Human-Agent Collaborative Optimization of Real-Time Distributed Dynamic Multi-Agent Coordination *

Rajiv T. Maheswaran
Univ. of Southern California
Information Sciences Institute
4676 Admiralty Way #1001
Marina Del Rey, CA, USA
maheswar@isi.edu

Craig M. Rogers
Univ. of Southern California
Information Sciences Institute
4676 Admiralty Way #1001
Marina Del Rey, CA, USA
rogers@isi.edu

Romeo Sanchez
Univ. of Southern California
Information Sciences Institute
4676 Admiralty Way #1001
Marina Del Rey, CA, USA
rsanchez@isi.edu

Pedro Szekely
Univ. of Southern California
Information Sciences Institute
Marina Del Rey, CA, USA
pszekely@isi.edu

ABSTRACT

Creating decision support systems to help people coordinate in the real world is difficult because it requires simultaneously addressing planning, scheduling, uncertainty and distribution. Generic AI approaches produce inadequate solutions because they cannot leverage the structure of domains and the intuition that end-users have for solving particular problem instances. We present a general approach where end-users can encode their intuition as guidance enabling the system to decompose large distributed problems into simpler problems that can be solved by traditional centralized AI techniques. Evaluations in field exercises with real users show that teams assisted by our multi-agent decision-support system outperform teams coordinating using radios.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

Real-Time Dynamic Planning and Scheduling, Human-Agent Interaction, Human Guidance, Decision Support, Multi-Agent, Uncertainty, Dynamism, Coordination

1. INTRODUCTION

Teams of people need to coordinate in real-time in many dynamic and uncertain domains. Examples include disaster rescue, hospital triage, and military operations. It is possible to develop plan *a priori*, but many parts of these plans must be left unspecified because people won't know exactly what needs to be done until they are executing the plan in the field. Additionally, requirements and tasks can evolve during execution. Our work addresses a fundamental multi-agent systems endeavor of creating decision support systems that help humans perform better in these domains. The technical challenges to compute good solutions for these problems have been well documented [10, 7, 3].

Established approaches address subsets of the problem, but none have adequately addressed the full problem. Classical planning techniques can barely compute the sets of actions that each person should perform for large problems involving metric resources and cannot cope at all with uncertainty and distribution. Decision-theoretic planning addresses uncertainty, but performance degrades with increased distribution and scale. Distributed constraint optimization techniques address distribution, but do not address temporal reasoning, uncertainty or scale. In practice, it is possible to address specific domains with custom algorithms that use powerful heuristics to leverage the structures unique to that domain. These solutions are expensive to create as even these domains involve planning, uncertainty and distribution. The goal remains to develop generic approaches that produce good solutions that help human teams in many domains.

We introduce a new approach, STaC, based on the premise that people have good intuitions about how to solve problems in each domain. The idea is to enable users to encode their intuition as guidance for the system and to use this guidance to vastly simplify the problems that the system needs to address. The approach is related to heuristic planning, but differs in two important aspects. First, the goal is to capture intuition about solving specific instances of the problem rather than providing heuristics that apply to many instances in the domain. End-users rather than domain experts or developers encode heuristics for the system. Second, in STaC, the intuition is not captured by rules of what actions to take in specific situations, but rather as a decomposition of the problem into simpler problems that can be solved independently.

The model is defined by software developers, and declares the capabilities agents possess and the capabilities that relevant actions in the domain require in order to be performed. These capabilities define the vocabulary for users to express the guidance that encodes their intuition about how to solve a particular problem in-

*The work presented here is funded by the DARPA COORDINATORS Program under contract FA8750-05-C-0032. The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them. Approved for Public Release, Distribution Unlimited.

Figure 1: Field Exercise Images from Rome, NY

stance. The key to STaC is using the model and guidance to produce sufficiently smaller task structures that can be centralized so that a single agent can determine who does what, when and where with respect to these significantly simpler task structures. This mitigates the distribution challenge and enables using auxiliary solvers based on established AI techniques which produce good solutions at a smaller scale. These smaller task structures are solved independently assuming that the human guidance has addressed any significant dependencies. STaC addresses tracking the dynamism in these task structures, the transitioning of agents assignment between these smaller task structures and the invocation of auxiliary solvers. Given that the task structures are treated independently and sufficiently small to be centralized, we call them sandbox reasoners. The sandbox reasoners required in each domain are different, so custom code must be written for each domain. However, the benefit of the approach is that sandbox reasoners are significantly simpler than the custom solvers required to produce a custom solution for a domain.

The rest of the paper is organized as follows. The next sections introduces the real-world domain where our approach was tested followed by related work. We then describe the details of the STaC approach and the particular sandbox reasoners used in our example domain. We close with evaluation results, conclusions and directions for future work.

2. FIELD EXERCISES

The field exercises were based on a simulated disaster rescue domain. The first two exercises were held in the city of Rome, NY, and the second three were in Stanton Wood Park in Herndon, VA. Images of the field exercise in Rome, NY are shown in Figure 1 and a map of the sites and road network of Stanton Wood Park are shown in Figure 2. They were organized and evaluated by independent parties contracted by the DARPA Coordinators program. The rules of the field exercise were created collaboratively by the teams building coordinator agents, the independent evaluation team, and subject matter experts. The specific instances or *scenarios* that comprised the test problems were chosen by the independent evaluation team.

