Sequential Resource Allocation in Multi-agent Systems with Uncertainties

Jianhui Wu and Edmund H. Durfee
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
jianhuiw, durfee@umich.edu

ABSTRACT

Exchanging scarce resources during execution among a group of agents is one way to improve the overall performance in multi-agent systems with limited shared resources, but implementing optimal sequential resource allocation is often a nontrivial problem in complex systems with uncertainties. In this paper, we present an MILP-based algorithm that can automatically break a large mission into multiple phases and make optimal resource (re)allocations at the entry of each phase. We illustrate our algorithms through several increasingly complex classes of sequential resource allocation problems, and show through experiments that our techniques can increase agents' rewards for varying levels of constraints on resources and constraints on exchanging resources.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE; I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Control Methods, and Search

General Terms

ALGORITHMS

Keywords

Sequential resource allocation, mission phasing, constrained MDPs, mixed integer linear programming

1. MOTIVATION AND INTRODUCTION

In multi-agent systems with scarce resources, an agent often cannot execute some of its possible actions because resources required by those actions are currently held by other agents. How the resources are allocated among the agents will dictate the actions each agent will be capable of performing, and thus how the agents will act and interact to accomplish their goals in their environments.

Even when agents are strictly cooperative, the problem of determining an allocation of resources among agents who are operating in a stochastic world is difficult. Assessing the value of a particular bundle of resources to an agent requires the agent to formulate an optimal policy based only on the actions that the bundle of resources enables. Since the number of resource bundles is exponential, the policy optimization process may have to be solved an exponential number of times. Furthermore, once the agents have computed valuations for each of the bundles, identifying the optimal assignment of bundles to agents (solving the winner determination problem) is NP-complete in the size of the bundle space, leading to a doubly-exponential algorithm. Fortunately, a more efficient approach has been developed that solves the combinatorial optimization (resource allocation) and stochastic optimization (policy formulation) problems simultaneously [7, 6].

In our work, we are interested in extending that approach to allow agents to reallocate the resources among themselves over time. That is, the previous work would generate an optimal allocation of resources assuming that, once the resources were distributed among the agents, the initial allocation would persist throughout the remainder of the agents' execution. We relax that assumption, and instead investigate the implications of allowing agents to swap resources in the midst of execution. This leads to solving a problem of sequential (as opposed to one-shot) allocation, along with the problem of optimizing agents' policies for the execution phases between reallocations.

To solve the sequential multi-agent allocation problem, we extend prior work on the single-agent version of this problem [13]. That work analyzed how a single agent should select a different subset of the available resources at different times, and developed strategies for deciding on the optimal states at which to switch resources, given constraints and/or costs on such states. In this paper, we follow a similar progression to what that work laid out, in terms of giving the agents increasing latitude in designating resource reallocation states, but tackle the considerably harder problem of making the decisions about the states given that the agents need to agree on these states if they are to "meet" (in some sense) to exchange resources.

More formally, the sequential resource allocation problem we study is an optimization problem for maximizing the total expected reward of a group of agents within a finite horizon by sequentially adjusting the distribution of non-consumable resources among the agents. In general, such sequential allocation problems are challenging because the utility of decomposing a problem into phases and the utility of allocating resources for a phase are unknown until

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

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

policies are formulated, but policies cannot be formulated until phases are built and resources are allocated, i.e., decomposition, resource allocation, and policy formulation are strongly interdependent. The key contribution that we make in this paper is to develop, analyze, and empirically evaluate a mixed-integer linear programming (MILP) approach that extends previous MILP techniques for solving the one-shot resource-allocation-and-policy-formulation problem to also solve the problem of optimally decomposing the agents' overall activities into sequential phases.

In keeping with prior work, we focus on loosely-coupled multi-agent systems [6, 9] where agents are coupled through sharing resources (i.e., actions selected for one agent might restrict the actions available to others), but the actions executed by one agent cannot impact rewards and transitions of others. In addition, in this paper we assume that a scheduled resource transfer can always succeed, and thus that reassigning resources that are currently being used by an agent will abort that agent's task. At the end of the paper, our discussion of future work talks about the implications of relaxing this assumption.

The rest of the paper is organized as follows. Section 2 summarizes the concepts behind (constrained) Markov Decision Processes that our work builds on. In Sections 3, 4 and 5, we look at several increasingly general variations of sequential resource allocation problems, and for each, we present, analyze, and illustrate solution algorithms. We evaluate the efficiency and optimality of our techniques in Section 6. Finally, Section 7 summarizes the contributions of the work presented and outlines future directions.

2. MDP AND CONSTRAINED MDP

A classical Markov Decision Process (MDP) can be defined as a four-tuple $\langle S, A, P, R\rangle$, where S is a finite state space, A is a finite action space, $P = \{p_{i,a,j}\}$ is the state transition probability where $p_{i,a,j}$ is the probability that the agent reaches state j if it executes action a in state i, and $R = \{r_{i,a}\}$ is the reward function where $r_{i,a}$ is the reward that the agent receives if it executes action a in state i.

Well-known Bellman backup, value iteration, and policy iteration algorithms have been commonly used in solving unconstrained MDPs [10]. However, it is surprisingly hard to make these algorithms work on constrained problems, and thus many researchers adopt an alternative formulation that is based upon linear programming [1, 2, 5]. In this section, we recap how to formulate an unconstrained MDP into a linear program, whose solution yields the optimal policy maximizing the total expected reward, because our subsequent techniques extend these foundations.

