Combinatorial Resource Scheduling for Multiagent MDPs

Dmitri A. Dolgov, Michael R. James, and Michael E. Samples
AI and Robotics Group
Technical Research, Toyota Technical Center USA
{ddolgov, michael.r.james, michael.samples}@gmail.com

ABSTRACT

Optimal resource scheduling in multiagent systems is a computationally challenging task, particularly when the values of resources are not additive. We consider the combinatorial problem of scheduling the usage of multiple resources among agents that operate in stochastic environments, modeled as Markov decision processes (MDPs). In recent years, efficient resource-allocation algorithms have been developed for agents with resource values induced by MDPs. However, this prior work has focused on static resource-allocation problems where resources are distributed once and then utilized in infinite-horizon MDPs. We extend those existing models to the problem of combinatorial resource scheduling, where agents persist only for finite periods between their (predefined) arrival and departure times, requiring resources only for those time periods. We provide a computationally efficient procedure for computing globally optimal resource assignments to agents over time. We illustrate and empirically analyze the method in the context of a stochastic job-scheduling domain.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Design

Keywords

Task and resource allocation in agent systems, Multiagent planning.

1. INTRODUCTION

The tasks of optimal resource allocation and scheduling are ubiquitous in multiagent systems, but solving such optimization problems can be computationally difficult, due to a number of factors. In particular, when the value of a set of resources to an agent is not additive (as is often the case with resources that are substitutes or complements), the utility function might have to be defined on an exponentially large space of resource bundles, which very quickly becomes computationally intractable. Further, even when each agent has a utility function that is nonzero only on a small subset of the possible resource bundles, obtaining optimal allocation is still computationally prohibitive, as the problem becomes NP-complete [14].

Such computational issues have recently spawned several threads of work in using compact models of agents' preferences. One idea is to use any structure present in utility functions to represent them compactly, via, for example, logical formulas [15, 10, 4, 3]. An alternative is to directly model the mechanisms that define the agents' utility functions and perform resource allocation directly with these models [9]. A way of accomplishing this is to model the processes by which an agent might utilize the resources and define the utility function as the payoff of these processes. In particular, if an agent uses resources to act in a stochastic environment, its utility function can be naturally modeled with a Markov decision process, whose action set is parameterized by the available resources. This representation can then be used to construct very efficient resource-allocation algorithms that lead to an exponential speedup over a straightforward optimization problem with flat representations of combinatorial preferences [6, 7, 8].

However, this existing work on resource allocation with preferences induced by resource-parameterized MDPs makes an assumption that the resources are only allocated once and are then utilized by the agents independently within their infinite-horizon MDPs. This assumption that no reallocation of resources is possible can be limiting in domains where agents arrive and depart dynamically.

In this paper, we extend the work on resource allocation under MDP-induced preferences to discrete-time *scheduling* problems, where agents are present in the system for finite time intervals and can only use resources within these intervals. In particular, agents arrive and depart at arbitrary (predefined) times and within these intervals use resources to execute tasks in finite-horizon MDPs. We address the problem of globally optimal resource scheduling, where the objective is to find an allocation of resources to the agents across time that maximizes the sum of the expected rewards that they obtain.

In this context, our main contribution is a mixed-integer-programming formulation of the scheduling problem that chooses globally optimal resource assignments, starting times, and execution horizons for all agents (within their arrival-

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS)

departure intervals). We analyze and empirically compare two flavors of the scheduling problem: one, where agents have static resource assignments within their finite-horizon MDPs, and another, where resources can be dynamically reallocated between agents at every time step.

In the rest of the paper, we first lay down the necessary groundwork in Section 2 and then introduce our model and formal problem statement in Section 3. In Section 4.2, we describe our main result, the optimization program for globally optimal resource scheduling. Following the discussion of our experimental results on a job-scheduling problem in Section 5, we conclude in Section 6 with a discussion of possible extensions and generalizations of our method.

2. BACKGROUND

Similarly to the model used in previous work on resource-allocation with MDP-induced preferences [6, 7], we define the value of a set of resources to an agent as the value of the best MDP policy that is realizable, given those resources. However, since the focus of our work is on scheduling problems, and a large part of the optimization problem is to decide how resources are allocated in time among agents with finite arrival and departure times, we model the agents' planning problems as finite-horizon MDPs, in contrast to previous work that used infinite-horizon discounted MDPs.

In the rest of this section, we first introduce some necessary background on finite-horizon MDPs and present a linear-programming formulation that serves as the basis for our solution algorithm developed in Section 4. We also outline the standard methods for combinatorial resource scheduling with flat resource values, which serve as a comparison benchmark for the new model developed here.

2.1 Markov Decision Processes

A stationary, finite-domain, discrete-time MDP (see, for example, [13] for a thorough and detailed development) can be described as $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$, where: $\mathcal{S}$ is a finite set of system states; $\mathcal{A}$ is a finite set of actions that are available to the agent; p is a stationary stochastic transition function, where $p(\sigma|s,a)$ is the probability of transitioning to state σ upon executing action a in state s; r is a stationary reward function, where $r(s,a)$ specifies the reward obtained upon executing action a in state s.

Given such an MDP, a decision problem under a finite horizon T is to choose an optimal action at every time step to maximize the expected value of the total reward accrued during the agent's (finite) lifetime. The agent's optimal policy is then a function of current state s and the time until the horizon. An optimal policy for such a problem is to act greedily with respect to the optimal value function, defined recursively by the following system of finite-time Bellman equations [2]:

$$v(s,t) = \max_a r(s,a) + \sum_\sigma p(\sigma|s,a)v(\sigma,t+1), \qquad \forall s \in \mathcal{S}, t \in [1, T-1];$$
$$v(s,T) = 0, \qquad \forall s \in \mathcal{S};$$

where $v(s,t)$ is the optimal value of being in state s at time $t \in [1,T]$.