Various locations were selected as *sites* and a feasible road network was constructed. If the site was *populated*, it could have injured people in either *critical* and *serious* condition. Populated sites would also have gas, power and water substations which may have been damaged. In addition, any site could have *facilities* such as a *hospital*, *clinic*, *warehouse*, *gas main station*, *power main station* and *water main station*. A team would obtain points by rescuing injured to hospitals or operational clinics (before a deadline associated with each injured person) and by repairing main stations and substations. The goal of a scenario was to accumulate as many points as possible before the scenario deadline.

The teams were composed of 8 field agents and 2 command agents. Each agent had a different set of skills. Three *specialists* in *gas*, *power* and *water* could perform *major* and *minor* repairs in their respective skill area. The *medical specialist* could load any type of injured person by themselves. The remaining four *survey specialists* could have any collection of skills involving minor repairs. The field agents could move throughout the field exercise area and perform actions. The command agents were located at a base where they helped to coordinate the activities of the team. The *Radio Team* communicated only with radios. Our *CSC Team* had ruggedized tablet computers on which our agents were loaded, in addition to radios. The tablets had cell modems and GPS.

Many outcomes were revealed during the game for which little or no likelihood information was given *a priori*, i.e., no probability distribution functions over outcomes. Teams did know the space of possible outcomes beforehand. A *survey for damage* at a main station or substation revealed the number and type of problems chosen from a set of known possible problems. A *survey for injured* at a populated site revealed the number, types and deadlines for the injured at that site. As the result of a survey, any team member might be injured, forcing them to go to an operational medical facility to recover before proceeding with any other action. A survey could also reveal that the vehicle of the agent doing the survey had failed and would require a vehicle repair before the agent could travel to any other site. While traveling, agents could encounter*road blocks* which could not be passed until fixed. Travel and repair times could vary and repairs could fail. These dynamic and uncertain events were planned parts of the exercise. In addition, the teams had to address uncertainties inherent in the environment, such as noisy radios, weather, and other activities in the public settings. Furthermore, most of these outcomes were only observable by the agent encountering the outcome.

The independent evaluation team chose the scenario from the space of possible exercises and informed the teams of the details below one day prior to the test: (1) the locations of populated sites and facilities, (2) the road network and ranges on travel times between sites, (3) a range for the total number of injured at each site, (4) the points for rescuing each type of injured, which could vary by type and site, (5) the points for repairing each substation or main station, which could vary by type and site, (6) potential problems after surveys for damage and corresponding repair options, (7) ranges on repair times, (8) likelihoods of failure for every repair activity, and (9) the skills of the survey specialist agents. The deadlines (for the scenario and injured) did not allow teams to do all possible repairs and rescues. The teams had one day to form a high-level strategy. The only element of uncertainty which could be modeled accurately with a probability density function was (8). When a team member completed a repair activity, they would call the evaluation team, which would report whether the repair was successful or a failure. The range in (3) was respected by the scenario designers, i.e., the number of injured did not fall outside the given range.

There were many rules and couplings that forced agents to coordinate. To do surveys, gas and power substations at the site had to be off, which required agents with those skills. Two agents had to

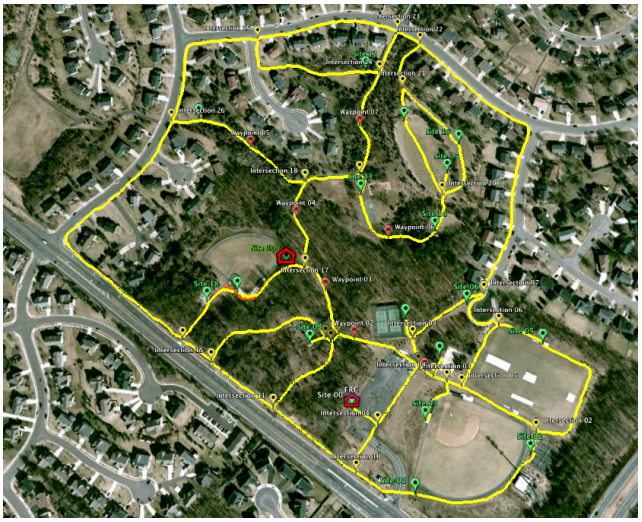

Figure 2: Stanton Woods Park, Herndon, VA

be at the same location simultaneously to load a critically injured person or repair a road block. Repair options could involve multiple tasks and require two agents with certain skills to act in synchrony or in a particular sequence. Some repair options required kits which guaranteed their success, but kits were available only at warehouses. Agents could transport at most one entity, i.e, either a repair kit or a single casualty. A substation was considered repaired only if the corresponding main station was also repaired. A clinic was not operational until all substations at the site and all corresponding main stations were repaired. These are examples of rules that, along with the dynamism and uncertainty in outcomes mentioned earlier, created challenging real-time real-world distributed coordination problems.

The goal was to see if humans operating with radios and a multi-agent decision-support system could outperform humans operating with only radios. Although the field exercises still abstracted some aspects of a real-world disaster scenario, we believe they closely approximated the challenges of helping a human team solve difficult real-world problems.

