Distributed Coordination in Uncertain Multiagent Systems

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

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

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

ABSTRACT

We consider real-time multi-agent coordination in a dynamic and uncertain domain addressing both distributed state information and partial knowledge of the common reward function. The challenge is to find functional strategies when bounded rationality hinders the ability to encompass the values of possible sample paths of the system. This paper discusses a new approach based on assigning agents to monitor portions of the reward structure for which they aggregate and propagate appropriate profiles which compactly represent relevant information used for policy modification. This approach shows promise as an alternate and potentially superior technique with respect to current decision-theoretic and scheduling approaches.

1. INTRODUCTION

We address coordinated execution of activities of a multi-agent team in domains with uncertainty. Joint operations in military settings, large-scale disaster rescue, project/personnel management in global enterprise settings, and multiple-rover missions in science-discovery are some examples of dynamic execution environments where effective and efficient coordination is crucial to success. In these domains, (i) *uncertainty and sequential decision-making* explodes the computational complexity for optimal policy generation, (ii) *partial state information* obfuscates the triggers for actions, (iii) *incomplete policy knowledge* hinders effective policy-modification, (iv) *subjective views of the team reward function* impedes evaluation of current and future performance or potential policy changes, and (v) *environmental instability* (meaning parameters that define the reward, uncertainty or constraints can change during the execution phase) requires fast adaptability in online reasoning. Discovering an approach that functions with computationally-bounded agents making decisions under these conditions in real-time is an extremely challenging task.

Centralization can generate fully coordinated policies but puts a high computational burden on a single agent. When bounded rationality is considered, this agent cannot make decisions in a timely manner. While we do not address communication failures or delay in this study, centralization will suffer under these extensions. Traditional AI methods that model the distributed and subjective natures of the problem such as SAT or DCR techniques (DCSP, DCOP) cannot currently handle uncertainty or sequential decision-making without cumbersome encodings. Standard OR methods such as mathematical programming or decision-theoretical approaches handle uncertainty but not partial observability and multi-agent decision-making under bounded rationality. While decentralized versions of those methods have developed to handle the latter concerns, none adequately address the subjective view of the the reward function. Indeed, researchers have been tackling problems with many of the characteristics mentioned above, however, conceiving techniques that handle *all* aspects of these settings is a nascent area of research.

In this paper, we present a concept where agents are assigned responsibility for portions of the reward structure. This responsibility is to aggregate and disseminate information, stored in *profiles* in a neighborhood of reward nodes. The key is to create profile parameters that (a) can be computed and communicated quickly and (b) contain metrics that are immediately relevant to policy modification. This approach allows agents to collectively deliver effective approximations of the relevant global information to the appropriate agents in a timely manner The benefits of this approach are verified by a comprehensive independent evaluation against extensions of currently prominent decision-theoretic and scheduling schemes.

2. PROBLEM MODEL

Here, we present a model that contains all the challenging properties discussed earlier. The model is an instantiation and extension of the TAEMS framework [5]. Every agent in the team has a set of activities that it can perform but it can execute at most only one at a time. Each activity m has probabilistic outcomes where duration δ^m and quality q^m occur with probability $p^{\delta,m}$ and $p^{q,m}$ respectively:

$$q^m \in \{q_1^m, \ldots, q_{N^{q,m}}^m\} \text{ w.p. } p^{q,m} \in \{p_1^{q,m}, \ldots, p_{N^{q,m}}^{q,m}\}$$
$$\delta^m \in \{\delta_1^m, \ldots, \delta_{N^{\delta,m}}^m\} \text{ w.p. } p^{\delta,m} \in \{p_1^{\delta,m}, \ldots, p_{N^{\delta,m}}^{\delta,m}\}$$

Here, $N^{q,m}$ and $N^{\delta,m}$ are the number of quality and duration outcomes, respectively, for activity m. Each activity can be started only once and must begin after a release time r^m and finish at or before a deadline d^m in order to obtain

positive quality. Let $q^m(t)$ denote the quality of method m at time t with $q^m(0) = 0$ until successfully executed. If an agents starts a method at $s^m \geq r^m$ and it ends at $e^m = s^m + \delta^m \leq d^m$ then $q^m(e^m) = q^m$ where δ^m and q^m are drawn from given distributions. Only the agent that owns a method knows its statistics at $t = 0$ and knows its status at any t. In addition, only the agent has current knowledge of its policy at all times. Thus, this model have incorporated uncertainty, decisions over time, partial state information and incomplete policy knowledge

