In *Proceedings of The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-06)*

Resource Allocation Among Agents with Preferences Induced by Factored MDPs

Dmitri Dolgov and Edmund Durfee
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109
{ddolgov, durfee}@umich.edu

ABSTRACT

Distributing scarce resources among agents in a way that maximizes the social welfare of the group is a computationally hard problem when the value of a resource bundle is not linearly decomposable. Furthermore, the problem of determining the value of a resource bundle can be a significant computational challenge in itself, such as for an agent operating in a stochastic environment, where the value of a resource bundle is the expected payoff of the optimal policy realizable given these resources. Recent work has shown that the structure in agents' preferences induced by stochastic policy-optimization problems (modeled as MDPs) can be exploited to solve the resource-allocation and the policy-optimization problems simultaneously, leading to drastic (often exponential) improvements in computational efficiency. However, previous work used a flat MDP model that scales very poorly. In this work, we present and empirically evaluate a resource-allocation mechanism that achieves much better scaling by using factored MDP models, thus exploiting both the structure in agents' MDP-induced preferences, as well as the structure within agents' MDPs.

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, (Multi-)agent planning.

1. INTRODUCTION

Resource allocation is recognized as an important problem in many research fields, such as computer science, economics, and operations research. The core question of resource allocation is how to distribute a set of scarce resources among a set of agents (either cooperative or self-interested) in the best possible way, with social welfare (sum of agents' utilities) as one of the most popular optimization criteria. This problem is computationally challenging (NP-complete [19]) when agents' utility functions are defined over *bundles* of resources and are not linearly decomposable. Such computational challenges in combinatorial resource-allocation problems have recently precipitated the move towards resource-allocation mechanisms that model the mechanisms underlying agents' resource preferences (e.g., [15]), as opposed to the more-traditional black-box-utility models.

In particular, we have previously proposed a model where an agent's resource preferences are defined by an underlying Markov decision process, where resources are required to enable actions, and the value of a resource bundle is the expected payoff of the optimal policy that is realizable given these resources [8, 9]. In other words, an agent's MDP is parameterized by the available resources, and the agent's utility for resources is defined as the expected value of the best realizable policy. The model assumes that agents' planning problems are *weakly-coupled* [16], i.e., once the resources are distributed, the agents are independent.

Under this model, a naive resource-allocation approach would require each agent to formulate and solve an exponential number of MDPs (one for each resource bundle) just to define its utility function, and then to solve an NP-hard allocation problem on the exponentially-large input. However, a superior method can be implemented that combines the resource-allocation and the policy-optimization problems and formulates them as a single constrained MDP that can then be reduced to a mixed integer program, leading to an exponential reduction in the number of binary decision variables. This approach exploits the structure in agents' preferences that stems from the regularities of the underlying Markov process, which leads to drastic improvements in scalability of the resource-allocation problem.

There is, however, another source of intractability in the model described above that is due to the MDP models themselves. The traditional *flat* way of representing an MDP requires an explicit enumeration of all possible states of the system, which subjects the model to the curse of dimensionality [2] and leads to an exponential blow-up of the state space in terms of the number of *state features* of the problem. This representational challenge is addressed by *factored* MDPs [3] that model the state space as a cross product of state features, represent the transition function as a dy-

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

namic Bayesian network, and assume the reward function can be additively decomposed into functions with compact domains. Unfortunately, a solution to a well-structured factored MDP does not, in general, maintain the structure and compactness of the problem [14, 7], which forces the need for approximate solutions. One popular family of such methods that is particularly well-suited for our goal is approximate linear programming (ALP) [20, 6], for which there exist several very efficient algorithms [17, 12, 5, 10].

The main contribution of this work is that it extends the line of research on resource-allocation methods for agents with MDP-induced preferences [8, 9] to factored MDP models. It builds on the ALP methodologies of [12] to construct an efficient algorithm that combines resource allocation and approximate policy optimization for factored MDPs. This effectively exploits both the structure in agents' preferences that is *due to the underlying MDPs*, as well as the structure *within the MDPs* themselves, leading to exponential improvements in efficiency both in terms of the size of the resource-allocation problem (as the number of resource types increases) and the size of the policy-optimization problem (as the number of state features in an MDP grows).

2. FACTORED MDP AND ALP

A discrete-time, infinite-horizon, discounted MDP (e.g., [18]) can be described as $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S} = \{s\}$ is the finite set of system states, $\mathcal{A} = \{a\}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition function ($P(\sigma|s, a)$ is the probability of moving into state σ upon executing action a in state s), $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the (bounded) reward function ($R(s, a)$ is the reward for executing action a in state s), and $\gamma \in [0, 1)$ is the discount factor (a unit reward at time τ is aggregated into the total reward as γ^τ).

A solution to such an MDP is a stationary, deterministic policy π. We define a policy as a mapping of states to probability distributions over actions: $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, where $\pi(s, a)$ defines the probability that action a is executed in state s (for deterministic policies, only one $\pi(s, a)$ per state is nonzero). One way of solving an MDP is to compute the optimal value function v, which defines, for every state, the total expected discounted reward of the optimal policy. It is then optimal to simply act greedily with respect to v.

The optimal value function can be obtained, for example, using the following minimization linear program (LP), which is often called the *primal LP* of an MDP (e.g., [18]):

$$\begin{aligned} &\min \sum_s \alpha(s) v(s) \qquad \text{s.t.:} \\ &v(s) \geq R(s, a) + \gamma \sum_\sigma P(\sigma|s, a) v(\sigma), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \end{aligned} \tag{1}$$

where α is an arbitrary strictly positive distribution over the state space ($\alpha(s) > 0$). This LP has $|\mathcal{S}|$ variables and $|\mathcal{S}||\mathcal{A}|$ constraints. Equivalently, the problem can be formulated as the *dual LP* with $|\mathcal{S}||\mathcal{A}|$ variables and $|\mathcal{S}|$ constraints:

$$\begin{aligned} &\max \sum_{s,a} R(s, a) x(s, a) \qquad \text{s.t.:} \\ &\sum_a x(\sigma, a) - \gamma \sum_{s,a} x(s, a) P(\sigma|s, a) = \alpha(\sigma), \quad \forall s \in \mathcal{S}, \end{aligned} \tag{2}$$

where $x(s, a) \geq 0$ is called the *occupation measure* ($x(s, a)$ is the discounted number of executions of a in s), and the constraints ensure the conservation of flow through each state.