3. RELATED WORK

The STaC framework was developed during the DARPA Coordinators program. In the first two years, DARPA ran competitive evaluations on simulated scenarios, and CSC, the underlying system behind the STaC framework, won such evaluations by considerable margins against two competing approaches: an MDP-based approach [11] and an STN framework [14].

The MDP-based [11] approach addressed the infeasibility of reasoning over the joint state space by setting the circumstance set to a subset of local state space that is reachable from the current local state, unrolling the state space by doing a greedy estimation of boundary values. It biased its local reward function on the commitments made by the agents during execution. However, such approximations lose critical information, exploring state spaces that are far from good distributed solutions.

The STN framework [14] addressed temporal uncertainty by using a time interval (instead of a point) as the circumstance that denoted feasible start times for a method to be executed. The system used *constraint propagation* to update the start intervals of the agents' activities during execution. A policy modification phase was triggered if execution was forced outside the given set of intervals. One of the problems of this approach is that agents tried to maintain consistency and optimize their local schedules, losing information that was needed to timely trigger policy modifications for their schedules.

We encoded scenarios of the field exercise as planning problems using PDDL [5]. The motivation was to identify to the extent to which current automated planning technology can address complex distributed, resource-driven, and uncertain domains. Unfortunately, this proved to be extremely difficult for state-of-the-art planning systems. From the set of planning systems tried, only LPG-TD [6], and SGPLAN [4] solved a few simplified problems, after uncertainty, dynamism, non-determinism, resource-metrics, partial observability and deadlines were removed. Planners were unable to scale to more than 5 sites. LPG-TD produced solutions more efficiently but less optimally.

In general, mixed-initiative approaches where humans and software collaborate can often produce better solutions for complex problems. Mixed-initiative planning systems have been developed where users and software interact to construct plans. Users manipulate plan activities by removing or adding them during execution while minimizing the changes from a reference schedule [1, 8, 12]. However, most of these systems are centralized, so humans and systems are fully aware of the entire plan, and of the consequences of updating it. In our scenario, agents (including humans) have subjective views of the world, and any decision may trigger many unknown global effects.

Multi-agent systems for disaster domains have been studied in the context of adjustable autonomy. The idea is to improve limited human situational awareness that reduces human effectiveness in directing agent teams by providing the flexibility to allow for multiple strategies to be applied. A software prototype, DEFACTO, was presented and tested on a simulated environment under some simplifications (e.g., no bandwidth limitations, reliable communications, omnipresence) [13]. Our work also recognizes the importance of flexible frameworks to allow better human-agent interactions. The test-bed presented in this paper does not make any major simplifications, being a first step toward creating multi-agent systems for real-world problems.

4. THE STaC APPROACH

Our goal is to create a general framework for incorporating human strategic guidance. We introduce the formalism for STaC guidance and give an example from our domain. We then describe how this guidance is executed with the use of Total Capability Requirement (TCR) sets. We provide an example of a TCR set and discuss how dynamic updates enable execution of the guidance.

4.1 STaC Guidance

We make the following assumptions about a general multi-agent coordination problem. There are a set of agents N and a set of actions A. Agents have capabilities from a set of capabilities: $\Theta_n \in \Theta$. Each action is mapped to a capability, i.e., $\gamma : A \rightarrow \Theta$. An agent can perform any action for which it has the capability.

The problem is composed of a collection of tasks T. Each task $t \in T$ is associated with a set of potential actions involved in completing it: $A^t \subset A$. It is not necessary that $\{A^t\}$ be disjoint. Fur-

thermore, for the purposes of guidance, it is not relevant how these tasks relate to the actual reward function. It is only important that the notion of tasks exists.

We can define a generic representation for human strategic guidance as follows. Guidance is an ordered set of *guidance groups*: $G = \{G_i\}$. Each guidance group G_i is associated with a *subteam* of agents $S_i \subset N$ and an ordered set of guidance elements E_i. Each *guidance element* $e_i^j \in E_i$ is composed of a task $t_i^j \in T_i$, a set of *constraints* C_i^j, and a temporal bound b_i^j. The constraints C_i^j are a collection of capability-number pairs $\{(\theta, n_\theta)\}$ where $\theta \in \Theta$ and $n_\theta \in \mathbb{Z}^*$ is a non-negative integer. The pair (θ, n_θ) indicates that each agent in the subteam can use the capability θ at most n_θ times for the task in the guidance element. The temporal bound $b_i^j \in \{0\} \cup \{<, >\} \times \mathbb{R}^+$ is another constraint that can indicate that the guidance element is only valid if the time remaining is greater or less than some number ($b_i^j = 0$ indicates no temporal constraint). Thus,

$$G = \{G_i\} = \{(S_i, E_i)\} = \{(S_i, \{(t_i^j, C_i^j, b_i^j)\})\}$$
$$= \{(S_i, \{(t_i^j, \{(\theta_i^{j,k}, n_{\theta_i^{j,k}})\}, b_i^j)\})\}.$$

We refer to this as the STaC (Subteam-Task-Constraints) formalism for strategic guidance. One can now define a strategy composed of a sequence of subteams, each responsible for a collection of tasks, each of which are to be performed under some constraints. We note that since agents will traverse the elements of this guidance in order, STaCs are actually queues.