The team reward is a function of $\{q^m(t)\}$, the qualities of all activities, and the agents' objective is to maximize this reward at some terminal time T. One way this function can be composed is with a tree where the activities are leaf nodes. Each non-leaf node n is associated with an *ancestral operator* $\odot_n(\cdot)$ (such as max, min, or sum) which takes the qualities of its children as input: $q^n(t) = \odot_n(\{q^{\tilde{n}}(t)\}_{\tilde{n}\in C(n)})$. The output of the root node is the team reward function. An extension is to have a link between nodes that represents a *directional operator* $\overrightarrow{\odot}_{n_s,n_t}(\cdot)$ where n_s and n_t are the source and target node. An example directional operator is *enables* where the quality of the source must be positive at the start time of the target for the target to obtain positive quality (if the target is a non-leaf node, it can be interpreted as multiple links from the source to all descendant leaf nodes of the target). The quality of a node would then be $q^n(t) = \odot_n(\cdot)\Pi_{\hat{n}\in S(n)}\overrightarrow{\odot}_{\hat{n},n}(\cdot)$ where $S(n)$ are the source nodes of all directional operators whose target is node n. The argument of all operators can vary based on their type. While additional modifications will increase the scope of expressible reward functions, the preceding is sufficiently rich to create problems of great complexity. In addition to frameworks such as TAEMS, DCOPs [9] also fit this representation where leaf nodes are variables and the operators are the matrix rewards on the DCOP links. We can now define an agent's *subjective view* of the reward function, i.e., knowledge of only a subset of the reward network. We consider the case where each agent sees all ancestral nodes of activities they own, and any nodes and links that connect to its activities and their ancestral nodes via directional operators. Finally, environmental instability is introduced because during execution, the reward network, the temporal constraints (r^m, d^m) and the distributions can change, so even an optimal policy for the original problem can be invalidated at run-time. Thus, this model incorporates all the complexities discussed earlier.

3. EXAMPLES

We present some examples to show the application of the model and illustrate some of the complexities of the problem. In Figure 1, we show how a problem can be represented with the model elements discussed, for an example where a team is trying to evaluate their software system by a deadline. The shaded squares at the lowest layer represent the activities that can be performed by various agents, in this case, human members of the team. The ovals represent subtasks of the overall goal. The quality in which the overall goal can be aided both by analyzing a set of experiments and engaging in a progress review meeting, and is represented by a *sum* operation of the qualities of those two subtasks. To analyze experiments, they must first be run and then the results analyzed. The quality of this subtask is bottlenecked by the quality of the experiments and the quality of the results analysis, thus the *min* operator. Furthermore, the analysis of the results can occur only after the experiments have been run. This is represented by the *enables* directional operator. The progress review meeting can occur at two different times. Here, the quality of the subtask is represented by a *max* operator, indicating that a second meeting only needs to be held if it can improve upon the quality of the first meeting. The *max* operation indicates on the "run experiments" tasks represents a similar relationship. Here, the analysis of the results can will be aided immediately with more help, thus the *sum*. The *sync* operator for the meetings, indicates that those activities must be scheduled to start at the same time by all participants.

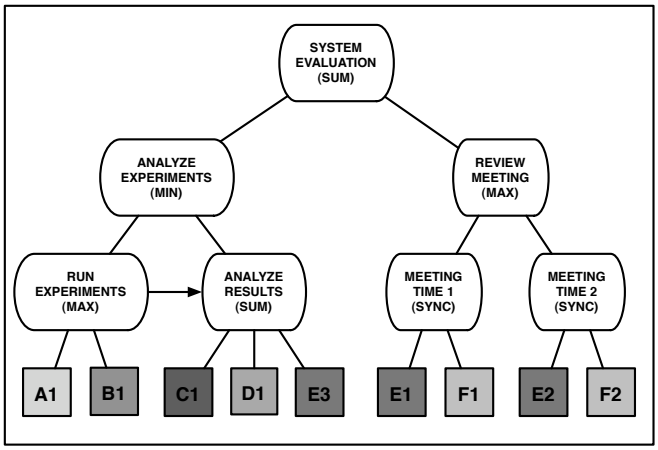

Figure 1: An example of model usage

Consider a scenario where activities $A1, C1, E3, E1, F1$ constitute the initial plan that the team is expecting to execute. Activity $A1$ fails due a hardware error (outcome uncertainty). Agent B then volunteers to execute $B1$ on his hardware, however, it will not be available until later (temporal constraints). This delay disallows Agent C from participating in the analysis due to his availability (resource constraints). Agent D picks up the slack ($D1$). Activity $E3$ takes longer than expected (duration uncertainty) forcing Agent E to cancel either $E3$ or $E1$. Agents E and F replace $E1, F1$ with $E2, F2$ (move the meeting) allowing for E3 to complete. The preceding examples is a microcosm of some of the difficulties that arise in coordinating in an uncertain temporally constrained environment.

The complexities of the subjective view can be illustrated through an example shown in Figure 2 which shows the full *objective view* of the reward function network and the *subjective views* of each agent. The leaves indicate various activities owned by agents A, B and C. The subjective view indicates both that portions of the reward are unknown and also the status of unseen activities is unknown. For simplicity, henceforth, quality outcomes will be either 0 or 1:

(1) Full status knowledge but subjective view of reward: If agent C knew that A1,B1,A2 failed ($q^n(t) = 0$) and B2 succeeded ($q^n(t) = 1$)) before it had to choose one of C1 or C2 to execute, it could not select the best activity accurately. If A1,A2,B1, and B2 were under the visible MIN, agent C

would choose C2. If A1,A2,B1, and B2 were under the visible MAX, agent C would choose C1.

(2) Objective view of reward but partial status knowledge: If agent B had the objective function but did not know the status of agent C's activities, it could not select the best activity accurately. If C1 and C2 failed, agent B would choose B2. If C1 and C2 succeeded, agent B would choose B1.

(3) Full status knowledge and objective view but no statistics or policy knowledge: If agent A knew had the objective view and the status of all activities (B1 and B2 have succeeded, C1 has not been started yet), it could not select the best activity accurately. The choice depends on the probabilities ($p^{q,A1} > p^{q,A2}p^{q,C1}$?) but $p^{q,C1}$ is unknown. Even if it was known, agent A does not know if agent C plans on executing C1.