Given a solution to (2), an optimal policy can be computed as: $\pi(s, a) = x(s, a) / \sum_a x(s, a)$. A policy thus obtained is *uniformly-optimal* (optimal for all initial states) when $\alpha(s)$ is strictly positive. The existence of uniformly-optimal policies does not hold for the resource-parameterized MDP models that we use in this work (detailed in Section 3), so for such problems the initial conditions α will be an important part of problem specification.

2.1 Factored MDPs

The standard MDP model defined above requires an enumeration of all system states and thus scales very poorly. This problem is addressed by a compact MDP representation [3, 4] that defines the state space as the cross-product of state features: $\mathcal{S} = z_1 \times z_2 \ldots z_N$, and uses a factored transition function and an additively-separable reward function.

The transition function is specified as a two-layer dynamic Bayesian network (DBN), with the current state features viewed as the parents of the next time-step features:

$$P(\sigma|s, a) = P(\mathbf{z}(\sigma)|\mathbf{z}(s), a) = \prod_{j=1}^{J_P} p_j(\mathbf{z}_j(\sigma)|a, \mathbf{z}_{p_j}(s)), \tag{3}$$

where $\mathbf{z}(\cdot)$ is the instantiation of all $\mathcal{Z}$ features corresponding to a state, $z_j(\cdot)$ denotes the value of the j^{th} state feature of a state, and $\mathbf{z}_{p_j}(\cdot)$ is the instantiation of the DBN-parents $\mathcal{Z}_{p_j}$ of z_j. Likewise, in the rest of this paper, we will use $\mathcal{Z}_\varphi$ to refer to the set of features in the domain of function φ, and $\mathbf{z}_\varphi$ to refer to an instantiation of these features.

The reward function is compactly defined as

$$R(s, a) = \sum_{j=1}^{J_R} r_j(\mathbf{z}_{r_j}(s), a), \tag{4}$$

where $\mathbf{z}_{r_j}(\cdot)$ is an instantiation of the features $\mathcal{Z}_{r_j} \subseteq \mathcal{Z}$ that are in the domain of the j^{th} local reward function r_j.

Clearly, this factored representation is only beneficial if the local transition functions $p_j(z_j(\sigma)|\mathbf{z}_{p_j}, a)$ and local reward functions $r_j(\mathbf{z}_{r_j}, a)$ have small domains: $|\mathcal{Z}_{p_j}| \ll |\mathcal{Z}|$ and $|\mathcal{Z}_{r_j}| \ll |\mathcal{Z}|$, i.e., each reward and transition component depends on a small subset of all state features $\mathcal{Z}$.

2.2 Primal ALP

Approximate linear programming [20, 6] lowers the dimensionality of the primal LP (1) by restricting the optimization to the space of value functions that are linear combinations of a predefined set of K basis functions h:

$$v(s) = v(\mathbf{z}(s)) = \sum_{k=1}^{K} h_k(\mathbf{z}_{h_k}(s)) w_k, \tag{5}$$

where $h_k(\mathbf{z}_{h_k})$ is the k^{th} basis function defined on a small subset of the state features $\mathcal{Z}_{h_k} \subset \mathcal{Z}$, and w are the new optimization variables. This technique is similar to linear regression, where a function is approximated as a linear combination of a given (in general, non-linear) basis. The difference, however, is that here instead of minimizing a measure of error for the given data points, the goal is to minimize the measure to the unknown optimal value function. For the approximation to be computationally effective, the domain of each basis function has to be small ($|\mathcal{Z}_{h_k}| \ll |\mathcal{Z}|$).

As a notational convenience, we can rewrite the above as $v = Hw$, where H is a $|\mathcal{S}| \times |w|$ matrix composed of basis

functions h_k.[1] Thus, the LP (1) becomes:

$$\min \alpha^T H w \big| AHw \geq r, \tag{6}$$

where the constraint matrix is $A_{sa,\sigma} = \delta(s,\sigma) - \gamma P(\sigma|s,a)$ (where $\delta_{s\sigma}$ is the Kronecker delta, $\delta_{s\sigma} = 1 \Leftrightarrow s = \sigma$).

For this method to be effective, we need to be able to efficiently compute the objective function $\alpha^T H$ and the constraints AH, which can be done as described in [12]. Consider a factored initial distribution:

$$\alpha(s) = \prod_{j=1}^{J_\alpha} \mu_j(\mathbf{z}_{\mu_j}(s)),$$

where, as usual, the domain of each factor μ_j is taken to be small ($|\mathcal{Z}_{\mu_j}| \ll |\mathcal{Z}|$). The objective function is computed as:

$$\begin{aligned}(\alpha^T H)_k &= \sum_s \alpha(s) H_{sk} = \sum_s \prod_i \mu_j(\mathbf{z}_{\mu_j}(s)) h_k(\mathbf{z}_{h_k}(s)) \\ &= \sum_{\mathbf{z}'} \prod_{j'} \mu_{j'}(\mathbf{z}'_{\mu_{j'}}) h_k(\mathbf{z}'_{h_k}),\end{aligned}$$

where $\mathbf{z}'$ iterates over all features in the union of the domain of h_k and the domains of those $\mu_{j'}$ that have a non-zero intersection with the domain of h_k: $\mathbf{z}' = \{\mathbf{z}_{\mu_j} \cup \mathbf{z}_{h_k} : \mathcal{Z}_{\mu_j} \cap \mathcal{Z}_{h_k} \neq \varnothing\}$, because all μ_i that do not have any variables in common with h_k factor out and their sum is 1 (since it is a sum of a probability distribution over its domain).[2]

The constraints in (6) are also computed efficiently:

$$\begin{aligned}(AH)_{sa,k} &= \sum_\sigma A_{sa,\sigma} h_k(\sigma) = \sum_\sigma (\delta_{s\sigma} - \gamma P(\sigma|s,a)) h_k(\sigma) \\ &= \sum_{\mathbf{z}} \delta(\mathbf{z}(s), \mathbf{z}) h_k(\mathbf{z}) - \gamma \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{z}(s), a) h_k(\mathbf{z}).\end{aligned} \tag{7}$$

The first sum can be computed efficiently, because it is simply $h_k(\mathbf{z}_{h_k}(s))$, since δ is nonzero only at $\mathbf{z}(\sigma) = \mathbf{z}(s)$. The second term can also be computed efficiently, since $P(\mathbf{z}(\sigma)|\mathbf{z}(s), a)$ is a factored probability distribution (just like in the case of α above).

