Just-in-Time Backfilling in Multi-Agent Scheduling

Anthony Gallagher
anthonyg@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213

Luke Hunsberger
hunsberg@cs.vassar.edu
Vassar College
Poughkeepsie, NY 12604

Stephen F. Smith
sfs@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper addresses the problem of how a group of agents cooperating on a complex plan with interdependent actions can coordinate their scheduling and execution of those actions, particularly in domains where actions may fail or have uncertain durations. If actions fail (or fail to meet their deadlines), the repercussions for the rest of the team's plan can be dramatic. This paper presents a pro-active strategy, called Just-in-Time Backfilling (JIT-BF), that agents can use to increase the fault tolerance of their interdependent schedules by identifying actions in danger of failing and inserting redundant (or back-up) actions into their schedules. The insertion of redundant actions can be done locally (i.e., by the agent whose action is in danger of failing) or through negotiations with the rest of the team. The computations performed by agents following the JIT-BF strategy depend on probabilistic models of action durations and the "quality" achieved by successfully executing actions. The paper presents an experimental evaluation of the JIT-BF strategy within a simulated real-time dynamic environment that demonstrates that teams using the pro-active JIT-BF strategy significantly out-perform teams that rely solely on reactive strategies.

Introduction

The primary focus of this paper is on coordinating the interdependent actions of multiple agents in uncertain execution environments—in particular, in environments in which actions can fail or have uncertain durations. In such environments, agents must make predictive assumptions about actions they are inserting into their schedules. If those assumptions later turn out to have been wrong, the agents must have some way of reconciling the conflicts that arise when contradictory information becomes available at execution time.

In multi-agent scheduling domains, agents frequently face inconsistencies involving temporal beliefs. For example, an agent may believe that one action shall end and enable a second action to begin, only to find later on that the first action overran its deadline and failed. As this example suggests, the root cause of conflicts among temporal beliefs is the uncertainty of action durations. Approaches that deal with execution uncertainly fall into two broad categories: *reactive* and *proactive*. In reactive approaches, agents react to conflicting temporal beliefs as they occur during execution. In contrast, pro-active approaches make use of models of uncertainty to reason about potential conflicts before they arise, and decide what to do in advance. Reactive approaches suffer in circumstances where the execution state does not provide any basis for recovery, while proactive approaches tend to be overly conservative with respect to what can be accomplished in a given time frame.

In earlier work (Smith et al. 2007), we developed an agent system in which agents use *Simple Temporal Networks* (STNs) (Dechter, Meiri, and Pearl 1991) to manage the temporal constraints in their schedules, not only when constructing initial schedules, but also over time as actions are executed and agents learn new information. STNs are useful because they allow agents to maintain flexibility in their schedules. Thus, for example, if an action is originally expected to take 30 time units, but actually ends up taking 40 time units, the STN may be able to absorb the changes without sacrificing consistency.

Using our agent system, we previously presented reactive strategies for maintaining the consistency of temporal beliefs and resolving conflicts when they arise (Gallagher and Smith 2008). Such strategies are fundamental to the use of STNs in a dynamic, multi-agent scheduling environment. In this paper, we build on these results and consider the addition of a pro-active strategy designed to anticipate inconsistency and take advance action to insulate the plan against it. Our approach, called Just-In-Time Backfilling (JIT-BF), aims to minimize the impact of activity failure in quality maximization contexts through the introduction and execution of redundant activities by other agents.

The Problem Domain

We take as our starting point the problem domain used in DARPA's Coordinators Program (Wagner et al. 2004), which uses a variant of the TAEMS (Decker 1996) modeling framework called CTAEMS (Boddy et al. 2005). Below, we give the basic assumptions about temporal constraints, actions and models of uncertainty in this domain.

- Actions are subject to temporal constraints. This paper restricts attention to *simple temporal constraints* each having the form, $t_j - t_i \leq \delta_{ij}$, where t_i and t_j are variables, called *time-points*, that represent the starting or stopping

times of actions, and δ_{ij} is a real number (Dechter, Meiri, and Pearl 1991).