Let α_j denote probability that the agent is initially in state j, and let $x_{i,a}$, often named the *occupation measure*, denote the expected number of times action a is executed in state i. The objective function $\sum_i \sum_a x_{i,a} r_{i,a}$ can then represent the total expected reward. Now, consider the linear program shown in Eq.1.

$$\max \sum_i \sum_a x_{i,a} r_{i,a} \;\Big|\; \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} x_{i,a} \\ x_{i,a} \geq 0 \end{array} \tag{1}$$

where the constraint $\sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} x_{i,a}$ indicates that the expected number of times state j is visited must equal the initial probability distribution at state j plus the expected number of times state j is entered via all possible transitions. It should be noted that Eq.1 (and other relevant formulations that will be presented later) can be used in finite horizon MDPs as well by encoding temporal information in the state representation.

If we solve the linear program in Eq.1, it is trivial to derive the optimal policy that specifies the action(s) to take in a given state. Specifically, assigning a probability of executing action a at state i as $\pi_{i,a} = \frac{x_{i,a}}{\sum_a x_{i,a}}$ will maximize the total expected reward. If any $\pi_{i,a}$ has a value other than zero or one, the optimal policy is randomized; otherwise it is deterministic.

Formulating unconstrained MDPs as linear programs makes it easier to add other constraints. A wide range of such constrained optimization problems have been investigated by Dolgov and Durfee [5, 7, 6]. In order to familiarize readers with some background knowledge, we briefly introduce their solution approach for one-shot resource allocation problems in loosely-coupled multi-agent systems in Eq.2, where $p^m_{i,a,j}$, $r^m_{i,a}$, and α^m_j, respectively, represent state transition probability, reward function, and initial probability distribution of agent m. $x^m_{i,a}$ is the expected number of times agent m executes action a in state i. Δ^m_a, an integer in the interval [0, 1], is used to indicate whether action a can be scheduled in the policy of agent m, and $u^m_{a,o}$ represents the amount of resource o required by agent m to execute action a.

$$\max \sum_m \sum_i \sum_a x^m_{i,a} r^m_{i,a}$$

$$\left|\begin{array}{ll} \textit{Probability conservation:} & \\ x^m_{i,a} \geq 0 & \forall m, \forall i, \forall a \\ \sum_a x^m_{j,a} - \sum_i \sum_a p^m_{i,a,j} x^m_{i,a} = \alpha^m_j & \forall m, \forall j \\ & \\ \textit{Resource constraints:} & \\ \Delta^m_a \in \{0,1\} & \forall m, \forall a \\ \omega^m_o \geq u^m_{a,o} \Delta^m_a & \forall o, \forall m, \forall a \\ \sum_m \omega^m_o = \hat{\omega}_o & \forall o \\ \sum_i x^m_{i,a} \leq X \Delta^m_a & \forall m, \forall a \end{array}\right. \tag{2}$$

The objective function is the sum of accumulative rewards of all agents, i.e., $\sum_m \sum_i \sum_a x^m_{i,a} r^m_{i,a}$, since it is assumed that agents are loosely coupled (i.e., actions taken by one agent cannot impact other agents' rewards and transitions). The constraint $\sum_a x^m_{j,a} - \sum_i \sum_a p^m_{i,a,j} x^m_{i,a} = \alpha^m_j$ guarantees probability conservation at every state of every agent. The constraint $\omega^m_o \geq u^m_{a,o} \Delta^m_a$ says that, to allow agent m to execute action a, the amount of resource o that agent m holds, which is symbolized by ω^m_o, must be no less than $u^m_{a,o}$. The constraint $\sum_m \omega^m_o = \hat{\omega}_o$ guarantees that the total amount of resource o allocated across all agents must equal the amount of available resource o (assuming resource will be completely distributed).[1] Letting X be a constant greater than $\sup \sum_i x^m_{i,a}$ (where in finite horizon MDPs, we can set X to the finite horizon T since any action can be executed by an agent at most T times during one execution), the constraint $\sum_i x^m_{i,a} \leq X \Delta^m_a$ implies that $x^m_{i,a}$ must be zero (i.e., action a cannot be executed by agent m) when $\Delta^m_a = 0$ and $x^m_{i,a}$ is unrestricted otherwise.

In Eq.2, $p^m_{i,a,j}$, $r^m_{i,a}$, α^m_j, $u^m_{a,o}$, $\hat{\omega}_o$, and X are constants while $x^m_{i,a}$ and ω^m_o are continuous variables and Δ^m_a are integer variables, which means that Eq.2 is a mixed integer

[1] This assumption can be easily relaxed by introducing a "dummy" agent to hold unallocated resources.

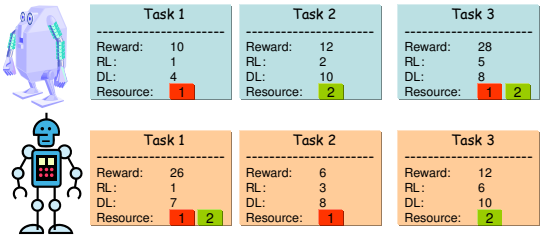

Figure 1: A simple example.