Notice that in the above, the second term in (7) can be precomputed efficiently by performing a backprojection of every basis function h_k:

$$g_k(\mathbf{z}_{g_k}, a) = \sum_{\mathbf{z}} P(\mathbf{z}'|\mathbf{z}(s), a) h_k(\mathbf{z}) \quad \forall k \in [1, K], \tag{8}$$

where the domain of the backprojection g_k is the union of the DBN-parents of all features that are in the domain of the k^{th} basis function: $\mathcal{Z}_{g_k} = \bigcup_{j: z_j \in \mathcal{Z}_{h_k}} \mathcal{Z}_{p_j}$.

The primal ALP described above reduces the number of optimization variables from $|\mathcal{S}|$ to $|w| = K$, and the coefficients of the objective function and every constraint row can be computed efficiently. However, the number of rows in the constraint matrix remains exponential at $|\mathcal{S}||\mathcal{A}|$, so the ALP has to undergo some additional transformation (or approximation) to become feasible. To address this issue, several techniques have been proposed, such as exploiting problem structure [12], sampling [5], and linear approximations [10]. The benefit of the former, compared to the latter two, is that it does not introduce a new source of approximation error, while the downside is that the number of constraints needed to represent the feasible region exactly is exponential in the induced tree width of the associated cluster graph (a graph with a vertex per variable and edges between variables that appear together in the domain of any function).

[1]Note that the exponentially-sized H is never explicitly written out, since each column is a basis function that can be represented compactly.

[2]See [11] or [10] for examples of how the LP coefficients can be computed efficiently.

2.3 Dual ALP

When dealing with constrained MDPs (flat or factored), such as the ones that arise in the resource allocation problem discussed in Section 4, it is more practical to work with the duals of the (A)LPs. Guestrin [11] considers the dual of (6):

$$\max r^T x \Big| H^T A^T x = \alpha. \tag{9}$$

This LP has $|\mathcal{S}||\mathcal{A}|$ variables and $|w| = K$ constraints (approximated flow conservation), where each of the K constraints corresponds to a linear combination of $|\mathcal{Z}_{h_k}|$ real flow constraints, with h_k defining the weights with which they are aggregated into the new approximate constraint. Clearly, (9) is always feasible. Furthermore, as shown in [11], a solution to (9) constitutes a valid density function, summing to $(1-\gamma)^{-1}$, which implies that the LP is bounded.

A problem with the LP (9) is that it has an exponential number of optimization variables. However, as proposed by Guestrin [11], the exact occupation measure x can be represented more compactly by using *marginal occupation measures* (or marginal visitation frequencies) ξ, which define the occupation measure over subsets of the state features. Let us define a marginal occupation measure over each of the following clusters of variables: the domain of every primal basis function $\mathcal{Z}_{h_k}$, the domain of every local reward function $\mathcal{Z}_{r_j}$, and the domains of all backprojections of the basis functions $\mathcal{Z}_{g_k}$. The challenge then becomes in ensuring that the marginal occupation measures ξ are consistent and correspond to a valid global distribution over $\mathcal{Z}$. Fortunately, global consistency (but not necessarily correspondence to a valid flat occupation measure x) can often be achieved via local marginal consistency constraints [11] that ensure that marginal occupation measures ξ_1 and ξ_2 agree on the values of features they share (with non-overlapping features marginalized out):

$$\sum_a \sum_{\widetilde{\mathbf{z}_j}, a} \xi_j(\mathbf{z}_j, a) = \sum_a \sum_{\widetilde{\mathbf{z}_i}, a} \xi_i(\mathbf{z}_i, a), \qquad \forall \mathbf{z}_{ji}, \tag{10}$$

where $\mathcal{Z}_{ji} = \mathcal{Z}_i \bigcup \mathcal{Z}_j$ are the shared features, and $\widetilde{\mathcal{Z}_i} = \mathcal{Z}_j \setminus \mathcal{Z}_i$ and $\widetilde{\mathcal{Z}_j} = \mathcal{Z}_i \setminus \mathcal{Z}_j$ are unique. As is the case in graphical models [13] more generally, and as observed for this particular case in [11], when the cluster graph associated with the domains of the marginal occupation measures ξ forms a *junction tree*[3] local consistency implies global consistency. If the original cluster graph does not form a junction tree, it can be converted to one by applying the widely-used in inference techniques of *moralization* and *triangulation*.[4]

Using the marginal occupation measures, the LP (9) can be represented more compactly as the following LP defined

[3]A junction tree is an undirected tree with clusters of state features as nodes, with the property that if a feature appears in clusters A and B, it also appears in any cluster in the path from A to B.

[4]Note that while the problem of computing the junction tree with the smallest width is an NP-hard problem [1], there are efficient approximations.