4.1.1 Field Exercise Example

We first defined a set of capabilities that were relevant to the field exercise. We also associated each capability with several capability classes for more compact expression of constraints. Below are the set of capabilities and associated classes for actions involving gas and injured, respectively. Capabilities and classes for power and water are analogous to those for gas.

```
gas_major: gas, gas_main
gas_minor: gas, gas_main
survey_gas_main: gas, gas_main, survey
survey_gas_sub: gas, survey
turn_off_gas_main: gas, gas_main, turnoffs
turn_off_gas_sub: gas, turnoffs
pickup_gas_kit: gas, pickup
dropoff_gas_kit: gas, dropoff
```

```
load_critical: critical, injured
assist_load_critical: critical, injured
survey_injured: injured, survey
generic: injured
```

Consider the STaC guidance fragment below. We see an ordered set of guidance groups, each with a subteam of agents and an ordered set of guidance elements. The `only()` operator sets the capability-number pairs for all capabilities not in the argument to zero. The `no()` operator sets the capability-number pairs for all capabilities in the argument to zero. The intent of this plan fragment is for the survey specialist to turn off services at the substations at Site 4 and Site 3, enabling other agents to work there. The gas and survey specialists go to the warehouse at Site 6, pick up gas kits, then restore the gas main station and gas substation at Site 1. The medical specialist and survey specialist are responsible for making sure they each rescue two critically injured people before rescuing all others. The gas and power specialist are then responsible for doing everything except water-related actions at Site 4, but if less than 10 minutes are left in the scenario, they switch to rescuing injured.

```
survey_specialist_1:
  ( task_site_04, [ only( turnoffs ) ], 0);
  ( task_site_03, [ only( turnoffs ) ], 0);
gas_specialist, survey_specialist_1:
  ( task_site_06, [ only( pickup_gas_kit ) ], 0),
  ( task_site_01, [ only( gas ) ], 0);
survey_specialist_1, medical_specialist:
  ( task_site_04, [ (load_critical, 2) ],0),
  ( task_site_04, [ only( injured ) ],0);
gas_specialist, power_specialist:
  ( task_site_03, [ no(water) ], >10),
  ( task_site_03, [ only( injured ) ], 0);
```

Here, the tasks chosen for each guidance element are all those associated with a particular site. This is not a requirement in the guidance formalism. For example, the second guidance group could have also been:

```
gas_specialist, survey_specialist_1:
  ( task_site_06, [ only( pickup_gas_kit ) ], 0),
  ( task_gas_main, [ ], 0),
  ( task_gas_substation_site_01, [ ], 0);
```

This would have specified a fixed ordering between repairing the main station and the substation which did not exist in the original. The expression of guidance is not necessarily unique and can be tailored to the intuition and structure that the designer finds most appropriate.

4.2 STaC Execution

While STaC guidance is compact and has intuitive meaning for a human, the agents have no semantic awareness of what it signifies beyond identifying tasks and limiting actions. This is due to the generality of the formalism. Furthermore, the guidance does not specify which actions to perform, which agents should perform them, the timing of these actions or how to react to any dynamism and uncertainty during execution. We address those challenges here.

4.2.1 Total Capability Requirement (TCR) Sets

Given the STaC formalism, one of the key decisions that every agent must make is when to transition from one task to another. A simple solution is to wait until the current guidance element is completed and then move to the next guidance element (which may involve going to the next relevant guidance group). This approach would lead to significant waste if the agent were unable to contribute to the current guidance element.

Consider the example shown in Section 4.1.1. If the gas specialist arrives at Site 1 first and discovers that all the repair options for the gas main station and gas substation can be completed by the gas specialist alone, or that there exists repair options for both the main station and the substation that can be performed by the gas specialist alone and are guaranteed to succeed, the gas capabilities of the survey specialist are not needed. It may make sense for the survey specialist to skip Site 1 and head to Site 4 to help the medical

specialist rescue injured, even though the repairs at Site 1 have not been completed. It is important to determine dynamically whether the capabilities of each agent in the subteam are needed for the task being executed in the guidance element.

Total Capability Requirement (TCR) sets are a mechanism to achieve this. For every task $t \in T$, there is an associated TCR set $R^t = \{R_i^t\}$, which is a set of *requirements*. Each requirement $R_i^t = (n_i^t, Q_i^t)$ is a tuple of a *requirement number* n_i^t and *requirement type* Q_i^t. A requirement type $Q_i^t = \{q_{i,j}^t\}$ is a collection of *requirement elements*, where each requirement element is a tuple $q_{i,j}^t = (c_{i,j}^t, l_{i,j}^t, n_{i,j}^t)$ where $c_{i,j}^t$ is a capability, $l_{i,j}^t$ is a location, and $n_{i,j}^t$ is an element number. Thus, $R^t = \{R_i^t\} = \{(n_i^t, Q_i^t)\} = \{(n_i^t, \{q_{i,j}^t\})\} = \{(n_i^t, \{(c_{i,j}^t, l_{i,j}^t, n_{i,j}^t)\})\}$.

```
2:[(gas_minor, site_01, 1)]
1:[(gas_minor, site_01, 2)]
4:[(assist_load_critical, site_01, 2)]
1:[(power_minor, site_01, 1) (power_minor, site_03, 1)]
```

Consider the example above which is a possible TCR set for `task_site_01`. This indicates that there are two instances of the need for a single agent with gas minor capability, one instance of a need for two agents with gas minor capability, four instances of a need for two agents capable of loading a critically injured person and one instance of a need for having an agent with power minor capability at Site 1 at the same time that there is an agent with power minor capability at Site 3. The first requirement could occur because the gas main station has two problems, each of which could be solved with a gas minor repair. The second requirement could occur because the gas substation has one problem that requires two agents with gas minor skills to perform a synchronized repair. The third requirement could be due to the discovery of four critically injured people. The fourth requirement represents the need for *remote synchronization*: the need for two agents at two different locations at the same time. In the field exercise, some power substations required an agent at the substation and another at the main station simultaneously to turn the power substation on.

