Interaction Structure and Dimensionality in Decentralized Problem Solving

Martin Allen, Marek Petrik, and Shlomo Zilberstein
Computer Science Department
University of Massachusetts
Amherst, MA 01003
{mwallen, marek, shlomo}@cs.umass.edu

University of Massachusetts, Amherst, Computer Science Department Technical Report #UM-CS-2008-11

Abstract

Decentralized Markov Decision Processes are a powerful general model of decentralized, cooperative multi-agent problem solving. The high complexity of the general problem leads to a focus on restricted models. While the worst-case hardness of such reduced problems is often better, less is known about the actual expected difficulty of given instances. We show tight connections between the structure of agent interactions and the essential dimensionality of various problems. Bounds can be placed on the difficulty of solving problems, based upon restrictions on the type and number of interactions between agents. These bounds arise from a bilinear programming formulation of the problem; from such a formulation, a more compact reduced form can be automatically generated, and the original problem can be rewritten to take advantage of the reduction. These results are of theoretical and practical importance, improving our understanding of multi-agent problem domains, and paving the way for methods that reduce the complexity of such problems by limiting the degree of interaction between agents.

Introduction

Decentralized Markov decision processes (Dec-MDPs) are an extension of the basic MDP framework to distributed, cooperative problems. The model is more general, and more complex, than that of *multiagent MDPs* (MMDPs, see (Boutilier 1999)). In the latter, each agent observes the complete system state at every point in time, and policies can be generated from a fundamentally centralized point of view. In a Dec-MDP, on the other hand, each agent possesses only some local, unshared information, and must often act without full knowledge of what others observe, or plan to do. Finding a globally optimal policy for general Dec-MDPs is NEXP-complete (Bernstein *et al.* 2002). Optimal solution algorithms face doubly-exponential growth in necessary space and time, rendering even simple problems intractable. The first-known optimal method uses dynamic programming to generate finite-horizon policies, applying iterated pruning techniques to reduce the number considered (Hansen, Bernstein, & Zilberstein 2004). However, such basic pruning does not make the general problem tractable; even for a very simple problem, the method cannot generate policies beyond a handful of time-steps. Similar results have been reported with respect to top-down methods employing heuristic search: again, only the smallest problems can be solved (Szer & Charpillet 2005; Szer, Charpillet, & Zilberstein 2005). A good overview can be found in Seuken and Zilberstein (2005).

Indeed, such problems are hard to solve even under other criteria than global optimality. Rabinovich *et al.* (2003) show that ϵ-approximate solutions are NEXP-hard. While locally optimal methods have been devised to deal with problem complexity (e.g., Nair *et al.* (2003)), no sharp guarantees can be given about overall output quality. Koller and Megiddo (1992) showed that even finding Nash equilibria in such problems is NP-Hard. Other approaches isolate special, simpler sub-classes. Decentralized MDPs with independent transition-functions are only NP-complete, and specialized algorithms solve many reasonably-sized problems (Becker *et al.* 2004). More special cases have been considered by such as Kim *et al.* (2006).

We concentrate here on the problem of reducing the dimensionality and complexity of certain such special problems, with limited interactions between agents. In the decentralized MDP domain, Shen *et al.* (2006) suggest that complexity of a decentralized problem increases with the "degree of interaction" between agents. We develop these ideas in one particular possible direction, specifying special cases in terms of a fixed number of *events* and *constraints on joint reward*, as defined below. We describe one method of isolating the essential dimensionality of such Dec-MDPs via formulation as separable bilinear programs. This leads to some new results. We show how the bilinear programming version of the problem can be converted back into the event-based structure, and that doing so often reduces (and provably never increases) a key factor governing solution algorithm performance.

Outline of the Paper

We begin by defining the specific type of Dec-MDP framework used in this work, and outlining the shared reward constraint structure that is the main source of problem complexity for such domains. Following that, we describe how such problems can be formulated and solved as bilinear programs, and reviews a method for compacting the problem to its essential dimensions. Next, we present proofs that demonstrate connections between the constraint structure and the essential dimensionality of a problem instance; it is shown how dimension compactification can be used to generate a compact constraint structure, reducing the hardest aspect of the problem as much as possible. Finally, we explore the implications of this work.

Decentralized MDPs

As already outlined, the NEXP-hardness of the general Dec-MDP problem translates into practical difficulties solving even the simplest fully decentralized instances, and leads to interest in techniques applicable to special sub-cases. We focus on a class of problems first introduced by Becker, Lesser, and Zilberstein (2004). In such domains, agents operate on Markov processes that are independent, but for shared influence on the joint reward. Such problems are truly decentralized, as agents only observe and operate on states of their own MDP, but the shared reward means that they must attempt to coordinate despite lacking information about observations and actions of other agents involved. These problems are defined based on single-agent MDPs.

Definition 1. *A* Markov decision process *is a tuple:*

$$\mathcal{M} = \langle S,\, A,\, P,\, R,\, \Delta_S,\, T\rangle$$

with individual components:

- S *is a finite set of world states.*
- A *is a finite set of available actions.*
- $P(s, a, s')$ *is a state-transition function.*
- $R : (S \times A) \rightarrow \Re$ *is the reward function.*
- Δ_S *is the initial state-distribution.*
- T *is the finite time-horizon of the problem.*

To define the shared reward structure of the multiagent Dec-MDP version, we require the following further notions.