This optimal value function can be easily computed using dynamic programming, leading to the following optimal policy π, where $\pi(s,a,t)$ is the probability of executing action a in state s at time t:

$$\pi(s,a,t) = \begin{cases} 1, & a = \text{argmax}_a\, r(s,a) + \sum_\sigma p(\sigma|s,a)v(\sigma,t+1), \\ 0, & \text{otherwise.} \end{cases}$$

The above is the most common way of computing the optimal value function (and therefore an optimal policy) for a finite-horizon MDP. However, we can also formulate the problem as the following linear program (similarly to the dual LP for infinite-horizon discounted MDPs [13, 6, 7]):

$$\max \sum_s \sum_a r(s,a) \sum_t x(s,a,t)$$

subject to:

$$\sum_a x(\sigma,a,t+1) = \sum_{s,a} p(\sigma|s,a)x(s,a,t) \quad \forall \sigma, t \in [1, T-1];$$
$$\sum_a x(s,a,1) = \alpha(s), \qquad \forall s \in \mathcal{S}; \tag{1}$$

where $\alpha(s)$ is the initial distribution over the state space, and x is the (non-stationary) *occupation measure* ($x(s,a,t) \in [0,1]$ is the total expected number of times action a is executed in state s at time t). An optimal (non-stationary) policy is obtained from the occupation measure as follows:

$$\pi(s,a,t) = x(s,a,t) / \sum_a x(s,a,t) \quad \forall s \in \mathcal{S}, t \in [1,T]. \tag{2}$$

Note that the standard unconstrained finite-horizon MDP, as described above, always has a *uniformly-optimal* solution (optimal for any initial distribution $\alpha(s)$). Therefore, an optimal policy can be obtained by using an arbitrary constant $\alpha(s) > 0$ (in particular, $\alpha(s) = 1$ will result in $x(s,a,t) = \pi(s,a,t)$).

However, for MDPs with resource constraints (as defined below in Section 3), uniformly-optimal policies do not in general exist. In such cases, α becomes a part of the problem input, and a resulting policy is only optimal for that particular α. This result is well known for infinite-horizon MDPs with various types of constraints [1, 6], and it also holds for our finite-horizon model, which can be easily established via a line of reasoning completely analogous to the arguments in [6].

2.2 Combinatorial Resource Scheduling

A straightforward approach to resource scheduling for a set of agents $\mathcal{M}$, whose values for the resources are induced by stochastic planning problems (in our case, finite-horizon MDPs) would be to have each agent enumerate all possible resource assignments over time and, for each one, compute its value by solving the corresponding MDP. Then, each agent would provide valuations for each possible resource bundle over time to a centralized coordinator, who would compute the optimal resource assignments across time based on these valuations.

When resources can be allocated at different times to different agents, each agent must submit valuations for every combination of possible time horizons. Let each agent $m \in \mathcal{M}$ execute its MDP within the arrival-departure time interval $\tau \in [\tau_m^a, \tau_m^d]$. Hence, agent m will execute an MDP with time horizon no greater than $T_m = \tau_m^d - \tau_m^a + 1$. Let $\widehat{\tau}$ be the global time horizon for the problem, before which all of the agents' MDPs must finish. We assume $\tau_m^d < \widehat{\tau}, \forall m \in \mathcal{M}$.

For the scheduling problem where agents have static resource requirements within their finite-horizon MDPs, the agents provide a valuation for each resource bundle for each possible time horizon (from $[1, T_m]$) that they may use. Let Ω be the set of resources to be allocated among the agents. An agent will get at most one resource bundle for one of the time horizons. Let the variable $\psi \in \Psi_m$ enumerate all possible pairs of resource bundles and time horizons for agent m, so there are $2^{|\Omega|} \times T_m$ values for ψ (the space of bundles is exponential in the number of resource types $|\Omega|$).

The agent m must provide a value v_m^ψ for each ψ, and the coordinator will allocate at most one ψ (resource, time horizon) pair to each agent. This allocation is expressed as an indicator variable $z_m^\psi \in \{0, 1\}$ that shows whether ψ is assigned to agent m. For time τ and resource ω, the function $n_m(\psi, \tau, \omega) \in \{0, 1\}$ indicates whether the bundle in ψ uses resource ω at time τ (we make the assumption that agents have binary resource requirements). This allocation problem is NP-complete, even when considering only a single time step, and its difficulty increases significantly with multiple time steps because of the increasing number of values of ψ.

The problem of finding an optimal allocation that satisfies the global constraint that the amount of each resource ω allocated to all agents does not exceed the available amount $\widehat{\varphi}(\omega)$ can be expressed as the following integer program:

$$\max \sum_{m \in \mathcal{M}} \sum_{\psi \in \Psi_m} z_m^\psi v_m^\psi$$

subject to:

$$\sum_{\psi \in \Psi_m} z_m^\psi \leq 1, \qquad \forall m \in \mathcal{M};$$

$$\sum_{m \in \mathcal{M}} \sum_{\psi \in \Psi_m} z_m^\psi n_m(\psi, \tau, \omega) \leq \widehat{\varphi}(\omega), \quad \forall \tau \in [1, \widehat{\tau}], \forall \omega \in \Omega; \tag{3}$$

The first constraint in equation 3 says that no agent can receive more than one bundle, and the second constraint ensures that the total assignment of resource ω does not, at any time, exceed the resource bound.