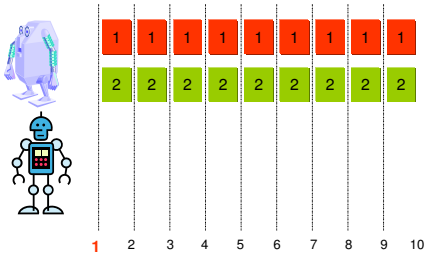

Figure 2: Optimal one-shot resource allocation.

linear program (MILP). Mixed integer linear programming is the discrete version of linear programming with an additional requirement that some of variables must be integers. MILPs can be solved by a variety of highly optimized algorithms and tools [4, 12]. In recent years, there has been substantial progress on using MILPs in automated decision making [3, 8, 11, 14].

We conclude this section by introducing a simple example that we will use over the next several sections of this paper to illustrate the approach just described and our increasingly advanced techniques as we go along. In the example problem, two cooperative agents attempt to maximize their total expected reward within 10 time steps. Each agent has three tasks. At each time step, an agent can choose to continue its previously started task (if there is one and if the agent still holds the required resources), to start a new task (and abort its current task if there is one), or simply to do nothing. In addition, we assume that a task that has previously been aborted (and thus has failed) can be re-tried, but no task can be successfully accomplished more than once.

Figure 1 specifies the detailed information of the tasks in our example problem, including release (RL) time (i.e., when the task becomes available), deadline (DL) (i.e., when the task becomes unavailable), reward, and resource prerequisite. For example, agent *1* can start (or continue) its task *1*, which will incur a reward 10 if accomplished, at any time step within the interval $[1,\ 4)$ given that it has one unit of resource *1* at that time. The uncertainty in this problem is in the time required to execute a task. Here, we say that, if an agent starts a task and does not abort it during execution, then it has 30%, 40%, and 30% probability of accomplishing it within one, two, and three time steps, respectively.

When the resource upper bound $\hat{\omega}_o$ is unlimited, this is an unconstrained MDP. Using Bellman backup or linear programming (i.e., a multi-agent version of Eq.1), we can easily compute the optimal unconstrained policy, which yields the total expected reward 93.64.

Suppose instead that resources are limited, i.e., there is only one unit of resource *1* and one unit of resource *2* in the system at any time step. Now it is not obvious how to distribute these resources. Adopting Eq.2, we can find that the optimal one-shot allocation is to give all resources to agent *1* and let agent *2* idle over the entire execution as shown in Figure 2, and the total expected reward in this case is 49.64, much lower than the reward in the above unconstrained case. In the following sections, we will use this example as we go along to see the degree to which sequential allocation of resources can improve the expected reward in environments with limited shared resources.

3. FIXED SCHEDULE OF SEQUENTIAL RESOURCE ALLOCATION

In many application domains, the opportunities to reallocate resources are dictated by the state of the world rather than being choices of agents. We begin our examination of sequential resource allocation techniques from this simple case where the times at which resources can be redistributed are dictated *a priori*. We will investigate more general sequential allocation problems in Sections 4 and 5.

With predetermined resource reallocation opportunities, the problem can be decomposed into multiple smaller subproblems, which we call *phases*, where each phase starts at a time when resources are allocated and lasts until the allocation alters again (and agents enter the subsequent phase).

Not surprisingly, each phase can be represented as a one-shot resource allocation problem. However, computing an executable policy for each phase independently (using Eq.2) and then piecing these local policies together for a global policy is, in general, not a practicable solution for our sequential allocation problems. The underlying reason is that Eq.2 requires initial probability distribution α_j^m to be known *a priori*; however, α_j^m of a phase generally depends on the policy of the previous phase. In addition, note that the number of possible policies is $|A|^{|S|}$ where $|A|$ and $|S|$ are the sizes of action and state spaces respectively. Hence, enumerating possible combinations of phase policies to find an optimal global policy is only applicable for simple problems with very small state and action spaces.

In this work, we propose an alternative solution which is directed at addressing this shortcoming. Rather than dealing with each phase independently, we link them together through the relationship of probability conservation, and manage resources by imposing resource constraints on each phase instead of on the entire execution. The details are shown in the MILP Eq.3.

$$\max \sum_m \sum_i \sum_a x_{i,a}^m r_{i,a}^m$$

$$\begin{array}{ll}
\textit{Probability conservation:} & \\
x_{i,a}^m \geq 0 & \forall m, \forall i, \forall a \\
\sum_a x_{j,a}^m - \sum_i \sum_a p_{i,a,j}^m x_{i,a}^m = \alpha_j^m & \forall m, \forall j \\
\textit{Resource constraints:} & \\
\Delta_a^{m,k} \in \{0,1\} & \forall k, \forall m, \forall a \\
\omega_o^{m,k} \geq u_{a,o}^m \Delta_a^{m,k} & \forall o, \forall k, \forall m, \forall a \\
\sum_m \omega_o^{m,k} = \hat{\omega}_o & \forall o, \forall k \\
\sum_{i \in \mathbb{S}^k} x_{i,a}^m \leq T \Delta_a^{m,k} & \forall k, \forall m, \forall a
\end{array} \quad (3)$$

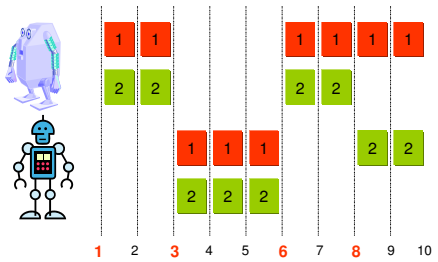

Figure 3: Optimal sequential resource allocation for 4 predetermined phases.

where transition probability $p^m_{i,a,j}$, reward $r^m_{i,a}$, initial probability distribution α^m_j (at the beginning of the execution), occupation measure $x^m_{i,a}$, resource prerequisite $u^m_{a,o}$, resource upper bound $\hat{\omega}_o$, finite horizon T, and the objective function $\sum_m \sum_i \sum_a x^m_{i,a} r^m_{i,a}$ are the same as in Eq.2.