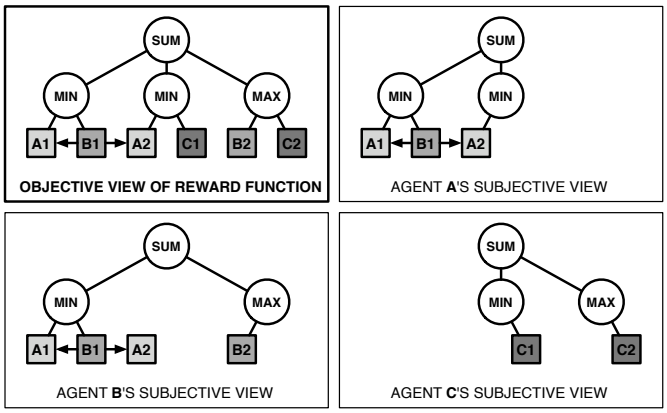

Figure 2: Objective and subjective views of a reward function

While the problems shown in these examples might seem trivial, the difficulties in uncertainty and partial knowledge of state and reward amplify as the scale of the problem gets larger. A reward network with over 1000 nodes is shown in Figure 3.

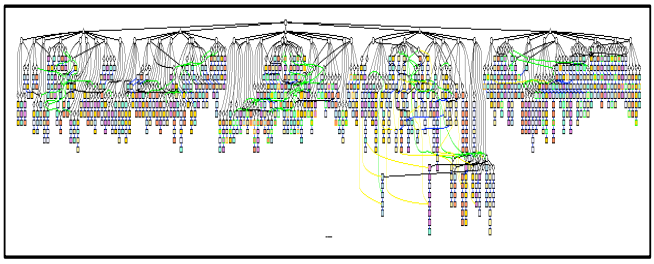

Figure 3: Rewards network for a large-scale problem

4. APPROACH

The key to solving this problem is to find a method to disseminate the relevant global information that agents need but do not have, while respecting bounded-rationality to produce real-time decisions (e.g., a naive centralization procedure would incur large computation cost and communication delays). The key to our solution is the creation of profiles for each node in the reward network. Profiles are a characterization of the substructure of the reward network, rooted at the given node, that are compact. Instead of centralizing state information and reward structure for all activities in the underlying structure, which would be cumbersome and would not scale, profiles try to capture the critical information needed for decision-making with a minimal number of parameters. These parameters are calculated from input obtained only from the neighboring nodes, thus the interaction scales as a function of link density of the reward node.

4.1 Reward Node Assignment

The first step in our approach is to assign an agent to each node in the objective view. The agent will be responsible for updating and disseminating a *profile* $\mathbb{P}_n$ for node n. Updates depend on the profiles on nodes that are neighbors of node n (i.e., connected to n via an ancestral or directional link). Then, if $B(n)$ are the neighbors of node n, a profile update can be represented as $\widetilde{\mathbb{P}}_n = \otimes_n(\mathbb{P}_n, \cup_{\tilde{n}\in Bn}\mathbb{P}_{\tilde{n}})$. The aggregation operator, $\otimes_n$, represents all the calculations necessary for metrics contained within the profile, and will depend on the operators that connect n to its neighbors. Currently, we assign nodes randomly among agents who can see the node in their subjective view to avoid bottlenecks[1]. Figure 4 displays an assignment and consequences for the example in Figure 2. The shaded nodes are those for which the assigned agent is responsible and the white nodes are those with which the assigned agent must communicate ($B(n)$). The agent may be required to communicate with a node that is not in its initial subjective view. Assignment of reward node responsibility allows an agent to expand its subjective view to include all nodes one link removed from the assigned node.. The responsibility graph may also be disjoint. Whenever a profile is updated, it creates a flow of messages throughout the multi-agent system. The computation for the update and the type and targets of the messages are a function of which metrics within the profiles are being updated. The sources of information updates are status observations made by the leaf nodes (e.g. activity starts, ends, etc.) which update a local profile and begin the process of dissemination.

4.2 Profile Metrics

Now that question of *who* has been answered, we address the *what* with respect to profiles. Each profile contains a set of metrics. While the metrics differ in their meaning and usage, they have the following in common: (1) the metrics can be updated quickly when input from neighboring nodes arrive; (2) the metrics have significant and immediate use for policy modification.

Before we delve into details, we present some caveats: (a) We omit discussion of some metrics, approximations and policy-modification schemes due to space. We hope to convey the high-level ideas that go into developing the general approach and their utility. In Section 5, the independent evaluation of our system prevented any possibility of tuning parameters for performance, as we did not know the problems we would

[1]Alternate assignment strategies are topics under investigation

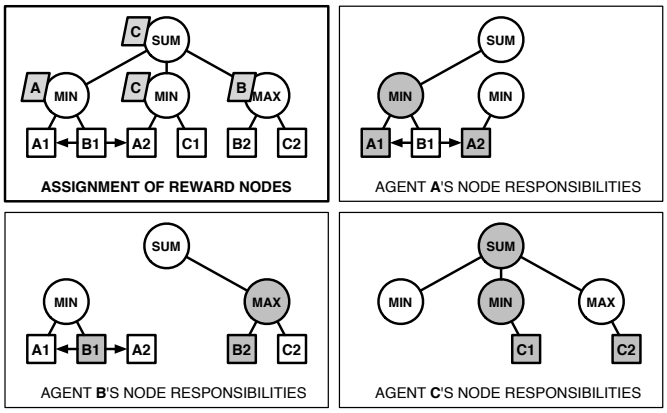

Figure 4: Reward node assignment and consequences

be asked to solve. (b) We discuss the metrics in the context where the team objective is to maximize the probability that the quality at the root node is positive ($P(q^0(T) > 0)$). The ideas here are extendable to the case where the team objective is to maximize expected quality ($E[q^0(t)]$), which is what the independent evaluation tested.

