Since the protocol also also supports specification of an OP value the KS tactic gives human planners two useful tools: 1) The planner can stage particular agents and kits to key sites in advance of planned surveys that may reveal repair problems requiring the kits. 2) It can be used as a positioning mechanism to ensure an agent travels a certain route and visits a specified site outside the vagaries of the online scheduling process. Kit staging is often used in conjunction with a gate; an agent with a particular capability carries a kit to a site and then waits to be released by a gate conditioned on completion of the survey for which the agent and kit were staged. This is a valuable stratagem for key sites since it guarantees that, should a repair be required for the target service that requires two agents and a kit, it can be immediately executed.

Figure 6 shows a sample of human planner guidance to the cMatrix scheduler in the specification protocol. This segment is focused on restoring the Gas Main at Site-11 (Figure 1 as one of the first high-priority tasks of the mission.

```
Gas Specialist and SS3 pickup gas kits and carry to gas main--
:kit-staging
     (Gas-Specialist kit: Gas-Repair-Kit Site-11 OP:100)
     (Survey-Specialist-3 kit: Gas-Repair-Kit Site-11 100)

Stall agents that may be needed to repair/restore Gas Main until survey by
Gas Specialist is complete--
:gates ((Survey-Specialist-3 Wait-for-GS-Survey-Results(Gas-Main))

Segment the mission into phases:
 * delay visibility of Clinic-1 surveys until after Clinic-2 is restored--
:model-release-times
((Site-08 Gas, Power, Water (Site-11 Clinic-2 Restore)))

 * prevent diversions to low-quality activities until late in mission--
:EST (Site-04 (Gas-Specialist (serious-casualty-rescue EST 5400)))
```

Figure 6: Portion of human planner guidance to cMatrix.

Results

Three similar but separate mock disaster rescue scenarios were run on consecutive days in a suburban park in the Washington DC area. A 90 minute time limit was imposed on each run to create an oversubscribed problem and force the agent team to make choices about which tasks to perform. The goal, as previously mentioned, was to score higher than a second human 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. Our team was instructed to act essentially as effectors, and only perform tasks prescribed by our agents. Our system was configured to run on a set of 9 ruggedized laptops (1 for each of 8 field agents and a 9th passive resource manager agent) communicating with each other via cell modems. Figure 7 shows a snapshot of the final field test agent.

Figure 7: The 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 complementary set of agent tactics and preferences 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 complexity associated with specifying team strategy, all of our effort in this 72 hour period went into the first scenario. After the field test, comparable initial plans were developed for the 2nd and 3rd scenarios, and used to run repeated simulations 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 1).

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 1: Field test scenario results

.

Future Directions

Our experience in applying the cMatrix scheduling system to the disaster response scenarios of the field exercise has pointed up two general areas for future research:

Steering of multi-agent systems - As indicated above, one key to effective performance was an ability to provide the team of agents with an overall strategy for approaching the scenario at hand. This strategy was encoded in part in the initial schedule that is generated a priori and distributed to the agents (e.g., restore the main substations first and then converge on the clinic site), and in part by a set of task priorities and preferences that associate primary and backup roles to individual agents. However, the mechanisms that were developed for specifying strategy were restrictive in a couple of important respects. First, they required strategy to be "programmed" in very low level terms, requiring complex sets of individual agent priorities and preferences for specifying simple high level concepts like specifying that two agents travel and work together as a sub-team. The refinement of strategic guidance for a given scenario was consequently a cumbersome and lengthy process of generate and simulate cycles. To streamline this process, we are currently investigating the definition of a higher-level strategy specification language, and the development of techniques for mapping such specifications to the sorts of primitives described in this paper.

A second restriction of the current approach to steering is that provision is only made for specifying guidance prior to execution. In fact, unexpected execution events can invalidate guidance. Consider the unexpected discovery of a "bridge out" event (explicitly excluded from the field exercise scenarios). If the bridge is a strategic link in the transportation network, then previously specified agent itineraries for visiting various survey sites might now be quite suboptimal (or even impossible). The development of mechanisms for detecting when strategic guidance should be reassessed and for interactively adjusting guidance during execution is a second area of current focus.

Partial Centralization - Another general obstacle to optimized multi-agent behavior in the disaster response domain was our basic dependence on peer-to-peer coordination strategies. This was due in large part to the mandate of the Coordinators program to understand what level of coordination is possible without maintaining a global view. However, in domains like disaster response, there is no compelling reason to make this strong assumption. For example, in the fairly recently initiated RoboCup Rescue Agent Simulation competition[4], which bears striking similarity to the disaster response field exercise in many respects, agents are organized into organizational structures (e.g., police department, fire department), and there are natural opportunities for centralizing various task allocation decisions to better optimize system behavior. We are interested generally in extending our scheduling and coordination mechanisms to better exploit organizational structure.

References

Barbulescu, L.; Rubinstein, Z. B.; Smith, S. F.; and Zimmerman, T. L. 2010. Distributed Coordination of Mobile Agent Teams: The Advantage of Planning Ahead. In van der Hoek; Kaminka; Lesperance; Luck; and Sen., eds., *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*.

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

Cesta, A., and Oddi, A. 1996. Gaining efficiency and flexibility in the simple temporal problem. In *Proc. 3rd Int. Workshop on Temporal Representation and Reasoning*.

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

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

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

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

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

[4] http://www.robocuprescue.org/agentsim.html