If the guidance element was:

```
( task_site_01, [ only( gas ) ], 0 )
```

then only the first two requirements involving the gas minor capability would be considered when deciding whether an agent should remain committed or released from the task. The TCR sets are dynamically updated such that once a skill is no longer needed, as repairs are completed or injured are loaded, the appropriate requirements are decremented or deleted.

4.2.2 *Calculating TCR Sets*

Our calculation of TCR sets can best be described in the context of our modeling specification for the field exercise scenarios. We used a hierarchical task network structure that was an extension of CTAEMS [2], which is itself a variant of TAEMS [9] developed for the DARPA Coordinators Phase 2 evaluation. The essential property was that tasks (including the root task which represented the overall reward function) were composed of subtasks iteratively until reaching a primitive task which was composed of actions. Tasks could also have non-hereditary relationships such as enablement and disablement. Every task was also associated with state aggregation functions that determined how the state of its subtasks (or child actions) affected the state of the task. An example of a template used to model power substations is shown in Figure 3. This

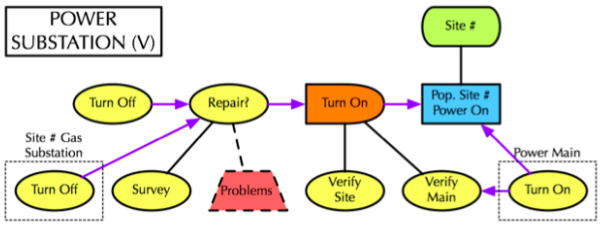

Figure 3: Model Template for Power Substation

also illustrates the issue of dynamism as the task node for *Problems* must remain without children until the power substation is surveyed for damage. Then, the appropriate problems and repair options are added dynamically to the model. It would be cumbersome and practically infeasible to express every possible combination of problems and repair options that could occur. The issues are similar when it comes to modeling the discovery of injured people.

The TCR set for a given task is calculated by applying a TCR aggregation function, chosen based on the state aggregation function associated with the task, to the TCR sets of its subtasks and enabling tasks. For example, a *sum* state aggregation function would correspond to a *union* TCR aggregation function, and a *sync* state aggregation function would correspond to a *cross-product* TCR aggregation function. Thus, TCR sets would start from actions, which are each associated capability and flow forward and up through enablement and ancestral links to form TCR sets for every task. These sets can be dynamically updated as tasks change states.

For example, once a task is completed, the TCR set can be set to null indicating that it does not require any more capabilities. This makes the TCR sets vanish as tasks are completed, allowing agents to be released as soon as possible. In order to address the dynamic nature of the model, tasks that might be expanded during execution must be marked with TCR sets that indicate reasonable upper bounds on needed capabilities. These sets are then changed to the actual TCR sets once outcomes has been observed in the environment. Having an HTN-based model helps to construct and manage TCR sets, but is not necessary. As long as there exists a non-cyclic mapping that describes the relationships of tasks to other tasks and actions, a dynamic methodology to assign TCR sets to tasks can be constructed.

4.3 Partial Centralization

STaC execution can be implemented such that a single agent is responsible for choosing all actions involved with a single task-constraint tuple of a guidance element. We create a mapping, $\omega : T \rightarrow N$, where every task has an owner. The task owner contacts agents who are responsible for related tasks and actions to subscribe to relevant state updates. When an agent reaches a particular task-constraint tuple, it cedes autonomy to the owner of that task until the task owner releases the agent from that commitment. The owner agent keeps track of the set of capabilities of all agents bound to that task as well as the TCR set of that tasks and repeatedly solves an optimization problem to find the best set of agents to keep for the current TCR set. If the solution is a set that is a strict subset of the bound agents, it can release the other agents. Our optimization problem minimized a weighted combination of the number of agents kept and their capabilities. The key insight here is that partial centralization of autonomy always occurs implicitly and thus,

it is beneficial to align the metric for partial centralization with the properties of the domain where it matters.

5. SANDBOX REASONING

Once the task owner has chosen which set of agents to keep, it must then also decide, subject to the constraints in the guidance, which actions to perform and which agents should perform to accomplish the task. We call this process *sandbox reasoning* because the task owner's deliberation over what to do for a single task-constraint tuple is isolated from all actions and tasks that are not related to the task at hand. The task owner does not need to consider impact on the future or on concurrently executing tasks. It is given a collection of agents and autonomy to use them however it sees fit to accomplish the task as well as possible. The consequences of the interactions have, in principle, been considered and addressed by human strategic guidance.