The characteristics of this problem make it intractable to generate a policy that can prescribe optimal actions for all states. Thus, the team is given an initial schedule which is simply a mapping from times to actions ($\{t_1$:start m_1, t_2:start m_2, ...$\}$) without any instruction for contingencies (m_1 still executing at t_2, etc.). Thus, all contingency planning (i.e., policy generation and modification) must be in the system. As mentioned, it is too complex to plan for all contingencies (states). One must then employ a policy that is reasonable for some subset of contingencies. The evolution of the system will eventually move the team outside the subset where the policy performs well. The key is to discover the *when* and *how* to modify policies. The metrics in the profiles serve this purpose. We now discuss the intuition, calculations, message flow, and utility for four profile metrics: schedule probability (p_n^S), potential probability (p_n^*), schedule importance (α_n^S), and potential importance (α_n^*).

Schedule Probability (p_n^S): The schedule probability metric is the likelihood that the current node will obtain positive quality at $t = T$ under the current policies of all agents. This metric combines status information (whether and when a method started executing), probability information (duration distribution), constraint information (release and deadline) and policy information (whether executing the method is part of a likely contingency). The schedule probability of an activity is defined as:

$$p_m^S = (1 - p^{uf}) p_m^{we} p_m^{success} \prod_{n \in E_m} p_n^S.$$

where $p^{uf} \ll 1$ is an estimate of unmodeled failure (addressing environmental instability and also assuring $p_m^S = 1$ only when completed with positive quality); $p_m^{success} = \sum_{i:q_i^m \neq 0} p_i^{q,m}$ is the probability of achieving positive quality intrinsic to the activity; E_m is the set of nodes that are enables sources whose target is activity m; and p_n^S is the schedule probabilities of those sources. The factor p_m^{we} captures the effect of the method's window as determined by its release, deadline, the current schedule and its execution status on its likelihood of completing successfully. The window for an executing method is the current time to the deadline. The window for a method to be executed is an estimated start time to the deadline. The window effect is the decrease in probability of success due to these factors and is calculated by aggregating the probabilities of durations that fit within the window. Thus, a schedule probability update occurs on an update from its enables sources or a local state update that changes a window-effect or success probability. After an update, it sends an update to its parent and its enables targets.

The schedule probability profile for a non-leaf node is equipped with an aggregator function that depends on the associated ancestor operator. For a *max* node, the schedule probability is the likelihood that at least one child will succeed. Formally, if the node n has $\odot_n(\cdot) = \max(\cdot)$, where $C(n)$ denotes the children of node n, then we have

$$p_n^S = p_n^{we} \left(1 - \prod_{\tilde{n} \in C(n)} \left(1 - \widetilde{p}_{\tilde{n}}^S \right) \right),$$

The window effect p_n^{we} is an approximation that accounts for the execution load put on various agents for that node to have the given probability of success. To do this, leaf-node pass up estimates of resource consumption (e.g. the maximum duration) for each activity. Non-leaf nodes aggregate this information along with window information available at the node. This allows each node to calculate a ration of the availability window and load for each agent, which can be used to scale down the probability of success for that node. A non-leaf node will update its schedule probability when it obtains an update from one of its children, or one of its enables sources. After an update, it also sends an update to its parent and its enables targets.

The significance of schedule probability is that it answers the *when* question. If the schedule probability of a particular node begins to fall, it means that we are moving into a regime where the current policies are deteriorating with respect to that node. The system can use this information to decide whether to begin a policy-modification procedure to rectify this situation. Thus, the schedule probability is the critical metric which decides whether multi-agent policy-modification will occur. In addition, the schedule probability is necessary to calculate the schedule importance which is discussed later.

Potential Probability (p_n^*): The potential probability metric is the likelihood that the current node can obtain positive quality at $t = T$ under *any* policy set available to all the agents. This metric combines status information, probability information, constraint information but does not use current policy information. A potential probability profile for an activity that cannot achieve positive quality with positive probability (e.g. the deadline is too near, it has been aborted, etc.) is $p_m^* = 0$. Otherwise, the potential probability profile is calculated identically to a scheduled probability profile for an activity that has not begun execution while ignoring the window effect p^{we} which depends on

current policy. Once an activity is completed successfully (with positive quality), $p^*_m = 1$.