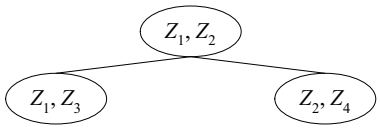


Figure 1: A cluster graph that forms a junction tree.

on marginal occupation measures $\xi_j(\mathbf{z}_j)$, one per each cluster in $\mathcal{Z}_{h_k}$, $\mathcal{Z}_{g_k}$, $\mathcal{Z}_{r_j}$.[5]

$$\begin{aligned}
&\max \sum_{m=1}^{M} \sum_{a} \sum_{\mathbf{z}_{r_j}} r_j(\mathbf{z}_{r_j}, a)\xi_{r_j}(\mathbf{z}_{r_j}, a) \qquad \text{s.t.:} \\
&\sum_{a} \xi_{h_k}(\mathbf{z}_{h_k}, a)h_k(\mathbf{z}_{h_k}) - \gamma \sum_{a} g_k(a)\xi_{g_k}(\mathbf{z}_{h_k}, a) = \\
&\qquad\qquad \sum_{\mathbf{z}_{h_k}} \alpha(\mathbf{z}_{h_k})h_k(\mathbf{z}_{h_k}), \quad \forall k \in [1, K]; \\
&\sum_{\widetilde{\mathbf{z}_j}} \xi_j(\mathbf{z}_j, a) = \sum_{\widetilde{\mathbf{z}_i}} \xi_i(\mathbf{z}_i, a), \\
&\quad \forall a, \xi_j, \xi_i, \mathbf{z}_j, \mathbf{z}_i, \ \widetilde{\mathcal{Z}_i} = \mathcal{Z}_j \setminus \mathcal{Z}_i, \ \widetilde{\mathcal{Z}_j} = \mathcal{Z}_i \setminus \mathcal{Z}_j; \\
&\sum_{a} \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) = (1-\gamma)^{-1}, \qquad \forall j, \ \mathbf{z}_j; \\
&\xi_j(\mathbf{z}_j, a) \geq 0, \qquad \forall a, j, \mathbf{z}_j.
\end{aligned} \tag{11}$$

Given a solution $\xi_j(\mathbf{z}_j)$ to the above LP, the optimal policy for any given state can be computed in the following way, detailed in [12]. Given the dual solution $\xi_j(\mathbf{z}_j)$, compute the primal solution w that defines the value-function approximation. Given the current state, do a one-step lookahead and, for each possible next state, compute the value-function estimate based on w, and greedily select the action that maximizes the sum of the one-step reward and the expected value of the next step (as in the standard MDP).

Another way of obtaining the policy is to calculate it directly from the marginal occupation measures ξ. Each marginal occupation measure forms a valid distribution on its domain, so we can use a variant of the junction-tree inference algorithm for calculating the distribution over actions, given the current state. The procedure is described in Algorithm 1 and works by simply traversing the cluster tree from the root and factoring in the marginal occupation measures along the way, while normalizing appropriately. Note that because the marginal occupation measures (even if globally consistent) do not necessarily correspond to a valid occupation measure, there is not a one-to-one correspondence between the marginals ξ and policy π, as between x and π in the flat case.

EXAMPLE 1. *Consider the cluster graph in Figure 1. The marginal occupation measures are: $\xi(z_1, z_2, a)$, $\xi(z_1, z_3, a)$, and $\xi(z_2, z_4, a)$. This graph forms a junction tree, so global consistency of marginal occupation measures is guaranteed whenever local consistency constraints are satisfied:*

$$\begin{aligned}
\sum_{z_2} \xi(z_1, z_2, a) &= \sum_{z_3} \xi(z_1, z_3, a), \qquad \forall z_1, a; \\
\sum_{z_1} \xi(z_1, z_2, a) &= \sum_{z_4} \xi(z_2, z_4, a), \qquad \forall z_2, a.
\end{aligned}$$

The global distribution for this problem is:

$$x(z_1, z_2, z_3, z_4, a) = \frac{\xi(z_1, z_2, a)\xi(z_1, z_3, a)\xi(z_2, z_4, a)}{\sum_{z_1} \xi(z_1, z_2, a) \sum_{z_2} \xi(z_1, z_2, a)},$$

which is what would be computed by Algorithm 1. □

[5]Here and below, for simplicity we assume the cluster graph is a junction tree. If not, the LP will have to undergo a preprocessing step, as described above.

Algorithm 1: Policy from occupation measures.

Function PolicyForState
in : $\mathbf{z}$ – current state
out : $\pi(\mathbf{z}, a)$ – policy for current state

$\pi(\mathbf{z}, a) \leftarrow$ PolicyForStateRec($\mathbf{z}$, root-cluster, $\mathbf{1}$, $\varnothing$)
if $\sum_a \pi(\mathbf{z}, a) = 0$ **then**
  /* `use any action supported by marginals` */
  $\pi(\mathbf{z}, a) \leftarrow 1$ for any a, s.t., $\exists j, \mathbf{z}'_j : \ \xi_j(\mathbf{z}'_j, a) > 0$
end

Function PolicyForStateRec
in : $\mathbf{z}$ – current state
: j – cluster id
: $\pi(\mathbf{z}, a)$ – marginal policy
: Ω – features already factored into policy
out : $\pi'(\mathbf{z}, a)$ – marginal policy
: Ω' – features already factored into policy

/* `update features already factored in` */
$\Omega' \leftarrow \Omega \bigcup \mathcal{Z}_{\xi_j}$
/* `factor in` j^{th} `marginal` */
$\pi'(\mathbf{z}, a) \leftarrow \pi(\mathbf{z}, a)\xi(\mathbf{z}_j, a)$
/* `normalize by overlapping features` */
$\mathcal{Z}' \leftarrow \Omega \bigcap \mathcal{Z}_{\xi_j}$
$\pi'(\mathbf{z}, a) \leftarrow \pi'(\mathbf{z}, a)/\xi(z')$
/* `factor in children` */
forall $m \in Children(j)$ **do**
  $\pi'' \leftarrow PolicyForStateRec(j, \pi', \Omega)$
  $\pi'(\mathbf{z}, a) \leftarrow \pi'(\mathbf{z}, a)\pi''(\mathbf{z}, a)$
end

3. SINGLE-AGENT MODEL

We now briefly describe the model of the capacity-limited agent with an MDP whose action set is parameterized by the resources available to the agent. This is a slight extension of the model described in [8] to factored problems and arbitrary resource requirements.