In creating the agent model for a field exercise scenario, we instantiated structure (sometimes, dynamically during execution) from a small set of templates. Examples include *power substation restoration* (as shown in Figure 3) or *critically injured rescue*. Similarly, the tasks in the guidance were also be a subset of tasks that make intuitive sense to the strategy designer. In our case, the task types involved in guidance were far fewer than the templates involved in model generation. We believe that the notion of using templates for generation and more significantly for guidance is a general principle that is applicable to many domains. This belief was also reflected in the DARPA Coordinators program and its Phase 2 evaluation. The main templates needed for guidance were a repair and rescue manager. We discuss the automated reasoners in these managers below.

5.1 Repair Manager

The repair manager would take as input (1) a collection of facilities that had been damaged, (2) a set of problems for each facility, (3) a set of repair options for each problem, (4) set of agents with (5) associated capabilities and (6) times that they would be available to be scheduled for activities, The output would be a policy that yielded a collection of agent-action tuples given a simplified version of the state. While this may seem field-exercise specific, this reasoner had no semantic knowledge of the field exercise.

The problem was generalized as follows: Given a set of tasks $\{T_i\}$, where each task is a conjunction of a set of problems, $T_i = \min(\{P_j\})$, each problem is a disjunction of repairs, $P_j = \max(\{R_k\})$, and each repair is a function of actions, $R_k = \odot(\{a_l\})$ where $\odot \in \{sync, sequence, min\}$ is a collection of operators that require the elements to have synchronized start times, sequential execution, or conjunctive success in order for the repair to succeed, respectively. Each action is associated with a capability, expected duration, and probability of failure. We also have a set of agents where each have an associated set of capabilities and an availability time. This is a straightforward optimization given an appropriate objective function.

We needed a fast solution (less than five seconds) because users needed guidance from the solver after performing a survey. Our solution to was to build a policy based on a simplified representation of state. The state vector is indexed by all possible actions and takes values from the set $\{NotStarted, Succeeded, Failed\}$. The policy output given for a state is a set of agent-action pairs.

The policy is constructed by running simulation traces of the optimization problem. At every time step in the simulation, if there are idle agents and no action-agent tuples for the current state in the policy, the agents are randomly assigned to an action they can perform and marked busy for the expected duration. These assignments are then associated with the state of the simulation where executing actions are interpreted to have succeeded. The outcomes of the actions in the simulation are determined using the given probability of failure. Multiple simulation runs are used to build a single policy which receives a score based on the average idle time of all agents. Multiple policies are generated using the same mechanism and the policy with the best score is stored. To execute the policy, the task owner maps the current state of the actions to the policy state by mapping executing actions to *Succeeded* and schedule the associated action-agent tuples as the next action for the agent. If there is no agent-action tuple for the translated state or if any of the input parameters (e.g. availability times, new tasks) change, policy generation restarts from scratch. While this is a simple stochastic sampling approach, it produces reasonable policies with very limited computation requirements. Policy generation was typically bounded to five seconds. Also, while the potential state space is exponential in the number of actions, typical policies had at most about 200 states.

The key idea is that we could create a sandbox reasoner that can solve a generic context-independent problem which could be applied to many tasks in the guidance. One could, in theory, use an MDP or STN-based approach if it yielded a solution within the limits of bounded rationality in the domain at hand.

5.2 Rescue Manager

The rescue manager would similarly take as input a list of agents and a set of injured with associated types and deadlines, and output a set of agent-action pairs when agents became idle. We used a simple reactive planner with a handful of simple heuristics to determine when to wait for an agent to help load a critically injured (which requires two agents to be present simultaneously) and when to take a serious (which could be done by one agent). This also can be formulated as a generic problem consisting of a set of tasks with associated deadlines and durations and the tasks can be either a singleton action or require synchronized actions by two different agents. The rules were variations of: "If an agent is on the way and it will arrive later than the duration of the singleton action, perform the singleton action and return to the site, otherwise, wait for the agent." The variations were due to the constraints placed on the rescue task and the number of agents available to do the rescues.

The general philosophy of the STaC approach to guidance, its execution and sandbox reasoning is to create a generic framework for human strategic input to decompose a very difficult problem into smaller problems that can be solved in isolation with automated tools created to solve large classes of task structures that appear in the guidance. Our system was completely unaware of any semantics of the field exercise, and a similar approach could be used in a completely different domain.

6. EVALUATION

Figure 4 shows the scores for the three scenarios run in Herndon. For each scenario, the top, lighter bar shows the radio-team score, and the bottom, darker bar shows the score of the team using the CSC system described in this paper. The middle bar shows the results of a simulation of the scenario using a baseline version of the system with simple sandbox reasoners. In order to calibrate the simulation scores, we also ran a simulation of the complete

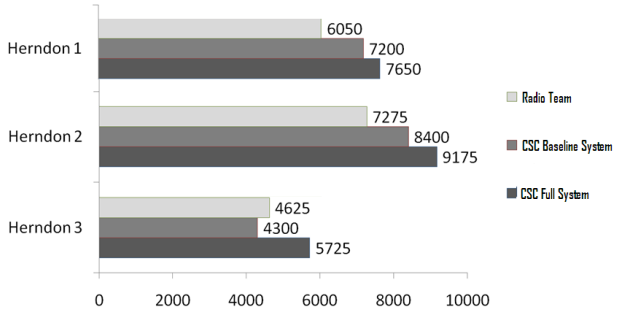

Figure 4: Herndon Evaluation

version of the system using the same dice rolls used in the physical scenario. The simulation results for the full system were within 200 points of the results obtained in the field, which suggests that if the baseline system had been used in the field, the results would also be close to those shown in the figure.