For the scheduling problem where the agents are able to dynamically reallocate resources, each agent must specify a value for every combination of bundles and time steps within its time horizon. Let the variable $\psi \in \Psi_m$ in this case enumerate all possible resource bundles for which at most one bundle may be assigned to agent m at each time step. Therefore, in this case there are $\sum_{t \in [1, T_m]} (2^{|\Omega|})^t \sim 2^{|\Omega| T_m}$ possibilities of resource bundles assigned to different time slots, for the T_m different time horizons.

The same set of equations (3) can be used to solve this dynamic scheduling problem, but the integer program is different because of the difference in how ψ is defined. In this case, the number of ψ values is exponential in each agent's planning horizon T_m, resulting in a much larger program.

This straightforward approach to solving both of these scheduling problems requires an enumeration and solution of either $2^{|\Omega|} T_m$ (static allocation) or $\sum_{t \in [1, T_m]} 2^{|\Omega| t}$ (dynamic reallocation) MDPs for each agent, which very quickly becomes intractable with the growth of the number of resources $|\Omega|$ or the time horizon T_m.

3. MODEL AND PROBLEM STATEMENT

We now formally introduce our model of the resource-scheduling problem. The problem input consists of the following components:

- $\mathcal{M}, \Omega, \widehat{\varphi}, \tau_m^a, \tau_m^d, \widehat{\tau}$ are as defined above in Section 2.2.
- $\{\Theta_m\} = \{\mathcal{S}, \mathcal{A}, p_m, r_m, \alpha_m\}$ are the MDPs of all agents $m \in \mathcal{M}$. Without loss of generality, we assume that state and action spaces of all agents are the same, but each has its own transition function p_m, reward function r_m, and initial conditions α_m.
- $\varphi_m : \mathcal{A} \times \Omega \mapsto \{0, 1\}$ is the mapping of actions to resources for agent m. $\varphi_m(a, \omega)$ indicates whether action a of agent m needs resource ω. An agent m that receives a set of resources that does not include resource ω cannot execute in its MDP policy any action a for which $\varphi_m(a, \omega) \neq 0$. We assume all resource requirements are binary; as discussed below in Section 6, this assumption is not limiting.

Given the above input, the optimization problem we consider is to find the globally optimal—maximizing the sum of expected rewards—mapping of resources to agents for all time steps: $\Delta : \tau \times \mathcal{M} \times \Omega \mapsto \{0, 1\}$. A solution is feasible if the corresponding assignment of resources to the agents does not violate the global resource constraint:

$$\sum_m \Delta_m(\tau, \omega) \leq \widehat{\varphi}(\omega), \ \forall \omega \in \Omega, \tau \in [1, \widehat{\tau}]. \tag{4}$$

We consider two flavors of the resource-scheduling problem. The first formulation restricts resource assignments to the space where the allocation of resources to each agent is static during the agent's lifetime. The second formulation allows reassignment of resources between agents at every time step within their lifetimes.

Figure 1 depicts a resource-scheduling problem with three agents $\mathcal{M} = \{m_1, m_2, m_3\}$, three resources $\Omega = \{\omega_1, \omega_2, \omega_3\}$, and a global problem horizon of $\widehat{\tau} = 11$. The agents' arrival and departure times are shown as gray boxes and are $\{1, 6\}$, $\{3, 7\}$, and $\{2, 11\}$, respectively. A solution to this problem is shown via horizontal bars within each agents' box, where the bars correspond to the allocation of the three resource types. Figure 1a shows a solution to a static scheduling problem. According to the shown solution, agent m_1 begins the execution of its MDP at time $\tau = 1$ and has a lock on all three resources until it finishes execution at time $\tau = 3$. Note that agent m_1 relinquishes its hold on the resources before its announced departure time of $\tau_{m_1}^d = 6$, ostensibly because other agents can utilize the resources more effectively. Thus, at time $\tau = 4$, resources ω_1 and ω_3 are allocated to agent m_2, who then uses them to execute its MDP (using only actions supported by resources ω_1 and ω_3) until time $\tau = 7$. Agent m_3 holds resource ω_3 during the interval $\tau \in [4, 10]$.

Figure 1b shows a possible solution to the dynamic version of the same problem. There, resources can be reallocated between agents at every time step. For example, agent m_1 gives up its use of resource ω_2 at time $\tau = 2$, although it continues the execution of its MDP until time $\tau = 6$. Notice that an agent is not allowed to stop and restart its MDP, so agent m_1 is only able to continue executing in the interval $\tau \in [3, 4]$ if it has actions that do not require any resources ($\varphi_m(a, \omega) = 0$).

Clearly, the model and problem statement described above make a number of assumptions about the problem and the desired solution properties. We discuss some of those assumptions and their implications in Section 6.

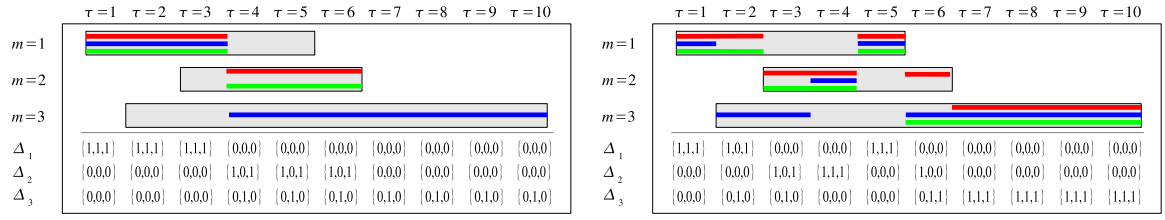

Figure 1: Illustration of a solution to a resource-scheduling problem with three agents and three resources: a) static resource assignments (resource assignments are constant within agents' lifetimes; b) dynamic assignment (resource assignments are allowed to change at every time step).