The justification behind the model described in this section is that often an agent has many capabilities that are all in principle available to it, but not all combinations of these capabilities are realizable within the agent's architectural limitations, because choosing to enable some of the capabilities might seize the resources needed to enable others. In other words, a particular policy might not be feasible

because the agent's architecture does not support the combination of capabilities required for that policy.

We model this constrained policy-optimization problem as follows. The agent has a set of actions that are potentially executable, and each action requires a certain combination of resources. All resources have capacity costs, and each agent has capacity constraints that limit what resources it can make use of. For example, a delivery company needs vehicles and loading equipment (resources) to make its deliveries (execute actions). However, all equipment costs money and requires manpower to operate it (capacity costs). Therefore, the amount of equipment the agent can acquire and successfully utilize is constrained by factors such as its budget and limited manpower (capacity bounds).

Given the above, the goal is to assign to the agent a subset of the available resources that does not violate its capacity constraints, such that the optimal feasible policy under that resource assignment yields the highest expected utility. We can model such an optimization problem as $\langle \mathcal{S}, \mathcal{A}, P, R, \alpha, \mathcal{O}, \rho, \mathcal{C}, \kappa, \widehat{\kappa}, \rangle$, where:

- $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \alpha \rangle$ are the components of a factored MDP, represented in a factored manner, as defined in Section 2.1.
- $\mathcal{O} = \{o\}$ is the set of resource types (e.g., $\mathcal{O} = \{$production equipment, vehicle , $\ldots\}$).
- $\rho : \mathcal{A} \times \mathcal{O} \mapsto \mathbb{R}$ is a function that specifies the resource requirements of all actions; $\rho(a, o)$ defines how much of resource o is needed for action a (e.g., $\rho(a, \text{vehicle}) = 1$ means that action a requires one vehicle to be executable).
- $\mathcal{C} = \{c\}$ is the set of capacities of our agent (e.g., $\mathcal{C} =$ space, money, manpower, $\ldots\}$).
- $\kappa : \mathcal{O} \times \mathcal{C} \mapsto \mathbb{R}$ is a function that specifies the capacity costs of resources; $\kappa(o, c)$ defines how much of capacity c a unit of resource o consumes (e.g., $\kappa(\text{vehicle}, \text{money}) = \$50000$ defines the cost of a vehicle).
- $\widehat{\kappa} : \mathcal{C} \mapsto \mathbb{R}$ specifies the upper bounds on capacities; $\widehat{\kappa}(c)$ gives the upper bound on capacity c (e.g., $\widehat{\kappa}(\text{money}) =$ \$1,000,000 defines the agent's budget constraints).

Given the above, the agent's optimization problem is to find the set of resources whose capacity costs satisfy the constraints imposed by $\widehat{\kappa}$, such that the best realizable policy, given these resources (as induced by action resource requirements ρ) yields the highest expected discounted reward. As we have previously shown in [8] (for flat MDPs and binary resource-requirements), this problem is NP-complete.[6] This complexity result still holds for factored MDPs (the models are equivalent in expressiveness and worst-case space requirements), as well as for non-binary resource requirements (binary is a special case). Furthermore, in [8] we presented a reduction of the problem to a mixed integer linear program (MILP). We now develop an analogous reduction for the factored model of the single-agent capacity-constrained MDPs. This factored MILP will serve as the basis for the resource-allocation algorithm in the multiagent setting, discussed in Section 4.

Given a flat occupation measure x, the agent's capacity constraints can be formulated as:

$$\sum_o \kappa(o, c) \max_a \Big\{ \rho(a, o) H\big(\sum_s x(s, a) \big) \Big\} \leq \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}, \tag{12}$$

[6]It is essentially a multidimensional KNAPSACK problem with item values defined by the MDP.

where H is the Heaviside "step" function of a nonnegative argument, defined as

$$H(z) = \begin{cases} 0 & z = 0; \\ 1 & z > 0. \end{cases}$$

To describe (12) in turns of marginal occupation measures, we use the following Lemma.

LEMMA 1. *Consider a set of marginal occupation measures $\{\xi_j\}$ that are globally consistent and a corresponding policy $\pi(\mathbf{z}, a)$ computer via Algorithm 1. Then*

$$\forall \mathbf{z}, a : \quad \pi(\mathbf{z}, a) > 0 \iff \exists j : \quad \xi_j(\mathbf{z}_j, a) > 0, \tag{13}$$

i.e., the policy prescribes non-zero probability of executing action a in state $\mathbf{z}$ if and only if there is a marginal that prescribes a non-zero probability to that action in same state.

PROOF. The proof is essentially be definition. The " $\Rightarrow$ " direction follows from how policy π is constructed in Algorithm 1: in order for the global distribution to assign a non-zero probability to some event, there must exist a marginal that assigns a non-zero probability to a subset of features corresponding to the same event. The " $\Leftarrow$ " direction follows from global consistency of the marginals. Indeed, assume the opposite: $\xi_j(\mathbf{z}_j, a) > 0$ and $\pi(\mathbf{z}, a) = 0$. This would violate the following consistency constraint:

$$(1 - \gamma)^{-1} \xi_j(\mathbf{z}_j, a) = \sum_{\widetilde{\mathbf{z}_j}} \pi(\mathbf{z}_j, a),$$

where $\widetilde{\mathcal{Z}_j} = \mathcal{Z} \setminus \mathcal{Z}_j$ are the features not in the scope of ξ_j. □

It immediately follows from Lemma 1 that the flat constraints (12) can be equivalently represented on the marginal occupation measures ξ:

$$\sum_o \kappa(o, c) \max_a \Big\{ \rho(a, o) H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) \big) \Big\} \leq \widehat{\kappa}(c), \forall c \in \mathcal{C}. \tag{14}$$

The difficulty is that these constraints are non-linear due to the maximization over a and the Heaviside function H. To get rid of the first source of nonlinearity, let us observe that a constraint with a single max over a finite set of discrete values can be trivially linearized by expanding out the set over which the maximization is taken:

$$\max_{z \in \mathcal{Z}} f(z) \leq a \iff f(z) \leq a, \quad \forall z \in \mathcal{Z}.$$