- Primitive actions are executed by individual agents. Agents can execute only one action at a time. Although agents control when actions start, actions can have uncertain durations. This paper presumes that the duration of each action is governed by a given discrete probability distribution. For example, the duration of an action A might be 25 with probability 0.5, 30 with probability 0.3, and 35 with probability 0.2.
- Actions accrue *quality* when successfully executed before their deadlines. However, actions can also be linked by contraints, called *enabling constraints*, such that the successful execution of the target action depends on the successful execution of one or more source actions. In this setting, a plan is a hierarchy of nodes, where leaf nodes correspond to primitive actions and interior nodes correspond to abstract actions. Each interior node accumulates quality according to a *quality-accumulation function* (or QAF), where different interior nodes can have different QAFs associated with them. For example, one interior node might have a *max* QAF, according to which the quality of that node is the maximum of the qualities of all of its child nodes; another interior node might have a *sum* QAF, according to which the quality of that node is the sum of the qualities of all of its child nodes.
- A group of agents receives an initial (potential) plan (i.e., action hierarchy) and their goal is to maximize the quality of that plan (i.e., to maximize the quality of the root node in that hierarchy). Given that the initial plan typically contains more actions than the group of agents can hope to execute, they must decide which subset of actions to attempt to execute. That subset of actions then populates their initial schedules.
- To provide flexibility to the agents, the initial (potential) plan contains redundant actions. For example, if an interior node has a *max* QAF, then it can accrue positive quality even if only one of its child nodes achieves positive quality. In this case, the rest of the nodes could be considered redundant (i.e., there is no harm in executing more than one child; but only the highest quality child that gets executed contributes quality to the parent node).

Approach

As described earlier, we have developed an agent system based on the use of STNs (Smith et al. 2007). In particular, the agents in our system use STNs to manage the temporal constraints in their schedules. Since STNs do not readily accommodate models of uncertainty, our agents, when inserting actions into their schedules, assume that the durations of those actions will be their *expected* durations according to the probabilistic models. As a result of this deterministic scheduling assumption, agents may be faced with temporal conflicts during execution of those actions.

Since agents cannot typically execute all of the actions in a given plan, our agents select a subset of actions from a given plan which they insert into their individual schedules. New temporal constraints are added to an agent's STN whenever the agent (1) inserts a new action into its schedule, (2) removes an action from its schedule, (3) updates the current time, (4) begins to execute an action, or (5) finishes executing an action. Contraints added to an STN are propagated throughout the network. Elsewhere, we have examined reactive approaches to maintaining the consistency of STNs and repairing them when they become inconsistent (e.g., when an action executes past its deadline) (Gallagher and Smith 2008). That work was based on indentifying and analyzing the *negative cycles* of constraints that characterize STNs containing inconsistencies.

In this paper, we present a complementary pro-active strategy, called the Just-In-Time Backfilling strategy, in which agents increase the fault-tolerance of their schedules by identifying actions that may be in danger of failing and, in advance of their execution, select appropriate redundant (or backup) actions to insert into their schedules. The computations are based on probabilistic models of action durations, as well as estimates of changes in quality.

Each agent follows the JIT-BF strategy by repeatedly—and independently—cycling through the following steps. First, the agent examines its own schedule to identify any soon-to-execute actions that might be in danger of failing. Second, for each such action, the agent attempts to insert redundant actions into its own schedule aiming to minimize adverse impacts to their overall quality. If that attempt fails, the agent can choose to coordinate with other agents to determine whether they might be able to schedule redundant actions. These steps are repeated each time an agent begins the execution of the next action in its schedule.

To bound the computation required by the JIT-BF strategy, the first step restricts attention to *pending actions*—that is, actions that are currently executing or will be executing soon. The redundant actions considered in the second step can be scheduled locally (i.e., by the agent whose action is in danger of failing) or remotely (i..e, by one of the other agents in the group). Crucially, except for the coordination in Step 2, each agent follows the JIT-BF strategy independently, without the need for large-scale multi-agent synchronization. These steps are described in more detail below.

Identifying Actions in Danger of Failing

The *pending actions* include the action that an agent is currently executing, together with a number of actions immediately following that action in the agent's schedule.[1] The number of pending actions considered by the JIT-BF strategy is limited by the following parameters:

- *MaxPendingActions* places an explicit upper bound on the number of pending actions.
- *PendingActionsHorizon* sets a width for the time-window to which the pending actions must belong. If the current time is c, then the time-window is: $[c, c + PendingActionsHorizon]$.

[1] Since the agent identifies the pending actions whenever it starts executing some action, the pending actions always include an action that the agent is currently executing.

For each pending action, A, the agent does the following:

(a) Collect the *predecessors* of A.