Integer variable $\Delta^{m,k}_a$ indicates whether action a can be scheduled in agent m's policy in phase k, and $\omega^{m,k}_o$, which restricts $\Delta^{m,k}_a$ through the constraint $\omega^{m,k}_o \geq u^m_{a,o}\Delta^{m,k}_a$, represents the amount of resource o that agent m has within phase k. $\sum_m \omega^{m,k}_o = \hat{\omega}_o$ says that, for any resource o within any phase k, the amount of the allocated resource must equal the amount of the available resource.

An agent can leave a phase and enter another phase as time passes, but, in a global view, the total expected number of times that agent m visits state j must equal the probability that agent m is initially at state j plus the total expected number of times that agent m enters state j via all possible transitions. That is, we can use the same constraint $\sum_a x^m_{j,a} - \sum_i \sum_a p^m_{i,a,j} x^m_{i,a} = \alpha^m_j$ as in Eq.2 to model probability conservation. On the other hand, a phase transition, which is caused by a resource reallocation, might change the set of executable actions. To model the characteristic that the set of executable actions might differ in different phases, we use the aforementioned integer variable $\Delta^{m,k}_a$, which indicates whether action a can be executed by agent m within phase k, and link $x^m_{i,a}$ and $\Delta^{m,k}_a$ with the constraint $\sum_{i\in\mathbb{S}^k} x^m_{i,a} \leq T\Delta^{m,k}_a$ (where $\mathbb{S}^k$ represents the set of states within phase k), i.e., for any state in phase k, $x_{i,a} \equiv 0$ (i.e., action a is not executable) when $\Delta^{m,k}_a = 0$, and $x_{i,a}$ is not restricted when $\Delta^{m,k}_a = 1$ since any action can be executed at most T times during an execution.

Deriving optimal sequential resource allocation and joint policies from the solution to Eq.3 is straightforward. At the start time of phase k, resources are redistributed where $\omega^{m,k}_o$ units of resource o are allocated to agent m. Every agent m should adopt its policy $\pi^m_{i,a} = \frac{x^m_{i,a}}{\sum_a x^m_{i,a}}$ to maximize the total expected reward of the group of agents.

We now return to our running example introduced in Section 2 to illustrate how the total expected reward can be improved if resources can be reallocated during execution. Let us say that resources can be redistributed at time *1*, *3*, *6*, and *8*; this reallocation schedule decomposes the example problem into four phases of roughly equal size. Formulating and solving this problem with Eq.3 yields the sequential allocation depicted in Figure 3. We can see that resources are managed in a smarter way, where agent *2* no longer idles over the entire execution. As a result, the total expected reward increases to 65.04, 31% higher than with one-shot resource allocation. However, it should be noted that, although evenly decomposing a problem into phases is often a good strategy, we might be able to do better by optimizing the reallocation schedule, which will be discussed in the next section.

4. LIMITED NUMBER OF TIMES OF RESOURCE REALLOCATION

We now investigate a more general class of sequential resource allocation problems where there is no predetermined schedule to reallocate resources, and so agents have to determine for themselves when to adjust their resource allocations for achieving their remaining tasks better. Specifically, in this section we will assume that there is an upper bound K on the number of times resources can be reallocated. For example, agents might only be able to afford to hire a redistribution service (that will collect and redistribute resources) at most K times.

Obviously, we can enumerate all possible schemes of decomposing a problem into phases, and, for each scheme, adopt the solution algorithm presented in Section 3 to find the optimal solution for the sequential allocation problems of interest in this section. However, in many application domains, the number of possible decompositions is large and thus this straightforward approach might become infeasible. Instead, we propose an alternative solution approach that extends the MILP in Eq.3 to also incorporate decision making about phase decomposition.

The extended MILP is shown in Eq.4, where the objective function and probability conservation constraints are the same as Eq.3. The distinction is that, to characterize limitations on the number of times for resource reallocation, we represent resource upper bound constraints at each time step (instead of at each phase), and put in supplementary constraints to capture phase transitions.