In (14), we have a finite sum of max's, but it can also be linearized by observing that the inequality

$$\sum_i^n g(u_i) \max_{z \in \mathcal{Z}} f(z, u_i) = g(u_1) \max_{z \in \mathcal{Z}} f(z, u_1) + \ldots + g(u_n) \max_{z \in \mathcal{Z}} f(z, u_n) \leq a$$

is equivalent to the system of $|\mathcal{Z}|^n$ linear inequalities:

$$g(u_1) f(z_1, u_1) + g(u_2) f(z_2, u_2) + \ldots + g(u_2) f(z_2, u_2) \leq a, \quad \forall z_1, z_2, z_n \in \mathcal{Z}.$$

Applying this to the constraints in (14), we can express the original system of $|\mathcal{C}|$ nonlinear constraints:

$$\sum_o \kappa(o, c) \max_a \Big\{ \rho(a, o) H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) \big) \Big\} \leq \widehat{\kappa}(c), \forall c \in \mathcal{C},$$

as the following system of $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ linear constraints:

$$\sum_o \kappa(o,c)\rho(a_o,o)H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a_o)\big) \le \widehat{\kappa}(c), \quad \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}. \tag{15}$$

Notice that this straightforward way of eliminating the maximization exponentially increases the number of constraints, because the above expansion enumerates all possible actions for each resource: it enumerates policies where each resource o is used by action a_1, where it is used by action a_2, action a_3, etc. However, in many problems not all resources are used by all actions. In such cases, most of the above constraints become redundant, and the number of constraints can be reduced from $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ to $|\mathcal{C}|\prod_o |\mathcal{A}_o|$, where $\mathcal{A}_o$ is the number of actions that use resource o. Furthermore, in a special case, where the resource requirements of actions are binary, the number of constraints can be reduced much more significantly, as discussed in Section 3.1.

To linearize the Heaviside function, we augment the original optimization variables x with a set of $|\mathcal{A}|$ binary variables $\Delta(a) \in \{0,1\}$, where

$$\Delta(a) = H\Big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a_o)\Big). \tag{16}$$

In other words, $\Delta(a)$ is an indicator variable that shows whether action a is used in the policy with non-zero probability. Using Δ and expanding the max as above, we can rewrite the resource constraints in (14) as:

$$\sum_o \kappa(o,c)\rho(a_o,o)\Delta(a_o) \le \widehat{\kappa}(c), \quad \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}, \tag{17}$$

which are linear in Δ. We can then synchronize Δ and each of the marginal occupation measures ξ_j via the following linear inequalities:

$$\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a)(1-\gamma) \le \Delta(a), \qquad \forall a \in \mathcal{A}. \tag{18}$$

Indeed, $\xi_j(\mathbf{z}_j,a)(1-\gamma) \le 1$, because $\sum_{\mathbf{z}_j,a} \xi_j(\mathbf{z}_j,a) = (1-\gamma)^{-1}$, which ensures that when any marginal $\xi(\cdot,a)$ assigns a non-zero probability to action a, this drives the corresponding $\Delta(a)$ to be 1. On the other hand, when all $\xi(\cdot,a)$ are 0 for a given a, both $\Delta(a)=0$ and $\Delta(a)=1$ satisfy (18), in which case $\Delta(a)$ will be driven down to $\Delta(a)=0$ by the capacity constraints (if they are binding; if they are not, the value of $\Delta(a)$ is irrelevant).

Putting it all together, the single-agent factored MDP with resources and capacity constraints can be formulated as an MILP by augmenting the LP (11) with binary variables $\Delta(a)$ and constraints (17) and (18).

3.1 Binary Resource Requirements

As a special case of the problem discussed in the previous section, consider a domain where the resource requirements are binary: $\rho(a,o) = \{0,1\}$. Under these conditions the single-agent MILP (11,17,18) can be simplified significantly.

First, observe that, when $\rho(a,o) = \{0,1\}$, the total resource requirements of a policy can be simplified as follows:

$$\max_a \Big\{\rho(a,o)H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a)\big)\Big\} = H\Big(\sum_a \rho(a,o) \sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a)\Big).$$

Therefore, instead of introducing $|\mathcal{A}|$ binary variables $\Delta(a)$ that specify whether action a is used in the policy, it suffices to use $|\mathcal{O}|$ variables $\delta(o) \in \{0,1\}$:

$$\delta(o) = H\Big(\sum_a \rho(a,o) \sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a) \le \widehat{\kappa}(c)\Big), \tag{19}$$

where $\delta(o)$ indicates whether the policy requires resource o.

We can then synchronize the marginals ξ and the binary variables $\delta(o)$ similarly to $\Delta(a)$ above, leading to the following equivalent of (17,18) for binary resources:

$$\begin{aligned} &\sum_o \kappa(o,c)\delta(o) \le \widehat{\kappa}(c), && \forall c \in \mathcal{C}; \\ &\sum_a \rho(a,o) \sum_{j=1} \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a) \le \delta(o), && \forall o \in \mathcal{O}. \end{aligned} \tag{20}$$

4. RESOURCE ALLOCATION

We now consider the multiagent problem of resource allocation between several agents, where the resource preferences of the agents are defined by factored MDPs with capacity-constraints, as described in the previous section. As mentioned previously, we assume that agents are weakly-coupled [16], i.e., they only interact through the shared resources, and once the resources are allocated, the agents' transitions and rewards are independent. Also, to simplify the following discussion, we assume that agents' resource requirements are binary (as discussed in Section 3.1), but all results directly map to to the non-binary case by making straightforward substitutions from Section 3.