(b) Estimate $ProbFail(A)$, the probability that A (or one of its successors) will fail due to one or more of A's predecessors finishing late. Any action A for which $ProbFail(A)$ is greater than a specified threshold value is identified as *endangered.*

(c) Estimate $ExpQualLoss(A)$: the expected loss in quality due to one or more of A's predecessors finishing late.

Each of the above steps is discussed in detail below.

Identifying the Predecessors of A. An action P is a predecessor of A if there is a constraint in the STN stipulating that P must finish executing before A begins. P is called a *problem predecessor* of A if the plan given to the agents includes a chain of one or more enabling constraints from P to A. Otherwise, P is called a *scheduler predecessor* of A since, in that case, the requirement that P precede A is due to sequencing constraints added by the scheduler. If P is linked by a single enabling constraint to A, or P immediately precedes A in the schedule, then P is called an *immediate predecessor* of A.

Estimating $ProbFail(A)$ and $ExpQualLoss(A)$. Let $\mathcal{P}(A) = \{P_1, \ldots, P_n\}$ be the set of predecessors of the pending action A. Each action P_i has a discrete probabilistic duration model. The n-tuple, $dc = ((d_1, p_1), \ldots, (d_n, p_n))$ is called a *duration combination* for $\mathcal{P}(A)$ if each d_i is a duration of P_i having positive probability p_i. For each duration combination, dc, the agent takes the following steps:

- $P(dc) = \Pi_i p_i$, is computed, which represents the probability that the duration combination, dc, will occur.
- For all durations, d_i in dc, insert the corresponding temporal constraints into the temporal network.
- If the resulting network is consistent, set $Q_{\text{Loss}}(dc) = 0$. In this case, dc does not lead to an action failure.
- If the resulting network is inconsistent, then dc does lead to an action failure. In this case, the network contains at least one negative cycle. That negative cycle must contain at least one action, F, that fails to meet its deadline. By construction, the possibilities for F are as follows:
  - F is a predecessor of A,
  - F is A, or
  - F is a successor of A.

  In general if the source action of an enabling constraint fails (and hence achieves zero quality), then the target action of that constraint must also fail (by definition). Thus, if F fails, all of its *problem successors* must also fail.[2] Thus, $Q_{\text{Loss}}(dc, F)$, the quality loss due to the failure of F, is approximated as follows:

$$Q_{\text{Loss}}(dc, F) = q(F) + \sum_{S \in PS(F)} Q_{\text{Loss}}(dc, S)$$

  where $q(F)$ is the local quality loss due to F itself, and $PS(F)$ is the set of all problem successors of F.

  After computing this sum, all of the constraints due to F and its problem successors are removed from the network. If the resulting network is still inconsistent, this step is repeated, with a new failed action playing the role of F. The step is repeated until the resulting network is consistent. Afterward, the following estimate of the total quality loss due to the duration combination, dc, is computed:

$$Q_{\text{Loss}}(dc) = \sum_{F} Q_{\text{Loss}}(dc, F)$$

After each duration combination is analyzed, the results are combined as follows:

- Let $\mathcal{F}$ be the set of all duration combinations that lead to an action failure.
- The probability of some predecessors of A finishing late and causing an action failure:

$$ProbFail(A) = \sum_{dc \in \mathcal{F}} P(dc).$$

- The expected quality loss due to some predecessors of A finishing late:

$$ExpQualLoss(A) = \sum_{dc \in \mathcal{F}} P(dc) Q_{\text{Loss}}(dc).$$

If $ProbFail(A)$ is greater than some fixed threshold, then the action A is identified as endangered.

Scheduling Redundant Actions

For each action, A, that has been identified as in danger of failing, the agent can safeguard against the potential failure—and the resultant loss in quality—by scheduling one or more backup actions.[3] The JIT-BF approach combines a passive strategy and an aggressive strategy, each of which is described below.

Passive Strategy. The passive strategy involves the opportunistic scheduling of redundant actions by agents when doing so has minimal deleterious effects on their schedules. The agent, G_A, in charge of the endangered action, A, broadcasts the following information to the rest of the agents:

- The endangered action A
- $ProbFail(A)$: the probability that A will fail
- $ExpQualLoss(A)$: the expected loss in quality from A failing

Any agent, G, that can schedule a redundant action, R, will do so only if the following conditions are met:

- Inserting R into G's schedule will not cause any of G's other actions to become newly endangered.

[2] G is a problem successor of F if there is a chain of enabling constraints from F to G.