In the baseline version, the repair manager uses a random strategy where agents randomly select a repair task to perform from the set of tasks that the agent is eligible to perform according to its capabilities. The baseline rescue manager uses a greedy strategy where agents select the injured with the earliest deadline that lives long enough to arrive to the medical facility before the deadline. In the baseline rescue manager, agents don't wait for a partner that enables loading a critically injured that they would not be eligible to load otherwise.

The results show that more sophisticated sandbox reasoners always resulted in better scores: 5.8%, 8.4% and 24.8% improvements. The differences in scenarios 1 and 2 were small, and in those scenarios the baseline system also outperformed the radio team. In scenario 3, the difference is more significant. This scenario emphasized injured, and the greedy strategy used in the simple version of the system delayed rescuing the critically injured. Agents rescue seriously injured with later deadlines instead of waiting for a partner to rescue a more valuable critically injured with an earlier deadline.

Figure 5 shows simulation results that compare the effects of alternative strategies. We organized these strategies along two dimensions: the number of clinics that would be made operational (0, 1 and 2), and the number of independent subteams (1, 2 and 4). In the strategies with 1 clinic, the team repaired the clinic that was considered most useful (closest to the most valuable injured). In the scenarios with 2 and 4 teams, we specified the teams so that they could perform repairs independently. In the strategies with 0 clinics, the teams performed no repairs and rescued injured to a medical facility that was always operational and required no repairs. In the strategies with 1 and 2 clinics, the agents first repair the main stations, then the clinics and then visit the remaining sites to rescue all injured and perform all repairs according to the following algorithm. First, the sites are ordered according to the total expected number of points achievable at the site. The teams take turns picking the next most valuable site from the ordered list until the list is exhausted. The idea is to complete the most valuable sites first so that when time runs out the most valuable sites have been completed.

Figure 5 shows that in the Herndon 1 and 2 scenarios, the strategies that repair 1 or 2 clinics are competitive with the radio team, outscoring them in 11 out of the 12 strategies involved. However, in all three scenarios, the CSC strategy used in the field was significantly better than all the alternative strategies. The difference is due mainly to the use of constraints. In the alternative strategies, the agents performed all tasks at a site, whereas the strategies used in the field used constraints to prevent agents from performing tasks that we deemed not worthwhile. In addition, we used constraints on the number of injured rescued to prevent agents from rescuing all injured at a site before moving to the next site. Instead, we used longer itineraries that visited sites multiple times in a round-robin, so agents would rescue the most urgent injured first.

7. CONCLUSIONS AND FUTURE WORK

Our 18-month experience working on a system to compete against radio teams in the field exercises provided significant evidence for the benefits of our approach. Our starting point was our generic CSC system developed during the previous two years to solve generic, synthetically generated problem instances specified in CTAEMS. Even though the synthetically generated problem instances were generated according to templates that combined "typical" coordination situations, the resulting problems were not understandable by humans. In contrast, the field exercise problems are natural, and appeal to our lifetime of experience coordinating every day activities. Intuitions about space, distance, time, importance and risk all came into play, enabling teams of humans to devise a sophisticated strategy within one hour of brainstorming. It became obvious early on that the generic CSC system would not be able to produce solutions comparable to the desired sophisticated, coordinated behavior of human-produced strategies.

Our existing system had performed extremely well in Phase 2 by using our Predictability and Criticality Metrics (PCM) approach. In the PCM approach, the policy modifications that agents consider are limited to those that can be evaluated accurately through criticality metrics that capture global information. These policy modifications were simple and thus the reasoners that implemented them were simple too.

For the field exercises, we extended our approach so that policy modifications would be constrained using the guidance provided by the users. This guidance was in the form of a sequence of sites to visit. The system was left to make decisions that we believed it could evaluate accurately (e.g., how to perform repairs or rescue injured at a single site). The system relied on the TCR set criticality metric to determine how to move agents along the list of guidance elements. The approach worked well. Our users outperformed the radio team because they were able to communicate their strategy to their agents, and the system optimized the execution of the strategy, adapting it to the dynamics of the environment.

The field exercises in Rome, NY used a simpler language for specifying guidance. It had a single guidance group consisting of the entire set of agents. Also, it did not support constraints to control the capabilities within a guidance element. In that evaluation, our system remained competitive with the radio team, but lost in two out of the three scenarios. The final language for guidance was inspired by our observations of the radio-team strategies, extensive discussions with subject matter experts and extensive numbers of simulations. We noted that while the human team could not execute a strategy as well as we could, the space of strategies that they were able to engage were far more sophisticated than ours. This led