4. RESOURCE SCHEDULING

Our resource-scheduling algorithm proceeds in two stages. First, we perform a preprocessing step that augments the agent MDPs; this process is described in Section 4.1. Second, using these augmented MDPs we construct a global optimization problem, which is described in Section 4.2.

4.1 Augmenting Agents' MDPs

In the model described in the previous section, we assume that if an agent does not possess the necessary resources to perform actions in its MDP, its execution is halted and the agent leaves the system. In other words, the MDPs cannot be "paused" and "resumed". For example, in the problem shown in Figure 1a, agent m_1 releases all resources after time $\tau = 3$, at which point the execution of its MDP is halted. Similarly, agents m_2 and m_3 only execute their MDPs in the intervals $\tau \in [4, 6]$ and $\tau \in [4, 10]$, respectively. Therefore, an important part of the global decision-making problem is to decide the window of time during which each of the agents is "active" (i.e., executing its MDP).

To accomplish this, we augment each agent's MDP with two new states ("start" and "finish" states s^b, s^f, respectively) and a new "start/stop" action a^*, as illustrated in Figure 2. The idea is that an agent stays in the start state s^b until it is ready to execute its MDP, at which point it performs the start/stop action a^* and transitions into the state space of the original MDP with the transition probability that corresponds to the original initial distribution $\alpha(s)$. For example, in Figure 1a, for agent m_2 this would happen at time $\tau = 4$. Once the agent gets to the end of its activity window (time $\tau = 6$ for agent m_2 in Figure 1a), it performs the start/stop action, which takes it into the sink finish state s^f at time $\tau = 7$.

More precisely, given an MDP $\langle \mathcal{S}, \mathcal{A}, p_m, r_m, \alpha_m \rangle$, we define an augmented MDP $\langle \mathcal{S}', \mathcal{A}', p'_m, r'_m, \alpha'_m \rangle$ as follows:

$$\mathcal{S}' = \mathcal{S} \cup s^b \cup s^f; \qquad \mathcal{A}' = \mathcal{A} \cup a^*;$$
$$p'(s|s^b, a^*) = \alpha(s), \ \forall s \in \mathcal{S}; \qquad p'(s^b|s^b, a) = 1.0, \ \ \forall a \in \mathcal{A};$$
$$p'(s^f|s, a^*) = 1.0, \ \ \forall s \in \mathcal{S};$$
$$p'(\sigma|s, a) = p(\sigma|s, a), \ \ \forall s, \sigma \in \mathcal{S}, a \in \mathcal{A};$$
$$r'(s^b, a) = r'(s^f, a) = 0, \ \ \forall a \in \mathcal{A}';$$
$$r'(s, a) = r(s, a), \ \ \forall s \in \mathcal{S}, a \in \mathcal{A};$$
$$\alpha'(s^b) = 1; \quad \alpha'(s) = 0, \ \forall s \in \mathcal{S};$$

where all non-specified transition probabilities are assumed to be zero. Further, in order to account for the new starting state, we begin the MDP one time-step earlier, setting $\tau^a_m \leftarrow \tau^a_m - 1$. This will not affect the resource allocation due to the resource constraints only being enforced for the original MDP states, as will be discussed in the next section. For example, the augmented MDPs shown in Figure 2b (which starts in state s^b at time $\tau = 2$) would be constructed from an MDP with original arrival time $\tau = 3$. Figure 2b also shows a sample trajectory through the state space: the agent starts in state s^b, transitions into the state space $\mathcal{S}$ of the original MDP, and finally exists into the sink state s^f.

Note that if we wanted to model a problem where agents could pause their MDPs at arbitrary time steps (which might be useful for domains where dynamic reallocation is possible), we could easily accomplish this by including an extra action that transitions from each state to itself with zero reward.

4.2 MILP for Resource Scheduling

Given a set of augmented MDPs, as defined above, the goal of this section is to formulate a global optimization program that solves the resource-scheduling problem. In this section and below, all MDPs are assumed to be the augmented MDPs as defined in Section 4.1.

Our approach is similar to the idea used in [6]: we begin with the linear-program formulation of agents' MDPs (1) and augment it with constraints that ensure that the corresponding resource allocation across agents and time is valid. The resulting optimization problem then simultaneously solves the agents' MDPs and resource-scheduling problems. In the rest of this section, we incrementally develop a mixed integer program (MILP) that achieves this.

In the absence of resource constraints, the agents' finite-horizon MDPs are completely independent, and the globally optimal solution can be trivially obtained via the following LP, which is simply an aggregation of single-agent finite-horizon LPs:

$$\max \sum_m \sum_s \sum_a r_m(s, a) \sum_t x_m(s, a, t)$$

subject to:

$$\sum_a x_m(\sigma, a, t+1) = \sum_{s,a} p_m(\sigma|s, a) x_m(s, a, t), \qquad \forall m \in \mathcal{M}, \sigma \in \mathcal{S}, t \in [1, T_m - 1];$$
$$\sum_a x_m(s, a, 1) = \alpha_m(s), \qquad \forall m \in \mathcal{M}, s \in \mathcal{S}; \tag{12}$$