The problem of distributing a set of indivisible resources $\mathcal{O}$ among a set of agents $\Lambda = \{\lambda\}$ is described as follows:

- $\{\langle \mathcal{S}, \mathcal{A}, P^\lambda, R^\lambda, \alpha^\lambda, \rho^\lambda, \widehat{\kappa}^\lambda \rangle\}$ is the collection of single-agent factored weakly-coupled MDPs, as defined in Section 3. For simplicity, but without loss of generality we assume that all agents have the same state and action spaces $\mathcal{S}$ and $\mathcal{A}$, but each has its own transition and reward functions P^λ and R^λ, initial conditions α^λ, as well as its own resource requirements $\rho^\lambda : \mathcal{A} \times \mathcal{O} \mapsto \{0,1\}$ and capacity bounds $\widehat{\kappa}^\lambda : \mathcal{C} \mapsto \mathbb{R}$.
- $\widehat{\rho} : \mathcal{O} \mapsto \mathbb{R}$ specifies the upper bound on the amounts of the shared resources (this defines the additional bound for the multiagent problem).

Given the above model, the goal is to find an allocation of the available resources among the agents that maximizes the sum of the expected discounted rewards of the agents' optimal policies under the resource allocation. This can be achieved via a straightforward combination of the single-agent MILPs from the previous section, augmented with an additional constraint that ensures that the allocation does not exceed the bounds on the total amounts of resources $\widehat{\rho}$.

Putting it all together, we arrive at an MILP that achieves the goal we set forth in this work – an algorithm for simultaneously solving the resource-allocation and the resource-parameterized policy-optimization problem for the factored

Markov model. The following MILP has, for every agent λ, a set of marginal occupation measures ξ^λ and a set of binary variables $\delta^\lambda(o)$. Each agent has its own basis functions h_k^λ, and therefore its own backprojections g_k^λ.

$$\max \sum_\lambda \sum_a \sum_j^{J_R} \sum_{\mathbf{z}_{r_j}^\lambda} r_j^\lambda(\mathbf{z}_{r_j}^\lambda, a)\xi_{r_j}^\lambda(\mathbf{z}_{r_j}^\lambda, a) \qquad \text{s.t.:}$$

$$\sum_a \xi_{h_k}^\lambda(\mathbf{z}_{h_k}^\lambda, a)h_k^\lambda(\mathbf{z}_{h_k}^\lambda) - \gamma \sum_a g_k^\lambda(a)\xi_{g_k}^\lambda(\mathbf{z}_{h_k}^\lambda, a) = \sum_{\mathbf{z}_{h_k}} \alpha^\lambda(\mathbf{z}_{h_k}^\lambda)h_k^\lambda(\mathbf{z}_{h_k}^\lambda), \quad \forall k \in [1, K],\ \lambda \in \Lambda;$$

$$\sum_{\widetilde{\mathbf{z}_j^\lambda}} \xi_j^\lambda(\mathbf{z}_j^\lambda, a) = \sum_{\widetilde{\mathbf{z}_i^\lambda}} \xi_i^\lambda(\mathbf{z}_i^\lambda, a), \quad \forall \lambda \in \Lambda, a \in \mathcal{A}, \xi_j^\lambda, \xi_i^\lambda, \mathbf{z}_j^\lambda, \mathbf{z}_i^\lambda, \widetilde{\mathcal{Z}_i^\lambda} = \mathcal{Z}_j^\lambda \setminus \mathcal{Z}_i^\lambda,\ \widetilde{\mathcal{Z}_j^\lambda} = \mathcal{Z}_i^\lambda \setminus \mathcal{Z}_j^\lambda;$$

$$\sum_a \sum_{\mathbf{z}_j^\lambda} \xi_j^\lambda(\mathbf{z}_j^\lambda, a) = (1-\gamma)^{-1}, \quad \forall j,\ \mathbf{z}_j;$$

$$\sum_o \kappa(o, c)\delta^\lambda(o) \le \widehat{\kappa}^\lambda(c), \quad \forall \lambda \in \Lambda, c \in \mathcal{C};$$

$$\sum_{j=1}^{J_e} \sum_a \sum_{\mathbf{z}_{\rho_j}^\lambda} \rho_j^\lambda(\mathbf{z}_{\rho_j}^\lambda, a)\xi_{\rho_j}^\lambda(\mathbf{z}_{\rho_j}^\lambda, a) \le \delta^\lambda(o), \quad \forall \mathbf{z}_j^\lambda, o; \tag{21}$$

As in the flat-MDP model of [8], this MILP has $|\Lambda||\mathcal{O}|$ binary variables, which is an exponential reduction from a naive resource-allocation scheme that has a decision variable per resource bundle (thus requiring on the order of $|\Lambda|2^{|\mathcal{O}|}$ variables). Furthermore, the number of occupation-measure variables can be drastically reduced in this factored MILP, compared to the flat variant. The actual gains in efficiency and the associated losses in quality depend on what basis functions h_k are used and on how well-structured the MDP is (which ultimately defines the domain sizes of ξ).

5. EXPERIMENTAL RESULTS

We investigated the behavior of the resource-allocation MILP (21) on the factored "SysAdmin" problem [12], which we extended to include resources and capacities.

The problem involves a network of N computers, each of which might fail with probability that depends on the status of the neighboring computers. The job of the decision-maker (the SysAdmin) is to keep the network running. The state of the system is defined by a set of N binary features, each corresponding to the status of a computer. At each time step the SysAdmin can reboot at most one computer and receives a reward for every computer that is functioning properly. We used a network with a unidirectional-ring topology, where each computer influenced and was influenced by the status of a single other computer. The parameters of the transition and reward model are described in detail in [12].

We extended the domain to the multiagent setting with resources as follows. Each SysAdmin agent was in charge of running its own network of N computers. In our domain, computer failures were caused by more serious malfunctions (e.g., a virus infection) that could not be fixed by simply rebooting the computer, and thus performing the recovery actions required resources (e.g., specialized software, hardware, or trained personnel). Further, the computers were heterogeneous (mail servers, web servers, data storage, etc.), and thus the recovery of each computer required a different subset of the resources. The total quantity of the resources was limited, and each SysAdmin also had a limited budget, which constrained the resources it could acquire.

For a local MDP with N computers, there were $|\mathcal{O}| = N$ resources, and each recovery action required two resources, which were randomly selected at the time of problem generation. There was a single capacity cost c (money) and all resources had a unit capacity cost $\kappa(o, c) = 1$.