$$\max \sum_m \sum_i \sum_a x^m_{i,a} r^m_{i,a}$$

$$\begin{array}{|ll|}
\multicolumn{2}{l}{\textit{Probability conservation:}} \\
x^m_{i,a} \geq 0 & \forall m, \forall i, \forall a \\
\sum_a x^m_{j,a} - \sum_i \sum_a p^m_{i,a,j} x^m_{i,a} = \alpha^m_j & \forall m, \forall j \\
\multicolumn{2}{l}{} \\
\multicolumn{2}{l}{\textit{Resource constraints:}} \\
\Delta^{m,t}_a \in \{0,1\} & \forall t, \forall m, \forall a \\
\omega^{m,t}_o \geq u^m_{a,o}\Delta^{m,t}_a & \forall o, \forall t, \forall m, \forall a \\
\sum_m \omega^{m,t}_o = \hat{\omega}_o & \forall o, \forall t \\
\sum_{i\in\mathbb{S}_t} x^m_{i,a} \leq \Delta^{m,t}_a & \forall t, \forall m, \forall a \\
\multicolumn{2}{l}{} \\
\multicolumn{2}{l}{\textit{Resource reallocation constraints:}} \\
\Psi_t \in \{0,1\} & \forall t \\
\Psi_{t=1} = 1 & \\
\omega^{m,t}_o - \omega^{m,t-1}_o \leq \omega_o \Psi_t & \forall o, \forall t > 1, \forall m \\
\sum_t \Psi_t \leq K &
\end{array} \quad (4)$$

where $p^m_{i,a,j}$, $r^m_{i,a}$, α^m_j, $x^m_{i,a}$, $u^m_{a,o}$, $\hat{\omega}_o$, and T have the same definitions as before. $\mathbb{S}_t$ represents the set of states at time t. Integer variable $\Delta^{m,t}_a$ indicates whether action a can be executed by agent m at time t, and $\omega^{m,t}_o$ represents the amount of resource o that agent m has at time t. The second subset of Eq.4's constraints (i.e., resource constraints) guarantees that the total amount of allocated resources must equal the

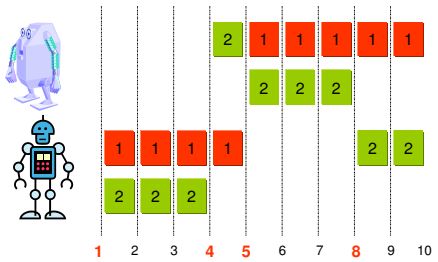

Figure 4: Optimal sequential resource allocation for 4 phases without a predetermined schedule.

total amount of available resources at any time.

To restrict the number of times for reallocation, we define variable Ψ_t, which is an integer in the interval [0, 1], to symbolize whether resources can be redistributed at time t. To sidestep the question of what the "default" resource allocation might be, we assume that resources are always initially allocated at the beginning of the execution, i.e., $\Psi_{t=1} = 1$. Note that $\omega_o^{m,t} - \omega_o^{m,t-1}$ can never be greater than ω_o since the incremental amount of resource o can never exceed the total amount of resource o; the constraint $\omega_o^{m,t} - \omega_o^{m,t-1} \leq \omega_o \Psi_t$ thus points out that Ψ_t must be 1 if any agent m has any extra amount of any resource at time t compared to time $t-1$. In other words, any resource exchange at time t will lead to $\Psi_t = 1$, which suggests that we can use the constraint $\sum_t \Psi_t \leq K$ to control the total number of times for reallocation, where K is the upper bound of the number of reallocation opportunities.

We conclude this section by illustrating the solution algorithm on our running example again. Recall that, in Section 3, we choose a fixed set of reallocation times $\{1, 3, 6, 8\}$ that result in a reward of 65.04. Now, we say that the agents will determine for themselves a set of reallocation times given an upper bound of 4 for the size of this set. Using Eq. 4, the optimal schedule to reallocate resources is computed as $\{1, 4, 5, 8\}$. Figure 4 depicts the detailed allocation. We can see that this schedule is more sophisticated; it gives high priority and allots sufficient time for agents to accomplish their highly rewarded tasks (i.e., task *3* of agent *1*, and task *1* of agent *2*). As a result, the total expected reward for the two agents increases to 72.25.

5. INCORPORATING COSTS FOR SEQUENTIAL RESOURCE ALLOCATION

Now we neither assume that the phases are predefined (Section 3) nor that the number of phases is bounded (Section 4). Instead, we assume that resource exchanges can occur at any time, and as many times as desired, but a cost is incurred for transferring resources. For example, agents can request the redistribution service as frequently as they desire, but each time step at which they use the service will cost the agents a constant fee. Or, in a more general case, the cost is determined by how many resources are transferred among the agents.

In general, coping with a constant reallocation cost is relatively easy since Eq.4 has paved the way to characterize the optimal number of times for redistributing resources. In brief, if the costs are calibrated with the utilities of the MDP policies, then the objective is to maximize the total expected reward, accounting for the costs of redistributing resources at particular times during the execution. In other words, a sequential allocation problem with constant reallocation cost c can be solved by slightly revising Eq.4: adopting the new objective function $\sum_m \sum_i \sum_a x_{i,a}^m r_{i,a}^m - c \sum_t \Psi_t$, and removing the constraint $\sum_t \Psi_t \leq K$ which is no longer applicable since agents can reallocate resources as frequently as they desire in such problems.

In the rest of this section, we focus on sequential allocation problems where the cost incurred in redistributing resources is determined by the *amount* of resources transferred. Since agents are cooperative, it does not matter which agent pays the transition costs. We arbitrarily assume that agent m pays the cost c_o^m when it acquires one (extra) unit of resource o from someone else, and the agent releasing the resource pays no cost.