[3] Recall that the plan is presumed to contain multiple, redundant, non-mutually-exclusive actions.

- The expected quality loss due to G inserting R into its schedule is smaller than the expected quality loss due to A failing. In particular,

$$\sum_{A_i \in EA_G} ExpQualLoss(A_i) \; < \; ExpQualLoss(A)$$

where the EA_G is the set of endangered actions in G's schedule.

Aggressive Strategy. If the passive strategy does not lead any agent to insert a redundant action into its schedule, then the agent G_A can take more aggressive steps to avoid losing quality should A fail. The aggressive strategy only kicks in when the endangered action A is the next action to execute on G_A's schedule. The agent G_A broadcasts a *coercive* request for help to the rest of the agents. This coercive request compels each of the other agents to generate an optional schedule that includes some redundant action R, even if doing so would have a deleterious effect on their schedule. (The receiving agents can choose which redundant actions to attempt to schedule.) Each of the receiving agents responds by sending the following information back to G_A:

- $ExpQualLoss(R)$

where R is the redundant action. If all of the values, $ExpQualLoss(R)$, exceed $ExpQualLoss(A)$, then the agent G_A releases the other agents from doing their optional redundant actions. Otherwise, G_A selects the optional action R with minimal $ExpQualLoss(R)$ and compels that agent to do that action.

Aborting Redundant Actions

Using either of the above strategies, an agent inserts redundant actions into its schedule over time. Eventually, these actions must be executed. When an agent is executing one of these redundant actions, it can sometimes be advantageous to *abort* that action. Our agents abort a backup action, A, if any of the following conditions are met.

- The deadline for the action A has already passed—in which case A is guaranteed to generate no quality, and hence the continued execution of A cannot contribute anything useful to the multi-agent plan.
- Continuing to execute A would cause other, *more valuable* actions in the agent's schedule to fail to meet their deadlines, with a resultant loss in overall quality.
- An action, A', which is one of the alternatives to A, has already been executed, and the quality that it generated is higher than the quality A would generate should it be successfully completed (i.e., $q(A') > q(A)$).
- An action, A', which is one of the alternatives to A, has already been executed, and:
  - the quality accrued by A' is lower than the quality A would accrue should it finish successfully, however:
  - the probability that A will fail is greater than some fixed threshold (i.e., $p_{\text{fail}}(A) > p_{\text{threshold}}$); and
  - continuing to execute A could lead to greater losses than the quality that would be gained by finishing it:

$$ExpQualLoss(A) > [q(A) - q(A')]$$

Experimental Evaluation

This section describes our experimental evaluation of the JIT-BF strategy. We compared the performance of two teams of agents:

- REACTIVE TEAM: This team of agents used a purely *reactive* approach to dealing with the uncertainty of action durations. They use a centralized scheduler to generate their initial schedule based on the *expected values* of action durations. When inconsistencies arise, they take conflict-resolution actions to repair their schedules, as described in earlier work (Gallagher and Smith 2008).
- JIT-BF TEAM: This team of agents used an approach that augments the reactive strategy of (Gallagher and Smith 2008) with the pro-active JIT-BF strategy described in the last section. Like the reactive team, this team begins with an initial schedule based on expected action durations. The agents on the JIT-BF team used both the passive and aggressive strategies. The aggressive strategy kicks in when the *next-to-execute* action has a high probability of failure and no agent has stepped in to try to schedule a backup action for it.

For these experiments, we used the following parameter settings for the JIT-BF strategy:

- Neighbor Level $= 1$. This indicates that for each pending action, A, the agents collected only the *immediate predecessors* of A.
- Max Pending Actions $= 3$.
- Pending Actions Horizon $= 30$. (roughly equivalent to 3 consecutively executed activities)
- $p_{\text{Threshold}} = 0.1$.

To evaluate the JIT-BF strategy, we conducted two separate sets of experiments. First, we evaluated comparative performance on a set of 56 problems used by DARPA as part of the Year 2 evaluation of the Coordinators program. These problems were designed to test performance under increasing problem size and across a range of coordination problem structures. Second, we radomly generated a separate set of 1350 problem scenarios with varying levels of durational uncertainty and deadline flexibility. This problem suite was designed to more systematically assess the sensitivity of JIT-BF performance to these two problem factors. Both of these sets of experiments are summarized in turn below.