Potential probability profiles for non-leaf nodes are much more complex because their profiles depends on which subset of children one wishes to consider. the critical criteria is determining when the potential probability of a task is zero or one. It is zero ($p^*_n = 0$) if (as mentioned above) it cannot achieve positive quality with positive probability. This occurs for a **max** node if $p^*_{\tilde{n}} = 0\ \forall \tilde{n} \in C(n)$ and for a **min** node if $p^*_{\tilde{n}} = 0$ for any $\tilde{n} \in C(n)$. Similarly, the potential probability for a task is one ($p^*_n = 1$) if for a **max** node if $p^*_{\tilde{n}} = 1$ for any $\tilde{n} \in C(n)$ and for a **min** node if $p^*_{\tilde{n}} = 1\ \forall \tilde{n} \in C(n)$. The updates flows are identical to those in the schedule probability metric. The utility of this metric is that is necessary to compute the potential importance, which we discuss later.

Schedule Importance (α^S_n): While the probability profiles metrics might seem straightforward, they are critical as they are the base input for importance profile metrics. Schedule importance reflects the marginal rate at which a node contributes to the overall team goal, i.e., if the schedule probability of this node was increased by Δp^S_n, the schedule probability of the root node would increase by $\alpha^S_n \cdot \Delta p^S_n$, assuming all other schedule probabilities stayed constant. While this assumption is not always true, it is a useful approximation because profiles are typically updated at a faster time-scale than the intervals between agent policy-modification instances. We note that this method may suffer when agent policy-modification instances are rapid and aligned.

This method is very useful because performance depends heavily on agents finding the best possible way to allocate their execution time. When choosing between multiple local policy-modifications that involve adding an activity that was not part of the previous policy, it would be prudent to choose the one that contributes most to the team goal. The importance of an activity provides a value that allows for this discrimination among the available choices.

The schedule probability at the root, $p^S_{n^*}$ is ultimately a function of the schedule probabilities of the activities that have been propagated through the aggregator operators at the nodes. One can approximate the change in probability of success (positive quality) at the root as a function of an increase in schedule probability of a particular node as follows:

$$dp^S_{n^*} = \frac{\partial p^S_{n^*}}{\partial p^S_n} dp^S_n =: \alpha^S_n dp^S_n$$

where we define the *schedule importance* of a node n, denoted as α^S_n, to be the marginal rate of contribution to the root. As written above, the node cannot determine its importance factor as $\frac{\partial p^S_{n^*}}{\partial p^S_n}$ depends on full reward information (an objective view) that may not be available locally. However, we can decompose the expression as follows:

$$dp^S_{n^*} = \left(\frac{\partial p^S_{n^*}}{\partial p^S_{\rho(n)}} \frac{\partial p^S_{\rho(n)}}{\partial p^S_n} + \sum_{\tilde{n}: n \in E_{\tilde{n}}} \frac{\partial p^S_{n^*}}{\partial p^S_{\tilde{n}}} \frac{\partial p^S_{\tilde{n}}}{\partial p^S_n} \right) dp^S_n$$

where $\rho(n)$ is the parent node of node n, and $E_{\tilde{n}}$ is the set of enables source nodes for node $\tilde{n}$. If we let $g^{\hat{n}}(\cdot) = \frac{\partial p^S_{\hat{n}}}{\partial p^S_n}$, we have:

$$\alpha^S_n = \alpha^S_{\rho(n)} g^{\rho(n)}(\cdot) + \sum_{\tilde{n}: n \in E_{\tilde{n}}} \alpha^S_{\tilde{n}} g^{\tilde{n}}(\cdot)$$

This decomposes the schedule importance factor into a function of parameters of the node's parent and targets which lie in the node owner's subjective view and responsibility graph; thus, we are now able to calculate it as a profile. We obtain $\alpha^S_{\rho(n)}$ and $\{\alpha^S_{\tilde{n}}\}$ through profile propagation and $g^{\hat{n}}(\cdot)$ can be calculated locally given the locally available schedule probability profiles as follows: $g^{\hat{n}}(\cdot) = (1 - p^S_{\hat{n}})/(1 - p^S_n)$ if $\hat{n} = \rho(n)$ and $f^{\hat{n}}(\cdot) = \max(\cdot)$ and $g^{\hat{n}}(\cdot) = p^S_{\hat{n}}/p^S_n$ if $\hat{n} = \rho(n)$ and $f^{\hat{n}}(\cdot) = \min(\cdot)$ or if $n \in E_{\hat{n}}$. A node gets its importance through the importance of its parent and the importance of its targets scaled by a factor that quantifies how much it contributes to the importance of its relevant neighbors. Thus, importance is updated whenever a schedule probability update is received from a child or enables source, or an importance update is received from the parent or enables target. An importance update sends updates to all children and enables sources. Because the aggregator functions for the quality accumulation functions in this paper are linear in schedule probability, the partial derivative approximations are exact, however, the assumption that the node is increased in isolation may not always hold.

The utility of schedule importance is when an agent has a gap (i.e., the current policy implies that no methods are to be executed for the imminent future given the current (estimate of) state), an agent can consider a policy-modification to insert an activity. By applying the schedule importance factor and choosing the activity $m^* = \arg\max_m \{\alpha^S_m \Delta p^*_m\}$, the agent will choose the activity that maximizes the contribution to the root.