Similarly to Eq.4, we use $\omega_o^{m,t}$ to represent the amount of resource o held by agent m at time t. The cost that agent m should pay for getting more resources at time t can then be represented as $\sum_o \theta(\omega_o^{m,t} - \omega_o^{m,t-1})$ where function $\theta(z)$ is a piecewise linear function, defined as:

$$\theta(z) = \begin{cases} z & z > 0 \\ 0 & otherwise \end{cases}$$

In many situations, a piecewise linear constraint can be represented as multiple linear constraints. We exploit such properties by introducing continuous variables $\epsilon_o^{m,t}$ and formulate the problem into the following MILP:

$$\max \sum_m \sum_i \sum_a x_{i,a}^m r_{i,a}^m - \sum_o \sum_m \sum_t c_o^m \epsilon_o^{m,t}$$

$$\begin{array}{|ll|} \hline
\textit{Probability conservation:} & \\
x_{i,a}^m \geq 0 & \forall m, \forall i, \forall a \\
\sum_a x_{j,a}^m - \sum_i \sum_a p_{i,a,j}^m x_{i,a}^m = \alpha_j^m & \forall m, \forall j \\
 & \\
\textit{Resource constraints:} & \\
\Delta_a^{m,t} \in \{0,1\} & \forall t, \forall m, \forall a \\
\omega_o^{m,t} \geq u_{a,o}^m \Delta_a^{m,t} & \forall o, \forall t, \forall m, \forall a \\
\sum_m \omega_o^{m,t} = \hat{\omega}_o & \forall o, \forall t \\
\sum_{i \in \mathbb{S}_t} x_{i,a}^m \leq \Delta_a^{m,t} & \forall t, \forall m, \forall a \\
 & \\
\textit{Cost constraints:} & \\
\epsilon_o^{m,t} \geq 0 & \forall o, \forall t, \forall m \\
\epsilon_o^{m,t=1} = \omega_o^{m,t=1} & \forall o, \forall m \\
\epsilon_o^{m,t} \geq \omega_o^{m,t} - \omega_o^{m,t-1} & \forall o, \forall t > 1, \forall m \\ \hline
\end{array} \quad (5)$$

As shown in Eq.5, when $t > 1$, $\epsilon_o^{m,t}$ is constrained by $\epsilon_o^{m,t} \geq 0$ and $\epsilon_o^{m,t} \geq \omega_o^{m,t} - \omega_o^{m,t-1}$. In other words, $\epsilon_o^{m,t} \geq \omega_o^{m,t} - \omega_o^{m,t-1}$ when $\omega_o^{m,t} > \omega_o^{m,t-1}$, and $\epsilon_o^{m,t} \geq 0$ under other circumstances. Note that the objective function of Eq.5 is to maximize $\sum_m \sum_i \sum_a x_{i,a}^m r_{i,a}^m - \sum_o \sum_m \sum_t c_o^m \epsilon_o^{m,t}$, which implies that the second term $\sum_o \sum_m \sum_t c_o^m \epsilon_o^{m,t}$ should be as small as possible for an optimal solution that yields the highest expected utility. That is, $\epsilon_o^{m,t}$ should reach its lower bound for any optimal solution of Eq.5, i.e., $\epsilon_o^{m,t} = \omega_o^{m,t} - \omega_o^{m,t-1}$ when $\omega_o^{m,t} > \omega_o^{m,t-1}$ and $\epsilon_o^{m,t} = 0$ otherwise, which exactly matches our expectation of using $\epsilon_o^{m,t}$ to represent the piecewise linear cost function $\theta(\omega_o^{m,t} - \omega_o^{m,t-1})$.

We now revisit our running example to illustrate how the above algorithm manages the transfer of resources among agents based on the transfer cost. By formulating the problem into Eq 5, we can find that, at zero cost, the optimal

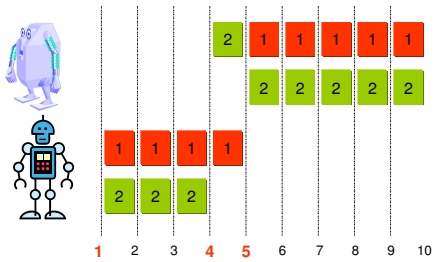

Figure 5: Optimal sequential resource allocation when the cost of transferring resources is 5 per unit.

sequential allocation is identical to Figure 4, but as the transfer cost increases, the amount of resources transferred decreases. As an example, when the cost of transferring one unit of a resource is 5, the optimal sequential allocation, which is shown in Figure 5, is to transfer 4 units of resources over the entire execution (with two at the initial time *1*, one at time *4*, and one at time *5*). The total expected reward is 48.72, which is higher than the reward of transferring 5 units of resources as in Figure 4 where the reward is $72.25 - 5 \times 5 = 47.25$.

6. EMPIRICAL RESULTS

To this point, we have described a series of solution algorithms for increasingly difficult sequential resource allocation problems, and illustrated them through a simple example. In this section, we give a preliminary evaluation of our techniques on more complex problems with larger state spaces and various degrees of constrainedness.

In each of the test problem instances we used, there are five cooperative agents where each agent has a set of five tasks. These cooperative agents try to maximize their total expected reward within $T = 15$ time steps. The reward of a task is uniformly randomly distributed in the range [40, 100]. Like our running example problems, the duration of completing a task is uncertain. The expected duration D_i of task i is an integer uniformly randomly selected within the range [2, 4]; its owner agent has $\frac{1-p}{2}$, p, and $\frac{1-p}{2}$ probability of completing it within the duration $\lfloor D_i \times 50\% \rfloor$, D_i, and $\lfloor D_i \times 150\% \rfloor$ respectively,[2] where p is uniformly randomly distributed in the range [0, 1]. Tasks are temporally constrained. The release time RL_i of task i is an integer uniformly randomly selected in the range $[1, T]$, and the deadline DL_i is its release time plus three times its expected duration but can never exceed the finite horizon T, i.e. $DL_i = \min(T, RL_i + 3 \times D_i)$. There are six different types of resources in the system. The resource prerequisite of each task is two units of resources whose types are independently randomly drawn from the six resource types.