where $x_m(s, a, t)$ is the occupation measure of agent m, and

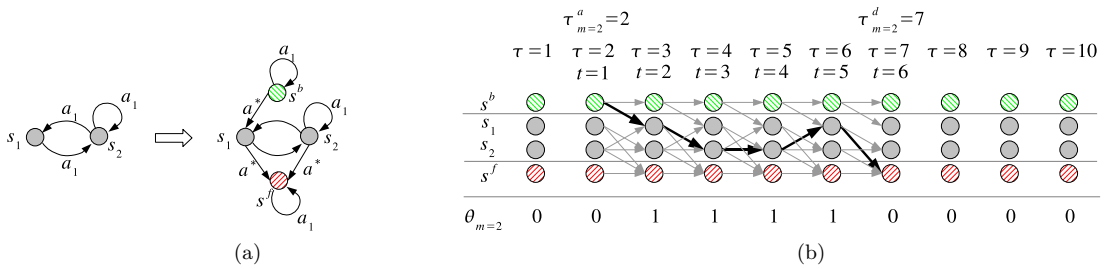

Figure 2: Illustration of augmenting an MDP to allow for variable starting and stopping times: a) (left) the original two-state MDP with a single action; (right) the augmented MDP with new states s^b and s^f and the new action a^* (note that the origianl transitions are not changed in the augmentation process); b) the augmented MDP displayed as a trajectory through time (grey lines indicate all transitions, while black lines indicate a given trajectory.

Objective Function (sum of expected rewards over all agents)		$\max \sum_m \sum_s \sum_a r_m(s,a) \sum_t x_m(s,a,t)$ (5)
Meaning	*Implication*	*Linear Constraints*
Tie x to θ. Agent is only active when occupation measure is nonzero in original MDP states.	$\theta_m(\tau)=0 \implies x_m(s,a,\tau-\tau^a_m+1)=0$ $\forall s \notin \{s^b,s^f\}, a \in \mathcal{A}$	$\sum_{s\notin\{s^b,s^f\}} \sum_a x_m(s,a,t) \le \theta_m(\tau^a_m+t-1)$ $\forall m \in \mathcal{M}, \forall t \in [1,T_m]$ (6)
Agent can only be active in $\tau \in (\tau^a_m,\tau^d_m)$		$\theta_m(\tau)=0 \quad \forall m \in \mathcal{M}, \tau \notin (\tau^a_m,\tau^d_m)$ (7)
Cannot use resources when not active	$\theta_m(\tau)=0 \implies \Delta_m(\tau,\omega)=0$ $\forall \tau \in [0,\widehat{\tau}], \omega \in \Omega$	$\Delta_m(\tau,\omega) \le \theta_m(\tau) \qquad \forall m \in \mathcal{M}, \tau \in [0,\widehat{\tau}], \omega \in \Omega$ (8)
Tie x to Δ (nonzero x forces corresponding Δ to be nonzero.)	$\Delta_m(\tau,\omega)=0, \varphi_m(a,\omega)=1 \implies$ $x_m(s,a,\tau-\tau^a_m+1)=0$ $\forall s \notin \{s^b,s^f\}$	$1/\lvert\mathcal{A}\rvert \sum_a \varphi_m(a,\omega) \sum_{s\notin\{s^b,s^f\}} x_m(s,a,t) \le \Delta_m(t+\tau^a_m-1,\omega)$ $\forall m \in \mathcal{M}, \omega \in \Omega, t \in [1,T_m]$ (9)
Resource bounds		$\sum_m \Delta_m(\tau,\omega) \le \widehat{\varphi}(\omega) \qquad \forall \omega \in \Omega, \tau \in [0,\widehat{\tau}]$ (10)
Agent cannot change resources while active. Only enabled for scheduling with static assignments.	$\theta_m(\tau)=1$ and $\theta_m(\tau+1)=1 \implies$ $\Delta_m(\tau,\omega)=\Delta_m(\tau+1,\omega)$	$\Delta_m(\tau,\omega) - Z(1-\theta_m(\tau+1)) \le$ $\Delta_m(\tau+1,\omega) + Z(1-\theta_m(\tau))$ $\Delta_m(\tau,\omega) + Z(1-\theta_m(\tau+1)) \ge$ $\Delta_m(\tau+1,\omega) - Z(1-\theta_m(\tau))$ $\forall m \in \mathcal{M}, \omega \in \Omega, \tau \in [0,\widehat{\tau}]$ (11)

Table 1: MILP for globally optimal resource scheduling.

$T_m = \tau^d_m - \tau^a_m + 1$ is the time horizon for the agent's MDP.

Using this LP as a basis, we augment it with constraints that ensure that the resource usage implied by the agents' occupation measures $\{x_m\}$ does not violate the global resource requirements $\widehat{\varphi}$ at any time step $\tau \in [0,\widehat{\tau}]$. To formulate these resource constraints, we use the following binary variables:

- $\Delta_m(\tau,\omega) = \{0,1\}$, $\forall m \in \mathcal{M}, \tau \in [0,\widehat{\tau}], \omega \in \Omega$, which serve as indicator variables that define whether agent m possesses resource ω at time τ. These are analogous to the static indicator variables used in the one-shot static resource-allocation problem in [6].
- $\theta_m = \{0,1\}$, $\forall m \in \mathcal{M}, \tau \in [0,\widehat{\tau}]$ are indicator variables that specify whether agent m is "active" (i.e., executing its MDP) at time τ.

The meaning of resource-usage variables Δ is illustrated in Figure 1: $\Delta_m(\tau,\omega) = 1$ only if resource ω is allocated to agent m at time τ. The meaning of the "activity indicators" θ is illustrated in Figure 2b: when agent m is in either the start state s^b or the finish state s^f, the corresponding $\theta_m = 0$, but once the agent becomes active and enters one of the other states, we set $\theta_m = 1$. This meaning of θ can be enforced with a linear constraint that synchronizes the values of the agents' occupation measures x_m and the activity

indicators θ, as shown in (6) in Table 1.