A thorough empirical analysis of ALP solutions to the SysAdmin problem is presented in [12], and it discusses the quality of solutions and the computational complexity of solving the ALP for various network topologies and different ways of selecting the basis functions. In particular, the experiments in [12] demonstrate that using basis functions over singletons of features leads to high-quality approximations at a very low computational cost. Therefore, we used the same basis functions and focused on evaluating the computational complexity of the resource-allocation MILP (21).

Figure 2a shows the running time of the MILP (21) as a function of the number of state features (computers). As can be seen, the factored MDP approach allows us to easily scale the resource-allocation problem up to MDPs with more than 10^{15} states. Figure 2b demonstrates the corresponding exponential gain in efficiency of the factored MILP (21), compared to the one with a flat-MDP model. Note that the flat MILP [8] in itself commonly achieves exponential speedup over a straightforward resource-allocation mechanism that enumerates all resource bundles [9].

Finally, Figure 2c shows the complexity of the factored MILP as a function of the number of agents. The graph demonstrates that although the number of binary variables in the MILP is proportional to the number of agents, the approach shows promise for scaling to large numbers of agents.

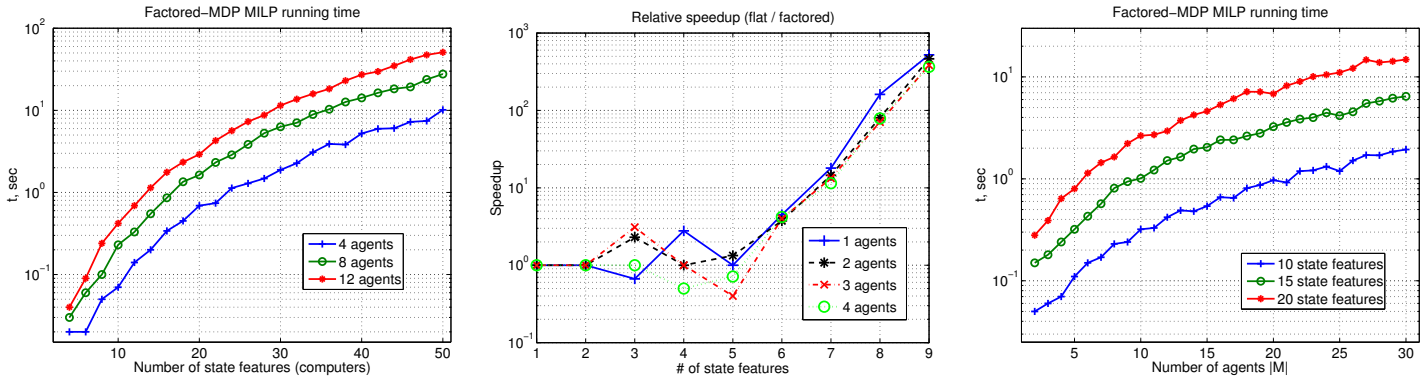
Overall, the trends in these results support the analytical claims about the efficiency gains of using factored models and ALP in the resource-allocation problem.

6. CONCLUSIONS

Recent work on resource-allocation problems has demonstrated that it can be extremely beneficial from the standpoint of computational efficiency to have a good model of the processes that define agents' preferences over resources. For a realistic agent that has to deal with uncertainty, Markov decision processes with action spaces that depend on the resources available to the agent provide a natural model of the agent's utility-formation mechanism. Moreover, resource preferences induced by MDPs are well-structured, which enables the construction of resource-allocation mechanisms that are very computationally efficient[8, 9].

We presented a mechanism that fruitfully exploits the structure in agents' utility functions that is due to the underlying MDPs, as well as the structure within the MDPs themselves (via an approximation). For well-structured MDPs, the latter can lead to another order of exponential speedup over flat MDP methods, as supported by our experiments.

Some interesting directions of our ongoing and future work in this area include exploring factored resource-allocation MILPs on cluster graphs that are not junction trees, as well as looking at other types of structure in agent's MDPs (e.g., context-specific structure [12]).



(a) (b) (c)

Figure 2: Efficiency of resource allocation with factored MDPs. (a): Running time of factored MILP as a function of the number of state features. (b): Relative speedup of factored MILP, compared to the flat MILP (ratio of running times). (c): Running time of the factored MILP as a function of the number of agents.

7. ACKNOWLEDGMENTS

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.

8. REFERENCES

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[2] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[3] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on AI*, pages 1104–1111, 1995.

[4] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *AI*, 121(1-2):49–107, 2000.

[5] D. de Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[6] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6), 2003.

[7] D. A. Dolgov and E. H. Durfee. Graphical models in local, asymmetric multi-agent Markov decision processes. In *Proc. of the 3rd Int. Joint Conf on Autonomous Agents and Multiagent Systems*, 2004.

[8] D. A. Dolgov and E. H. Durfee. Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, pages 315–324, 2004.

[9] D. A. Dolgov and E. H. Durfee. Computationally efficient combinatorial auctions for resource allocation in weakly-coupled MDPs. In *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, 2005.

[10] D. A. Dolgov and E. H. Durfee. Symmetric primal-dual approximate linear programming for factored MDPs. In *Proc. of the Ninth Int. Symposium on AI and Math. (AI&M-06)*, January 2006.

[11] C. Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, CS Dept., Stanford University, August 2003.

[12] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of AI Research*, 19:399–468, 2003.

[13] M. I. Jordan. Graphical models. *Statistical Science (Special Issue on Bayesian Stat.)*, 19:140–155, 2004.

[14] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. of the Sixteenth International Conference on Artificial Intelligence IJCAI-99*, pages 1332–1339, 1999.

[15] K. Larson and T. Sandholm. Mechanism design and deliberative agents. In *Proc. of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 650–656. ACM Press, 2005.

[16] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *AAAI/IAAI*, pages 165–172, 1998.

[17] R. Patrascu, P. Poupart, D. Schuurmans, C. Boutilier, and C. Guestrin. Greedy linear value-approximation for factored Markov decision processes. In *Proc. of the 18th National Conf. on AI*, pages 285–291, 2002.

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

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

[20] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568 582, 1985.