Potential Importance (α^*_n): While schedule importance of a node is a positive real quantity, $\alpha^S_n \in I\!R^+$, the potential importance of a node is a mapping to a binary space, $\alpha^*_n \in \{0, 1\}$. This is because it characterizes if a node is *capable* of contributing to root schedule probability under *any* policy, which is a boolean condition. Potential importance is calculated using potential probability in a manner analogous to schedule importance as follows:

$$\alpha^*_n = (p^*_n > 0) \wedge \left(\alpha^*_{\rho(n)} \vee \alpha^*_{\tilde{n}_1} \vee \ldots \vee \alpha^*_{\tilde{n}_L}\right)$$

where $\{\tilde{n}_1, \ldots, \tilde{n}_L\}$ are the targets of node n. Intuitively, for a node to have potential importance, it must be able to contribute under some policy ($p^*_n > 0$) and either its parent or one of its targets must have potential importance. The updates for potential importance are analogous to those of schedule importance. The utility of the potential importance metric is to determine when an activity can no longer contribute to the team goal. Whenever $\alpha^*_m = 0$, we can modify either status (by aborting an executing activity that has no potential importance) or local policy (by removing the insertion of m as an action in any contingency) to create opportunities for other activity insertions.

Pareto Profiles $\mathbb{P}_n$: The importance profiles capture information that is useful for single-agent policy-modifications which are essentially insertion and removal of activities. However, in order to improve performance or recover from failure

(due to an undesired outcome or exceedingly long duration), one may have to make a policy-modification involving multiple agents. In fact, this is the key to effective coordination. If a system is unable to make effective multi-agent policy modification, it cannot work in any problem where performance is coupled through actions of more than one agent. The profile-based approach is well suited to address these problems because the schedule probability, as mentioned earlier, is able to identify the node(s) in the reward network where the current policy (evaluated over multiple agents) is likely to perform poorly. Once identification of a problematic node occurs, we need a way to find to improve the performance of the node, if possible. To this end, each node's profile contains a set of options associated with it, in case a problem occurs. Each option is a combination of activities that is a subset of all activities that are descendants of that node. It would be too expensive and inefficient to keep every combination (i.e., every subset) at each node. However, we can prune the desirable set of combinations by using the concept of Pareto-optimality.

Each activity m has a load t_m which is an estimate of its resource consumption, i.e., how much of the agent's time it will use. The most conservative estimate would be the maximum duration of that activity. Thus, each activity is characterized by a pair $(p*_m, t_m)$ where p^*_m is the potential probability. Now consider a reward node n which has M children that are activities. The potential probability of n depends on which activities under it are to be executed. If a single activity $m \in \{1, \ldots, M\} = \mathcal{M}$ is to be executed, then $(p^*_n, t_n) = (p^*_m, t_m)$. If a subset of activities $c \subseteq \{1, \ldots, M\}$ are to be executed then the potential probability for that combination would be $p^*_c = 1 - \Pi_{m \in c}(1 - p^*_m)$ and $t_c = \sum_{m \in c} t_m$, if n was a *max* node. The number of combinations of activities that could be execution is the power set of $\mathcal{M}$. However, many of these combinations are dominated, i.e., there exists another combination that has a higher potential probability with a lower total load. The undominated combinations are determined by the Pareto frontier of the set of all combinations. It is this set of combinations and their potential-profile and total-load pairs that form the Pareto profile for node n

$$\mathbb{P}_n = \{(p^*_c, t_c) : c \in \mathcal{P}(C(n)), \nexists \tilde{c} \in \mathcal{P}(C(n)), \text{ s.t. } (p^*_{\tilde{c}} > p^*_c) \wedge (t_{\tilde{c}} < t_c)\}$$

where $\mathcal{P}(C(n))$ denotes the power set of the children of node n. A depiction of a Pareto profile is shown in Figure 5.

If a reward node's children are non-leaf nodes (i.e., tasks instead of activities), the input into the node is a Pareto profile instead of a potential-probability total-load pair. The Pareto profile for that nodes is calculated as follows. We consider the case where the node is a *max* task. The node with N children gets the profiles $\{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_N\}$. Given any T which is a subset of children $T \subset \{1, \ldots N\} : T \neq \emptyset\} =: C$, we then have a T-tuple of (p^*, t)'s , one from each child in T: $\mathbf{x_T} = \bigotimes_{i \in T}(p^*_i, t_i)$, where $\bigotimes$ denotes the operator that combines the potential-probability total-load pairs. We define $X^*_T = \{\mathbf{x_T} : (p^*_i, t_i) \in \mathbb{P}_i \ \forall i \in T\}$ to be the set of all T-tuples constructed from elements of Pareto profiles of children. The set of all (p^*, t)'s, one calculated for each T-tuple in X^*_T is then denoted by $S_T =$

$$\left\{(p^*, t) : \mathbf{x_T} \in X^*_T, p = p^{we}_{\mathbf{x_T}} \left(1 - \prod_{j \in T}(1 - p^*_j)\right), t = \sum_{j \in T} t_j\right\}$$

where $p^{we}_{\mathbf{x_T}}$ is an estimate to compensate for agent load similar to the window effect mentioned earlier. The Pareto set for the T-combinations of children is then $\mathbb{P}^T_n = ParetoSet(S_T)$ and the Pareto profile for the node is $\mathbb{P}*_n = ParetoSet(\cup_{T \subset C} \mathbb{P}^T_n)$.

When a problem node is identified, the combinations in the Pareto profile are sent to a scheduler that polls the agents involved in the combinations for their current policies. The scheduler determines which of the combinations are feasible (i.e., can these activities be inserted without damaging the performance of activities already planned on being executed) and chooses the combination that best helps improve the node. There are many addition details for alternate reward node types including optimizations that are not discussed here due to space, but the general idea of a Pareto profile to provide options for multi-agent policy modification is the key idea of the system. These Pareto profiles can be computed using nearest neighbor communication similar to schedule and potential probability propagation.