Another constraint we have to add—because the activity indicators θ are defined on the global timeline τ—is to enforce the fact that the agent is inactive outside of its arrival-departure window. This is accomplished by constraint (7) in Table 1.

Furthermore, agents should not be using resources while they are inactive. This constraint can also be enforced via a linear inequality on θ and Δ, as shown in (8).

Constraint (6) sets the value of θ to match the policy defined by the occupation measure x_m. In a similar fashion, we have to make sure that the resource-usage variables Δ are also synchronized with the occupation measure x_m. This is done via constraint (9) in Table 1, which is nearly identical to the analogous constraint from [6].

After implementing the above constraint, which enforces the meaning of Δ, we add a constraint that ensures that the agents' resource usage never exceeds the amounts of available resources. This condition is also trivially expressed as a linear inequality (10) in Table 1.

Finally, for the problem formulation where resource assignments are static during a lifetime of an agent, we add a constraint that ensures that the resource-usage variables Δ do not change their value while the agent is active ($\theta = 1$). This is accomplished via the linear constraint (11), where $Z \geq 2$ is a constant that is used to turn off the constraints when $\theta_m(\tau) = 0$ or $\theta_m(\tau + 1) = 0$. This constraint is not used for the dynamic problem formulation, where resources can be reallocated between agents at every time step.

To summarize, Table 1 together with the conservation-of-flow constraints from (12) defines the MILP that simultaneously computes an optimal resource assignment for all agents across time as well as optimal finite-horizon MDP policies that are valid under that resource assignment.

As a rough measure of the complexity of this MILP, let us consider the number of optimization variables and constraints. Let $T_{\mathcal{M}} = \sum T_m = \sum_m (\tau_m^a - \tau_m^d + 1)$ be the sum of the lengths of the arrival-departure windows across all agents. Then, the number of optimization variables is:

$$T_{\mathcal{M}} + \widehat{\tau}|\mathcal{M}||\Omega| + \widehat{\tau}|\mathcal{M}|,$$

$T_{\mathcal{M}}$ of which are continuous (x_m), and $\widehat{\tau}|\mathcal{M}||\Omega| + \widehat{\tau}|\mathcal{M}|$ are binary (Δ and θ). However, notice that all but $T_{\mathcal{M}}|\mathcal{M}|$ of the θ are set to zero by constraint (7), which also immediately forces all but $T_{\mathcal{M}}|\mathcal{M}||\Omega|$ of the Δ to be zero via the constraints (8). The number of constraints (not including the degenerate constraints in (7)) in the MILP is:

$$T_{\mathcal{M}} + T_{\mathcal{M}}|\Omega| + \widehat{\tau}|\Omega| + \widehat{\tau}|\mathcal{M}||\Omega|.$$

Despite the fact that the complexity of the MILP is, in the worst case, exponential[1] in the number of binary variables, the complexity of this MILP is significantly (exponentially) lower than that of the MILP with flat utility functions, described in Section 2.2. This result echos the efficiency gains reported in [6] for single-shot resource-allocation problems, but is much more pronounced, because of the explosion of the flat utility representation due to the temporal aspect of the problem (recall the prohibitive complexity of the combinatorial optimization in Section 2.2). We empirically analyze the performance of this method in Section 5.

[1]Strictly speaking, solving MILPs to optimality is NP-complete in the number of integer variables.

5. EXPERIMENTAL RESULTS

Although the complexity of solving MILPs is in the worst case exponential in the number of integer variables, there are many efficient methods for solving MILPs that allow our algorithm to scale well for parameters common to resource allocation and scheduling problems. In particular, this section introduces a problem domain—the *repairshop problem*—used to empirically evaluate our algorithm's scalability in terms of the number of agents $|\mathcal{M}|$, the number of shared resources $|\Omega|$, and the varied lengths of global time $\widehat{\tau}$ during which agents may enter and exit the system.

The *repairshop problem* is a simple parameterized MDP adopting the metaphor of a vehicular repair shop. Agents in the repair shop are mechanics with a number of independent tasks that yield reward only when completed. In our MDP model of this system, actions taken to advance through the state space are only allowed if the agent holds certain resources that are publicly available to the shop. These resources are in finite supply, and optimal policies for the shop will determine when each agent may hold the limited resources to take actions and earn individual rewards. Each task to be completed is associated with a single action, although the agent is required to repeat the action numerous times before completing the task and earning a reward.

This model was parameterized in terms of the number of agents in the system, the number of different types of resources that could be linked to necessary actions, a global time during which agents are allowed to arrive and depart, and a maximum length for the number of time steps an agent may remain in the system.

All datapoints in our experiments were obtained with 20 evaluations using CPLEX to solve the MILPs on a Pentium-4 computer with 2Gb of RAM. Trials were conducted on both the static and the dynamic version of the resource-scheduling problem, as defined earlier.

Figure 3 shows the runtime and policy value for independent modifications to the parameter set. The top row shows how the solution time for the MILP scales as we increase the number of agents $|\mathcal{M}|$, the global time horizon $\widehat{\tau}$, and the number of resources $|\Omega|$. Increasing the number of agents leads to exponential complexity scaling, which is to be expected for an NP-complete problem. However, increasing the global time limit $\widehat{\tau}$ or the total number of resource types $|\Omega|$—while holding the number of agents constant—does not lead to decreased performance. This occurs because the problems get easier as they become under-constrained, which is also a common phenomenon for NP-complete problems. We also observe that the solution to the dynamic version of the problem can often be computed much faster than the static version.