Considering first the experiments performed with the DARPA Coordinators program evaluation problems, we tested both teams of agents on a set of 56 problems drawn from the program's 25- and 50-agent evaluation suites. The larger 50-agent problems contained on average approximately 1500 methods, and were the largest problem size tested extensively in the Year 2 evaluation.[4] The 56 scenarios were divided into 32 25-agent scenarios and 24 50-agent

[4]There was one small set of 100 agent problems also included in the evaluation. However, these problems were artificially structured to make them easier to solve and were used mainly to test scalability of the agent implementation.

scenarios.[5]

To compare the performance of the teams, we computed the *quality ratio* for each (team, problem) pair. The quality ratio for a given team on a given problem is the quality achieved by that team on that problem divided by the highest quality achieved by either team on that problem. Thus, the quality ratio is a number between 0 and 1, where ratios closer to 1 indicate better performance. The quality ratios achieved by the agent teams on the problems described above are given below:

Num Agents	Reactive	JIT-BF
25	0.8997	0.9817
50	0.8873	0.9504

The results for the 25-agent scenarios showed that the agents using the JIT-BF strategy significantly out-performed the reactive agents (with $p = 0.0079 < 0.05$).[6] The results for the 50-agent scenarios also appeared to show a performance gain for the team using the JIT-BF strategy; however the results were not quite significant enough to draw certain conclusions ($p = 0.0635 > 0.05$). One additional point to note in interpreting these results is that the reactive strategy tested here was previously shown (Gallagher and Smith 2008) to dominate a purely proactive approach wherein schedules were generated using maximum durations rather than expected durations (hence guaranteeing that they will execute successfully).

To understand the performance sensitivity of the JIF-BF strategy to the level of duration uncertainty and deadline tightness, we also compared the performance of the reactive and JIT-BF teams on a separate set of randomly generated scenarios. Scenarios involving teams of ten, twenty or thirty agents were generated while varying two other parameters: (1) deadline flexibility (df), and (2) duration uncertainty (du). The deadline flexibility for a high-level task, w, is defined as the quantity

$$df(w) = \frac{deadline(w) - release(w)}{maxDuration(w)}$$

where:

- $deadline(w)$ is an upper bound on the finish time of w;
- $release(w)$ is a lower bound on the start time of w; and
- $maxDuration(w)$ is the maximum duration of w, which depends on the potential durations of its child actions as well as any necessary delays between them.[7]

Notice that a deadline flexibility of 1 indicates potentially no flexibility, whereas values larger than 1 indicate at least some slack in the bounding time window. In our experiments, we used deadline flexibilities of 1.0, 1.2, and 1.4.

The duration uncertainty of a primitive action, A, is the probability that the maximum duration of A will occur (as opposed to its expected duration). In our scenarios, we used three levels of uncertainty: 0.125, 0.25 and 0.333. We hypothesized that the advantage of "risk-taking" strategies would increase as the duration uncertainty decreases.

For each triple, $(numAgts, df, du)$, we randomly generated 50 scenarios, for a total of $9 * 150 = 1350$ scenarios. We computed the quality ratios for each set of 50 scenarios for the reactive and JIT-BF teams. The results are given in Table 1. The results in the table show *deadline flexibility* increasing from left to right, and *duration uncertainty* increasing from top to bottom. Each pair of quality ratios is accompanied by the p value generated by the student's t-test.

All but two of the 27 sets of scenarios showed a statistically significant advantage for the pro-active agents using the JIT-BF strategy over the reactive agents. The deadline flexibility proved to be the most important factor. Scenarios with the least amount of flexibility (i.e., 1.0) showed the largest performance gain. The performance gain decreased as the deadlines became more flexible.

Related Work

Morris et al. (Morris, Muscettola, and Vidal 2001) augmented STNs to include so-called *contingent links*—that is, temporal intervals that are subject to known constraints, but whose duration is not directly under the control of the planning agent. Reflecting the dynamic and uncertain nature of the execution environment, the planning agent does not learn of the duration of a contingent link until its terminating time-point has been executed. Morris et al. presented an algorithm that an agent can use to ensure the consistency of its temporal network no matter what durations the contingent links end up having. Such a network is said to be *dynamically controllable.* Unfortunately, their dynamic controllability algorithm typically requires the agent to add constraints to the network that can severely restrict its flexibility. Hunsberger (Hunsberger 2002) introduced a *temporal decoupling* algorithm that is related to dynamic controllability, but similarly requires restricting the flexibility of the agent. The work closest to ours is (Hiatt et al. 2008), which also investigates the concept of adding redundant actions to previously generated deterministic schedule. In this work, a probabilistic analysis of failure is used in a post-processing step to determine the activities whose potential failure represent the greatest threats to achieving expected quality and these activities are reinforced in order of importance until expected quality can no longer be increased. This work can be seen as complementary to ours, which focuses on optimizing the quality return of previously scheduled activities as execution results become known.