In our simulation, we used various bounds on the number of times for resource allocation, various numbers of units of resources, and various transfer costs, to evaluate our algorithms. We ran experiments on a Pentium IV machine, and used *cplex* 9.1 (www.ilog.com) as our MILP solver. Each data point shown in the figures of this section is an average

[2] $\lfloor z \rfloor$ is the floor function that rounds z to the nearest integer below z.

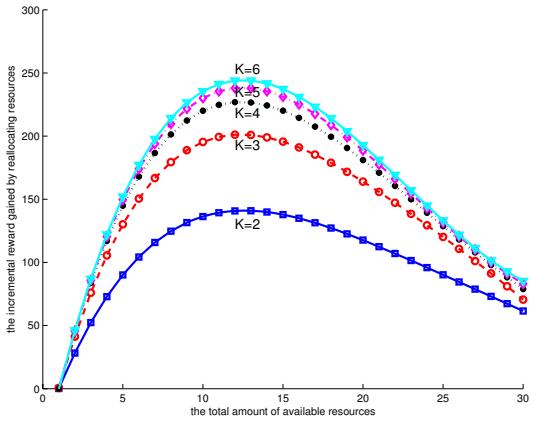

Figure 6: The increase in the total expected reward for various numbers of times for resource reallocation on varying levels of resource constraints.

over 100 randomly generated test problems.

Figure 6 illustrates the contribution of sequential resource allocation on the total expected reward of the agents under various degrees of resource constraints. The x-axis of the figure represents the total amount of available resources (i.e., $\sum_o \hat{\omega}_o$), and the y-axis specifies the incremental reward gained by adopting the optimal reallocation schedule and the optimal resource allocation and joint policy at each phase derived by Eq.4 where K indicates the number of times of resource reallocation. We can see that, when resources are very scarce, the gain by sequentially allocating resources is low. This is because resources are so limited that few tasks are executable. Although sequential resource allocation can let most of these tasks be accomplished, it does not result in a large gain in the absolute scale. The incremental reward by sequential resource allocation gradually increases as the amount of available resources increases. It reaches its peak when the total number of units of resources is around 12. However, as resources become more abundant, the incremental reward by reallocating resources starts to decrease (and will eventually reduce to zero), because when an agent has sufficient resources, it can achieve as much reward as it can without exchanging resources with others.

In Figure 6, we can also see that the gain by increasing the number of times for reallocation becomes lower and lower as the number of times for reallocation increases. The underlying reason is that completing a task needs some time, and its required resources should be held during this period of time. In other words, too frequent reallocation of resources is unnecessary.

We then fixed the amount of available resources to 12 (but types of resources are randomly set for each problem), and allowed agents to reallocate resources as many times as they desire, but transferring one unit of one type of resource will incur a cost of c. In Figure 7, we show the curve of the average amount of resources transferred during the execution (with the error bars showing standard deviation) as the transfer cost increases (recall that a resource can be trans-

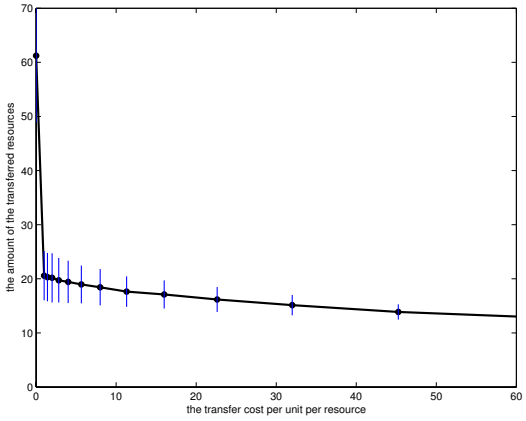

Figure 7: The number of units of transferred resources given various transfer costs.

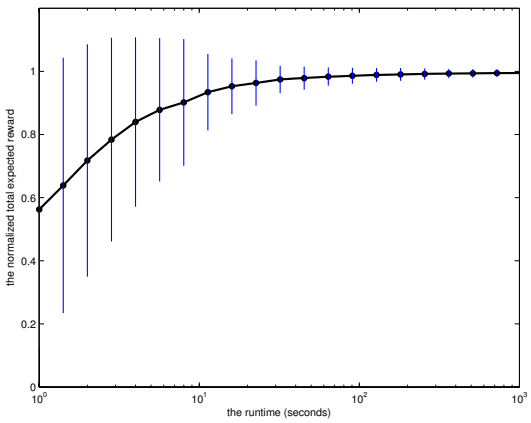

Figure 8: Anytime performance of the MILP-based algorithm (4 phases, 5 agents, and 12 units of resources) where the quality of the optimal solution is normalized to 1.

ferred more than once). An interesting discovery is that, when the cost increases from zero to some minor value (e.g., $c = 1$), the amount of transferred resources reduces dramatically. This is because, when transfer is no longer free, many resource transfers are discouraged since each of them can only incur a minor gain. Thereafter, the amount of transferred resources gradually decreases as the cost increases.