The bottom row of Figure 3 shows the joint policy value of the policies that correspond to the computed optimal resource-allocation schedules. We can observe that the dynamic version yields higher reward (as expected, since the reward for the dynamic version is always no less than the reward of the static version). We should point out that these graphs should not be viewed as a measure of performance of two different algorithms (both algorithms produce optimal solutions but to different problems), but rather as observations about how the quality of optimal solutions change as more flexibility is allowed in the reallocation of resources.

Figure 4 shows runtime and policy value for trials in which common input variables are scaled together. This allows

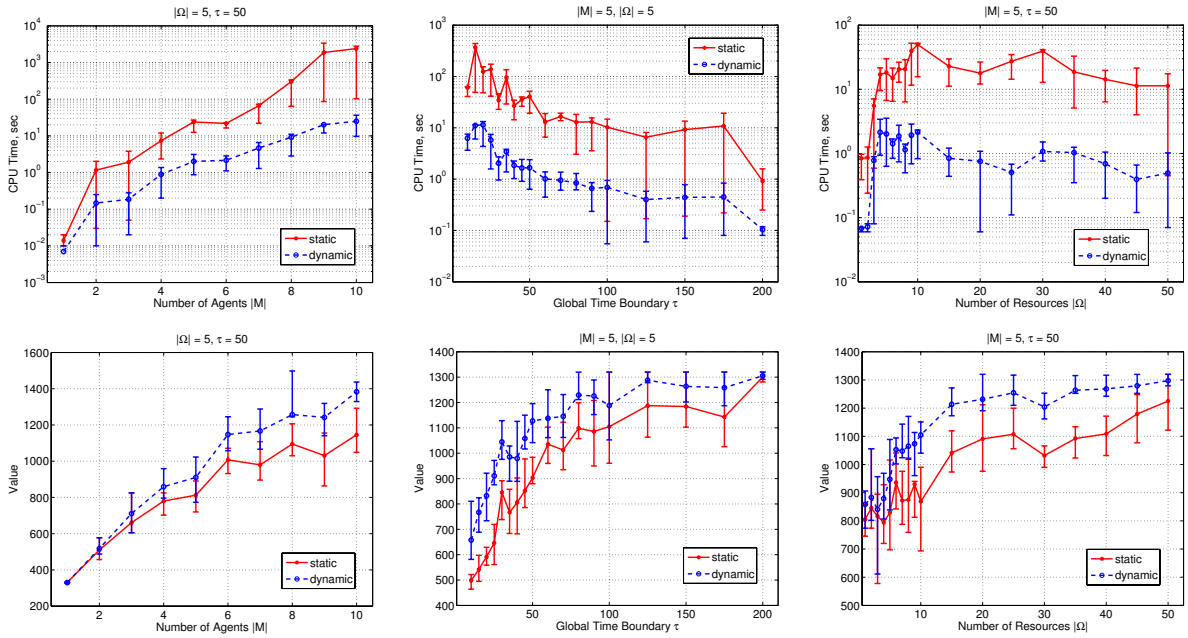

Figure 3: Evaluation of our MILP for variable numbers of agents (column 1), lengths of global-time window (column 2), and numbers of resource types (column 3). Top row shows CPU time, and bottom row shows the joint reward of agents' MDP policies. Error bars show the 1^{st} and 3^{rd} quartiles (25% and 75%).

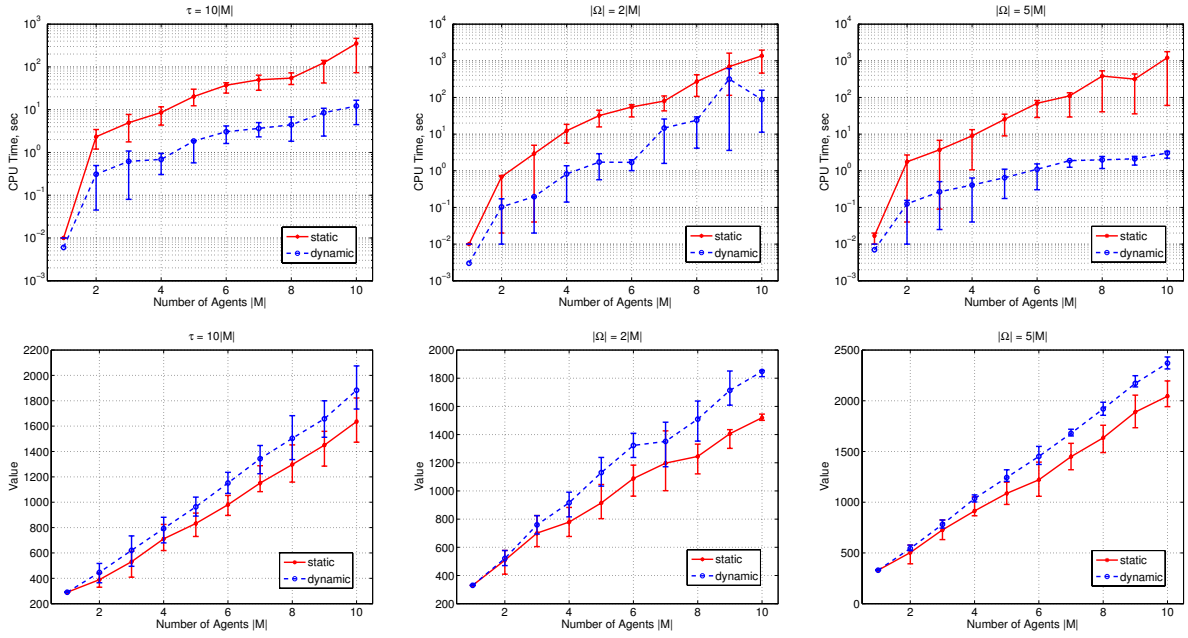

Figure 4: Evaluation of our MILP using correlated input variables. The left column tracks the performance and CPU time as the number of agents and global-time window increase together ($\widehat{\tau} = 10|\mathcal{M}|$). The middle and the right column track the performance and CPU time as the number of resources and the number of agents increase together as $|\Omega| = 2|\mathcal{M}|$ and $|\Omega| = 5|\mathcal{M}|$, respectively. Error bars show the 1^{st} and 3^{rd} quartiles (25% and 75%).