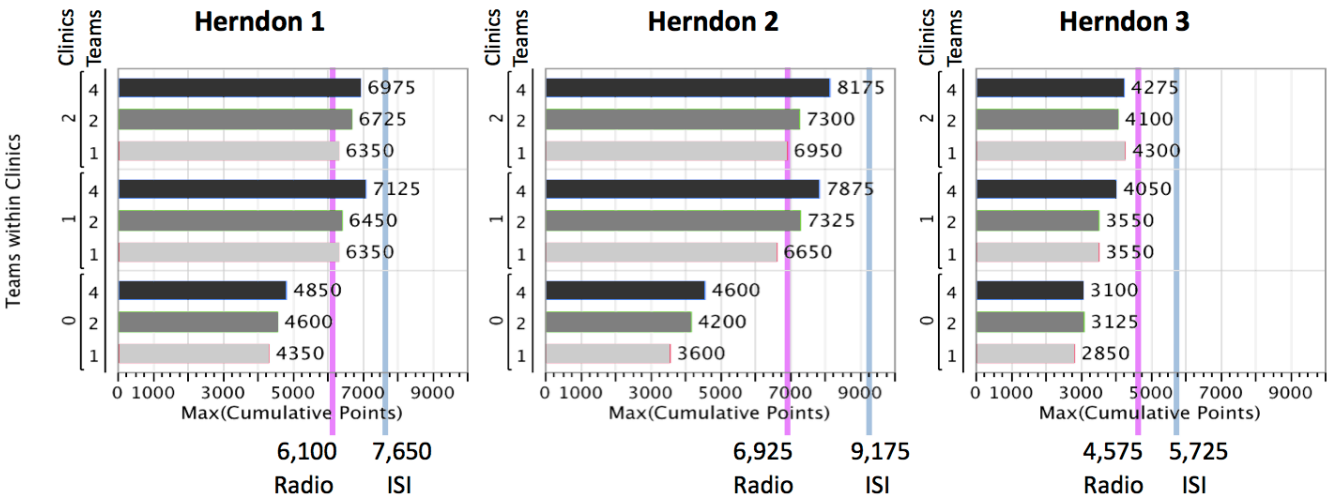

Figure 5: Baselines

to the creation of a the more sophisticated formalism for capturing human strategic guidance.

We have taken the first step towards generic coordination technology that end-users can tailor to specific problem instances. The approach was validated in one domain thanks to the extensive and expensive evaluations carried out by the DARPA Coordinators program. In the future, we hope to be able to apply this approach to other application domains. One key area that needs to be investigated is extensions to allow human users to make guidance adjustments *during* execution. There are situations where a series of outcomes either invalidates an assumption when creating the *a priori* guidance or creates an opportunity to improve on that guidance. Addressing this requires the ability for human users to quickly and easily understand and modify the guidance while it is being executed. Even more advanced steps would be evaluating and ultimately generating appropriate online guidance modifications.

8. REFERENCES

[1] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J. C. jung Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. G. Chafin, W. C. Dias, and P. F. Maldague. Mapgen: Mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems*, 19(1):8–12, 2004.

[2] M. Boddy, B. Horling, J. Phelps, R. P. Goldman, R. Vincent, A. C. Long, B. Kohout, and R. Maheswaran. CTAEMS language specification: Version 2.04, 2007.

[3] C. Boutilier. Multiagent systems: Challenges and opportunities for decision-theoretic planning. *AI Magazine*, 20(4):35–43, 1999.

[4] Y. Chen, B. Wah, and C.-W. Hsu. Temporal planning using subgoal partitioning and resolution in sgplan. *Journal of Artificial Intelligence Research (JAIR)*, 26(1):323–369, 2006.

[5] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research (JAIR)*, 27, 2006.

[6] A. Gerevini, A. Saetti, I. Serina, and P. Toninelli. Fast planning in domains with derived predicates: an approach based on rule-action graphs and local search. In *Proceedings of the 20th national conference on Artificial intelligence (AAAI)*, pages 1157–1162. AAAI Press, 2005.

[7] F. C. Groen, M. T. Spaan, J. R. Kok, and G. Pavlin. *Real World Multi-agent Systems: Information Sharing, Coordination and Planning*, volume 4363/2007 of *Lecture Notes in Computer Science. Logic, Language, and Computation*. Springer-Verlag, 2007.

[8] C. Hayes, A. Larson, and U. Ravinder. Weasel: A mipas system to assist in military planning. In *ICAPS05 MIPAS Workshop (WS3)*, 2005.

[9] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. N. Prasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):87–143, 2004.

[10] R. R. Murphy. Human-robot interaction in rescue robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):138–153, 2004.

[11] D. J. Musliner, E. H. Durfee, J. Wu, D. A. Dolgov, R. P. Goldman, and M. S. Boddy. Coordinated plan management using multiagent MDPs. In *Proceedings of the 2006 AAAI Spring Symposium on Distributed Plan and Schedule Management*, March 2006.

[12] K. L. Myers, P. A. Jarvis, W. Mabry, T. Michael, and J. Wolverton. A mixed-initiative framework for robust plan sketching. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS)*, 2003.

[13] N. Schurr, J. Marecki, P. Scerri, J. Lewis, and M. Tambe. *The DEFACTO System: Coordinating human-agent teams for the future of disaster response*, chapter Programming multi-agent systems: Third international workshop. Springer Press, 2005.

[14] S. Smith, A. T. Gallagher, T. L. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed management of flexible times schedules. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2007)*, Honolulu, HI, May 2007.