Finally, we evaluated the efficiency of our solution techniques. Recall that the search space of sequential resource allocation is usually very large. Let us consider one of our test problems where the amount of each type of resources is 2 units. Since there are 15 different ways to allocate 2 units of one type of resources to 5 agents (assuming that each agent will receive an integer amount of resources), the number of possible one-shot allocations for six different types of resources is $15^6 = 11390625$. If agents can (re)allocate resources K times during the execution (i.e., $K - 1$ times besides the initial allocation), the number of possible sequential allocations is $\binom{T-1}{K-1} \times 11390625^K$, which is about 6.1276×10^{30} when $T = 15$, $K = 4$. Our algorithms that formulate sequential resource allocation problems into MILPs are usually able to efficiently explore such large spaces. However, it should be noted that MILP is NP-Complete, which means that in the worst case the MILP-based solution algorithm might still take a long time to find an optimal solution. In our experiments, about 16% of the randomly generated problems required more than 600 seconds to find an optimal solution.

Fortunately, state-of-art MILP solvers (including *cplex* which we used in this work) are able to exploit the structure of the MILP formulation and return a good solution using much less time; our MILP-based algorithms can take advantage of this property. Figure 8 shows the average anytime performance (with the error bars showing standard deviation) of our algorithm for 12 units of resources and 4 times of resource reallocations (other parameters are the same as before) where the quality of the optimal solution for each test problem is normalized to 1. These results indicate that the MILP-based algorithm has good anytime performance. On average, it can find a good (approximately 90% of the optimal quality) solution within 20 seconds, and find a near-optimal (approximately 97% of the optimal quality) solution within 100 seconds for our test problems.

7. CONCLUSIONS AND FUTURE WORK

We have described a novel MILP-based solution approach for automatic sequential resource allocation among a group of agents in complex environments with limited shared resources and with uncertainties, and have shown through experiments that our technique can increase agents' rewards for varying levels of constraints on resources and on the reallocation of resources.

The contribution of this work is that it provides a framework that explicitly takes into account potential opportunities during execution to redistribute resources, and proposes solution algorithms to optimize the use of these opportunities in constrained and stochastic systems. Furthermore, it defines an algorithm for automating the process of finding and using phases, which eliminates the need for having resource reallocation schedules predefined in the description of a problem and thus enables sequential resource allocation techniques to be used in a wider variety of application domains.

An assumption that is made in this work is that, once a resource reallocation is scheduled, agents are able to send and receive resources among themselves at that scheduled time. We plan to relax this assumption in the future to consider sequential allocation problems with additional constraints on when agents can exchange resources. For example, physical agents might only be able to exchange resources when they are at the same location at the same time. Or, as another example, a task might be incapable of being aborted once it starts, and thus it is impossible to reallocate the resources used by that task until it is completed. One possible direction we would like to explore for these problems is to extend existing decentralized MDP (Dec-MDP) solution algorithms to characterize constraints on resources and reallocation of resources. In addition, we also plan to study more general

sequential reallocation problems by considering the fact that passing a resource might have some probability of failure.

The MILP-based algorithm presented in this paper can derive the optimal reallocation schedule and the optimal resource allocation and joint policy at each phase, but in larger problems it might take too long to run. Therefore, another future direction is to build on efficient approximate algorithms for sequential resource allocation problems. We expect that abstraction techniques and factored MDPs can be used to serve this purpose.

8. ACKNOWLEDGEMENTS

This material is based upon work supported in part by the DARPA/IPTO COORDINATORs program and the Air Force Research Laboratory under Contract No. FA8750–05–C–0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

9. REFERENCES

[1] E. Altman. Constrained Markov decision processes with total cost criteria: Lagrange approach and dual LP. *Methods and Models in Operations Research*, 48:387–417, 1998.

[2] E. Altman. *Constrained Markov Decision Processes*. Chapman and HALL/CRC, New York, 1999.

[3] A. Bockmayr and Y. Dimopolous. Mixed integer programming models for planning problems. In *CP-98 Constraint Problem Reformulation Workshop*, 1998.

[4] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, New York, 1998.

[5] D. A. Dolgov and E. H. Durfee. Constructing optimal policies for agents with constrained architectures. Technical Report CSE-TR-476-03, University of Michigan, 2003.

[6] D. A. Dolgov and E. H. Durfee. Computationally efficient combinatorial auctions for resource allocation in weakly-coupled MDPs. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, July 2005.

[7] D. A. Dolgov and E. H. Durfee. Resource allocation among agents with MDP-induced preferences. *Journal of Artificial Intelligence Research (JAIR-06)*, 27, November 2006.

[8] H. Kautz and J. Walser. Integer optimization models of AI planning problems. *Knowledge Engineering Review*, 15(1):101–117, 2000.

[9] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/innovative applications of artificial intelligence*, pages 165–172, 1998.

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

[11] T. Vossen, M. Ball, A. Lotem, and D. Nau. On the use of integer programming models in AI planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 304–309, 1999.

[12] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, 1998.

[13] J. Wu and E. H. Durfee. Automated resource-driven mission phasing techniques for constrained agents. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 331–338, 2005.

[14] J. Wu and E. H. Durfee. Mixed-integer linear programming for transition-independent decentralized MDPs. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1058–1060, 2006.