us to explore domains where the total number of agents scales proportionally to the total number of resource types or the global time horizon, while keeping constant the average agent density (per unit of global time) or the average number of resources per agent (which commonly occurs in real-life applications).

Overall, we believe that these experimental results indicate that our MILP formulation can be used to effectively solve resource-scheduling problems of nontrivial size.

6. DISCUSSION AND CONCLUSIONS

Throughout the paper, we have made a number of assumptions in our model and solution algorithm; we discuss their implications below.

- **Continual execution.** We assume that once an agent stops executing its MDP (transitions into state s^f), it exits the system and cannot return. It is easy to relax this assumption for domains where agents' MDPs can be paused and restarted. All that is required is to include an additional "pause" action which transitions from a given state back to itself, and has zero reward.
- **Indifference to start time.** We used a reward model where agents' rewards depend only on the time horizon of their MDPs and not the global start time. This is a consequence of our MDP-augmentation procedure from Section 4.1. It is easy to extend the model so that the agents incur an explicit penalty for idling by assigning a non-zero negative reward to the start state s^b.
- **Binary resource requirements.** For simplicity, we have assumed that resource costs are binary: $\varphi_m(a, \omega) = \{0, 1\}$, but our results generalize in a straightforward manner to non-binary resource mappings, analogously to the procedure used in [5].
- **Cooperative agents.** The optimization procedure discussed in this paper was developed in the context of cooperative agents, but it can also be used to design a mechanism for scheduling resources among self-interested agents. This optimization procedure can be embedded in a Vickrey-Clarke-Groves auction, completely analogously to the way it was done in [7]. In fact, all the results of [7] about the properties of the auction and information privacy directly carry over to the scheduling domain discussed in this paper, requiring only slight modifications to deal with finite-horizon MDPs.
- **Known, deterministic arrival and departure times**. Finally, we have assumed that agents' arrival and departure times (τ_m^a and τ_m^d) are deterministic and known *a priori*. This assumption is fundamental to our solution method. While there are many domains where this assumption is valid, in many cases agents arrive and depart dynamically and their arrival and departure times can only be predicted probabilistically, leading to online resource-allocation problems. In particular, in the case of self-interested agents, this becomes an interesting version of an *online-mechanism-design* problem [11, 12].

In summary, we have presented an MILP formulation for the combinatorial resource-scheduling problem where agents' values for possible resource assignments are defined by finite-horizon MDPs. This result extends previous work ([6, 7]) on static one-shot resource allocation under MDP-induced preferences to resource-scheduling problems with a temporal aspect. As such, this work takes a step in the direction of designing an online mechanism for agents with combinatorial resource preferences induced by stochastic planning problems. Relaxing the assumption about deterministic arrival and departure times of the agents is a focus of our future work.

We would like to thank the anonymous reviewers for their insightful comments and suggestions.

7. REFERENCES

[1] E. Altman and A. Shwartz. Adaptive control of constrained Markov chains: Criteria and policies. *Annals of Operations Research, special issue on Markov Decision Processes*, 28:101–134, 1991.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[3] C. Boutilier. Solving concisely expressed combinatorial auction problems. In *Proc. of AAAI-02*, pages 359–366, 2002.

[4] C. Boutilier and H. H. Hoos. Bidding languages for combinatorial auctions. In *Proc. of IJCAI-01*, pages 1211–1217, 2001.

[5] D. Dolgov. *Integrated Resource Allocation and Planning in Stochastic Multiagent Environments*. PhD thesis, Computer Science Department, University of Michigan, February 2006.

[6] D. A. Dolgov and E. H. Durfee. Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proc. of ICAPS-04*, pages 315–324, June 2004.

[7] D. A. Dolgov and E. H. Durfee. Computationally efficient combinatorial auctions for resource allocation in weakly-coupled MDPs. In *Proc. of AAMAS-05*, New York, NY, USA, 2005. ACM Press.

[8] D. A. Dolgov and E. H. Durfee. Resource allocation among agents with preferences induced by factored MDPs. In *Proc. of AAMAS-06*, 2006.

[9] K. Larson and T. Sandholm. Mechanism design and deliberative agents. In *Proc. of AAMAS-05*, pages 650–656, New York, NY, USA, 2005. ACM Press.

[10] N. Nisan. Bidding and allocation in combinatorial auctions. In *Electronic Commerce*, 2000.

[11] D. C. Parkes and S. Singh. An MDP-based approach to Online Mechanism Design. In *Proc. of the Seventeenths Annual Conference on Neural Information Processing Systems (NIPS-03)*, 2003.

[12] D. C. Parkes, S. Singh, and D. Yanovsky. Approximately efficient online mechanism design. In *Proc. of the Eighteenths Annual Conference on Neural Information Processing Systems (NIPS-04)*, 2004.

[13] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, New York, 1994.

[14] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[15] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proc. of IJCAI-99*, pages 542–547, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.