Concluding Remarks

In this paper, we have described an approach to minimizing the impact of activity failure due to durational uncertainty in the presence of deadlines on a joint schedule being exe-

[5] In addition to enabling constraints, scenarios for the Coordinators Program can include other kinds of constraints (e.g., *disabling*, *facilitating* or *hindering* constraints). To conform to the assumptions in this paper, we removed all instances of these other kinds of constraints for the scenarios used in our experiments.

[6] Statistical significance was determined using the student's t-test, for which the common threshold for significance is $p < 0.05$.

[7] An enabling constraint can include a delay. Thus, for example, an action A might enable another action B, but the start of B might be required to occur at least k units after the end of A.

			Deadline Flexibility (df)					
		$numAgts$	1.0		1.2		1.4	
			Reactive	JIT-BF	Reactive	JIT-BF	Reactive	JIT-BF
Duration Uncertainty (du)	0.125	10	0.7545	0.9148	0.8345	0.9523	0.8428	0.9707
			$p = 0$		$p = 0$		$p = 0$	
		20	0.7484	0.9898	0.8937	0.9659	0.9353	0.9685
			$p = 0$		$p = 0$		$p = 0.0132$	
		30	0.7796	0.9894	0.9356	0.9790	0.9364	0.9792
			$p = 0$		$p = 0$		$p = 0$	
	0.25	10	0.6699	0.9091	0.8143	0.9427	0.8653	0.9472
			$p = 0$		$p = 0$		$p = 0.0017$	
		20	0.6749	0.9942	0.8733	0.9730	0.9275	0.9633
			$p = 0$		$p = 0$		$p = 0.0153$	
		30	0.6766	0.9914	0.8677	0.9850	0.9398	0.9660
			$p = 0$		$p = 0$		$p = 0.0762$	
	0.333	10	0.7041	0.9344	0.7680	0.9119	0.8426	0.9383
			$p = 0$		$p = 0$		$p = 0$	
		20	0.6948	0.9776	0.8638	0.9660	0.9055	0.9708
			$p = 0$		$p = 0$		$p = 0$	
		30	0.6866	0.9790	0.8751	0.9598	0.9339	0.9611
			$p = 0$		$p = 0$		$p = 0.0722$	

Table 1: Quality Ratios for Reactive vs. JIT-BF Strategies

cuted by a team of collaborative agents. Our approach assumes the availability of an explicit model of durational uncertainty and centers around the idea of an agent recruiting other agents to schedule redundant activities when it detects that one of its currently scheduled activities is in jeopardy of not completing by its deadline. Whenever an agent commences execution of the next activity in its schedule, it uses its underlying STN-based schedule representation to analyze the likelihood of a deadline failure within the immediate future. If an endangered activity is identified, the agent coordinates with other agents who might be able to perform a redundant (back up) activity to ensure that the executing plan will continue to produce the best possible result. We have presented experimental results which show that this technique, referred to as *Just-In-Time Backfilling*, significantly amplifies a previously developed reactive strategy for this same class of multi-agent scheduling problem across a range of quality-maximization scheduling scenarios.

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

References

Boddy, M.; Horling, B.; Phelps, J.; Goldman, R.; Vincent, R.; Long, A.; and Kohout, B. 2005. C_taems language specification v. 1.06.

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. F. 2008. Recovering from inconsistency in distributed simple temporal networks. In *Proc. FLAIRS- 2007*.

Hiatt, L.; Zimmerman, T.; Smith, S. F.; and Simmons, R. 2008. Reasoning about executional uncertainty to strengthen schedules. In *Proceedings ICAPS-2008 Workshop on A Reality Check for Planning and Scheduling Under Uncertainty*.

Hunsberger, L. 2002. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 468–475. Menlo Park, CA: American Association for Artificial Intelligence.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 494–499.

Smith, S. F.; 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 (AAMAS 07)*.

Wagner, T.; Phelps, J.; Guralnik, V.; and Riper, R. V. 2004. COORDINATORS: Coordination managers for first responders. In *Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*. IEEE Computer Society.