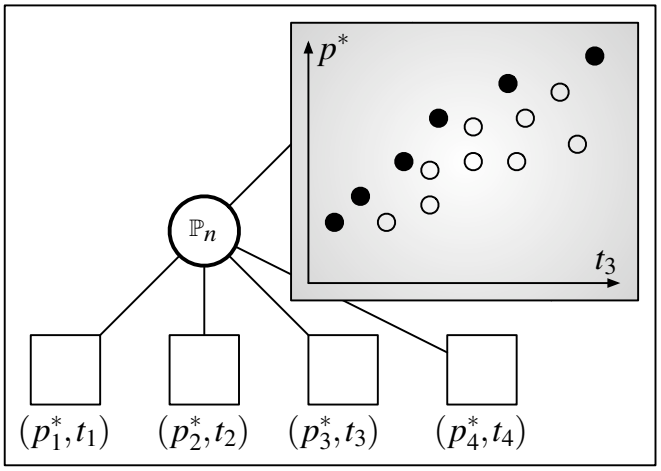

Figure 5: Pareto profile for a reward node

In summary, our approach prescribes the assignment of agents to update and communicate profiles for all nodes in a network representation of the team reward function. The profiles contain metrics which can be calculated quickly using simple operations and neighbor information (from the reward network) to provide immediate and significant input for policy-modification. The metrics described here combine to create a flow of information necessary to address the characteristics of the problem, while generating good policies for agents.

5. EXPERIMENTS

This research was part of substantial effort to address the problems with the all the complexities discussed in the introduction[2]. It involved three main teams which included many

[2]The work presented here is funded by the DARPA COORDINATORS Program under contract FA8750-05-C-0032.

universities, research laboratories and corporations both cooperating and competing to design novel solutions.

Our Profile-Based Coordination (PBC) approach was compared to one that utilized a Distributed-Markov-Decision-Process (Dist-MDP) method and another, Flexible Interval Scheduling (FIS), which utilized temporal networks in a classical scheduling network. The Dist-MDP approach is based on calculating the appropriate subspace of the state space that the evolution of the system will follow and reasoning over that subspace. The rewards at the frontier of this space and policies when outside this space are obtained through greedy search [10]. The FIS approach utilizes a flexible interval for starting methods to absorb the uncertainty and uses incremental revisions to the schedule that attempt to maximize stability of agent activities [15]. While all three approaches are different in their solution methodology, they all attempt to solve the same problem, i.e., given a reward network and an initial schedule, create a multi-agent system that can adapt in real-time to uncertainty and dynamism, to maximize the quality achieved at the root of the reward network.

All approaches were required to be embedded in a real agent that was capable of sensing, computing and communication on its own machine while collaborating with agents on other machines driven by an independent simulator[3]. All research groups submitted their respective agent systems to be evaluated independently on a large suite of test cases. There were over 2600 problems (reward networks) that were run multiple times for problems where the optimal solution was calculable. In addition, there was testing on larger-scale problems that had up to 10 agents and about 500 network nodes. The problems were not known ahead of time so no approximations or thresholds within the system could be tuned to the evaluation. The results for the small-scale experiments are shown in Table 5. Each class denotes a particular type of experiment, e.g., "Synchronization" denotes experiments with a large percentage of *sync* operators, "Temporal Tightness" denotes experiments where the temporal constraint (release and deadline) intervals were very close to the largest durations of the activities, "Chains" denotes a large number of linked directional operators. These tests involved a handful of agents and a number of nodes sufficiently small such that a policy from a centralized MDP solver could be calculated. The numbers shown are the mean qualities of the various approaches as a fraction of the expected quality from the centralized MDP solution. The overall results, which were determined to be statistically significant, showed that our approach outperformed these extensions of classical solution methods.

Of more interest are the results for the large-scale experiments are shown in Table 5. In this evaluation, a score of 1 was given to the trial with the highest quality for a particular experiment. The scores of the other trial of all systems were scaled proportionally. Each system had a regime in which it performed well, however, the overall score of our system was significantly higher than the score of the other systems (.91 vs .77 and .62). We note how increasing scale has a drastic effect on the ranks of the performance of the systems as captured by the "New Task Arrival" class, which denotes experiments with dynamics in the reward network. Most importantly, our system outscored the other systems by the largest difference (.99 vs .58 and .22) on the hardest problem category, "Hard Mix + Overlap" which combined many of the challenges of the other classes.

Class	PBC	Dist-MDP	FlS
Dynamics	0.98	0.99	0.98
Interdependence	1.00	0.95	0.98
Chains	1.00	0.95	0.93
Temporal Tightness	0.98	0.94	0.90
Synchronization	0.99	0.92	0.81
New Task Arrival	0.86	0.96	0.95
Overall	**0.97**	**0.95**	**0.91**

Table 1: Results for small-scale experiments

Class	PBC	Dist-MDP	FlS
New Task Arrival	0.88	0.61	0.94
Window Tightness	0.84	0.93	0.64
Synchronization	0.97	0.87	0.70
Hard Mix	0.88	0.88	0.60
Hard Mix + Overlap	0.99	0.58	0.22
Overall	**0.91**	**0.77**	**0.62**

Table 2: Results for large-scale experiments

6. RELATED WORK

The coordination of multi-agent systems in dynamic, distributed, stochastic, temporally-constrained and partially-observable domains is a challenging problem. One way to control cooperative multi-agent systems under such conditions is through Decentralized MDPs (DEC-MDP or DEC-POMDP). Unfortunately, the most general decision-theoretic models for this problem have been proved to be extremely complex (NEXP-complete) [14, 1, 13].

In order to lower complexity, some models restrict the types of interactions allowed in the system, the amount of communication among agents, or the general features they are able to address. One of such models is the Opportunity Cost DEC-MDP (OC-DEC-MDP) [2]. This model bases its computation in local policies, taking into account the loss in value produced by its local computation. The approach also enforces temporal constraints among the activities to eliminate the communication among agents. Although,our approach also computes local estimates of the probability of success for each activity, such estimates are propagated in a distributed manner to the interacting agents. Other approaches allow the agents to communicate to exchange local policies. The DEC-POMDP with communication (DEC-POMDP-Com) [7] presents a first greedy meta-level approach

[3]Communication between agents was handled via the simulator. There were no bandwidth limitations, packet loss, or noisy channels in this study

to agent communication, unfortunately the number of agents considered by the framework is very small.

Another way of modeling the multi-agent coordination problem is through traditional AI methods. Distributed Constraint Optimization Problems (DCOP) have been adapted to capture the locality of interactions among agents with a small number of neighbors [4], but it fails to capture the uncertainty factor in the general problem. Network Distributed POMDPs have been proposed to address these issues [11]. Unfortunately, they solve only small problems.

Decision-theoretic planning can also be used to model our problem [8]. In this model, execution monitoring of the system and replanning are very important. Conditional plans are generated in order to deal with contingencies during execution. Replanning is invoked when an agent identifies unsatisfied, or likely to fail conditions. However, the requirements for a rich model for actions and time are generally problematic for planning techniques based on MDP, POMDP, SAT, CSP, planning graphs or state space encodings [3]. Finally, scalability is also an issue given the size of the problem and the number of potential contingencies. Our approach tries to alleviate these problems by being proactive, and focusing on activities with high probability of failure. Some other techniques that follow similar reasoning are *Just-In-Case* (JIC) contingency scheduling [6] and Mahinur [12], but they focus in a centralized, single-agent solution.

7. CONCLUSION

The keys concepts or profile-based coordination are the assignment of agents to nodes in a network representation of the team reward function and generating appropriate profile metrics that are computable and communicable quickly to create an information flow that has significant and immediate impact on policy modification. There are several directions for future research involving open questions in our approach including (i) optimal assignment of reward nodes to agents, (ii) better methods for evaluating the window effect, and (iii) automated determination for thresholds for which multi-agent policy modification is triggered. In addition, we developed our methodology on the premise that preventing failure (instances of zero quality at the root node of the reward network) was the key to maximizing quality. We are moving towards techniques that use root node quality maximization as the direct objective function. Despite the limitations in the aforementioned areas, our approach outperformed extensions of traditional approaches, especially in the most difficult regimes that were of most interest. We hope to develop profile-based coordination as a promising new direction for solving problems in this challenging setting of real-time multi-agent coordination under uncertainty and sequential decision-making, partial state information, incomplete policy knowledge, subjective views of the team reward function and environmental instability.

8. ADDITIONAL AUTHORS

Pedro Szekely, Information Sciences Institute, University of Southern California, 4676 Admiralty Way #1001, Marina Del Rey, CA, USA.

9. REFERENCES

[1] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:819–840, 2002.

[2] A. Beynier and A. Mouaddib. A polynomial algorithm for decentralized Markov decision processes with temporal constraints. In *AAMAS*, 2005.

[3] J. Bresina, R. Dearden, N. Meuleau, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *UAI*, 2002.

[4] J. Cox, E. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS)*, pages 821–827, 2005.

[5] K. Decker and V. Lesser. Quantitative modeling of complex computational task environments. In *AAAI*, pages 217–224, 1993.

[6] M. Drummond, J. Bresina, and K. Swanson. Just-in-case scheduling. In *AAAI*, pages 1098–1104, 1994.

[7] C. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS*, pages 137–144, 2003.

[8] M. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *JAIR*, 9:1–36, 1998.

[9] P. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 161:149–180, 2005.

[10] D. Musliner, E. Durfee, R. Goldman, M. Boddy, J. Wu, and D. Dolgov. Coordinated plan management using multiagent MDPs. In *2006 AAAI Spring Symposium on Distributed Plan and Schedule Management*, 2006.

[11] R. Nair, P. Varakantham, M.'Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAMAS*, 2005.

[12] N. Onder and M. Pollack. Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms. In *AAAI*, pages 577–584, 1999.

[13] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, pages 389–423, 2002.

[14] J. Shen, R. Becker, and V. Lesser. Agent interaction in distributed MDPs and its implications on complexity. In *AAMAS*, 2006.

[15] S. F. Smith, A. Gallagher, T. Zimmerman, L. Barbulescu, and Z. Rubinstein. Multi-agent management of joint schedules. In *2006 AAAI Spring Symposium on Distributed Plan and Schedule Management*, 2006.