Definition 2. *For any MDP* $\mathcal{M}$*, an* event from $\mathcal{M}$ *is some set of state-action pairs,*

$$\mathcal{E} = \{\langle s, a\rangle_1,\, \langle s, a\rangle_2, \ldots,\, \langle s, a\rangle_m\} \subseteq (S \times A).$$

When $\mathcal{E}$ *is a singleton* $\{\langle s, a\rangle\}$ *we call* $\mathcal{E}$ *a* primitive event, *and we also refer to* $\langle s, a\rangle$ *itself as a (primitive) event.*

This definition is a novel simplification of that given by Becker *et al.* (2004), as we do not require uniqueness conditions present there (although such conditions could be accommodated without affecting the results given here). Our notion of event is to be considered disjunctive, i.e. the event

$$\mathcal{E} = \{\langle s_1, a_1\rangle,\, \langle s_2, a_2\rangle, \ldots,\, \langle s_m, a_m\rangle\} \subseteq (S \times A)$$

can be thought of as a statement to the effect that an agent *performs action* a_1 *in state* s_1 OR *performs action* a_2 *in state* s_2 ... OR *performs action* a_m *in state* s_m.

Definition 3. *For a pair of MDPs* $\mathcal{M}^1$, $\mathcal{M}^2$*, a* reward-constraint on $\mathcal{M}^1$, $\mathcal{M}^2$ *is a triple* $c = \langle \mathcal{E}^1,\, \mathcal{E}^2,\, r_c\rangle$*, where each* $\mathcal{E}^i$ *is an event from* $\mathcal{M}^i$*, and* $r_c \in \Re$.

A reward-constraint is the basis for defining a shared dependency between the two processes $\mathcal{M}^1$ and $\mathcal{M}^2$. Again, such a constraint $\langle \mathcal{E}^1,\, \mathcal{E}^2,\, c\rangle$ can be regarded as a statement to the effect that *if event* $\mathcal{E}^1$ *occurs* AND *event* $\mathcal{E}^2$ *occurs, then the system receives additional reward* r_c. Such structures provide an intuitive definition of shared reward, and naturally describe many domains in which agents are engaged, for instance, in complementary or redundant subtasks. These problems allow separate execution, but can still make the overall system reward a complex function of combined agent behaviors, requiring coordination.

To properly define such a problem, the reward-constraints must obey a particular simple condition, however.

Definition 4. *Let* $\mathcal{C} = \{c_1, \ldots, c_m\}$ *be a set of reward-constraints on some pair of MDPs* $\mathcal{M}^1$, $\mathcal{M}^2$. $\mathcal{C}$ *is* feasible *iff all distinct reward-constraints are non-intersecting:*

$$\begin{aligned}&(\forall\, \langle s, a\rangle^1,\, \langle s, a\rangle^2)(\forall\, c_i,\, c_j)\\ &\qquad \langle s, a\rangle^1 \in \mathcal{E}^1_i \wedge \langle s, a\rangle^1 \in \mathcal{E}^1_j\\ &\qquad\qquad \wedge \langle s, a\rangle^2 \in \mathcal{E}^2_i \wedge \langle s, a\rangle^2 \in \mathcal{E}^2_j \;\Rightarrow\; r_{c_i} = r_{c_j}.\end{aligned}$$

That is, a feasible set of reward-constraints can never assign more than one reward to a pair of primitive events $(\langle s, a\rangle^1,\, \langle s, a\rangle^2)$. Note that such sets need not assign values to all pairs of primitive events; only that each such pair may be assigned at most one supplementary shared reward. Such pairs define the interaction structure in a Dec-MDP.

Definition 5. *For two agents* x *and* y*, a two-agent* decentralized Markov decision process (Dec-MDP) *is a triple*

$$\mathcal{D} = \langle \mathcal{M}^x,\, \mathcal{M}^y,\, \rho\rangle$$

where $\mathcal{M}^x$ *and* $\mathcal{M}^y$ *are MDPs and* ρ*, the* shared-reward structure for $\mathcal{D}$*, is a feasible set of reward-constraints.*

An optimal solution to a Dec-MDP is a pair of deterministic policies, π^x, π^y*, one per agent, maximizing the expected sum of individual rewards (*$R^i \in \mathcal{M}^i$*) and joint reward (*ρ*).*

Note that this defines a proper subclass of the general Dec-MDP (or Dec-POMDP), which do not restrict their connections only to the reward function, and feature dependencies between state-action transitions and state-observations. As defined by Becker *et al.* (2004), these special problems are properly *transition and observation-independent, locally and jointly fully observable Dec-MDPs*; for convenience, we simply refer to them as Dec-MDPs.

While these are but restricted versions of the general class, they are still useful for representing many real-world problems in which agents can work separately, without interfering with one another, but overall value of actions is a function of all the agents together. Examples include domains in which tasks can be divided into components that can be accomplished separately; given uncertainty about progress and outcome of subtasks, such problems still prove challenging.

Indeed, Becker *et al.* (2004) show that solving these Dec-MDPs is NP-complete. While this significantly reduces worst-case complexity from NEXP-hardness, solution can still be quite difficult in practice. They apply a specialized method, the *Coverage Set Algorithm* (CSA), to such problems; they show that it can perform quite well on some cases, although it is not applicable to Dec-MDPs in general. The main hurdle in using the CSA on a given Dec-MDP $\mathcal{D}$ comes from the shared-reward structure ρ: while most of the algorithmic heavy lifting is performed efficiently using linear programming and hill-climbing methods, the algorithm iterates exponentially in the number of reward-constraints, $|\rho|$.

This motivates our current research. As we will show, the structure of ρ is tightly bound to the *dimensionality* of a Dec-MDP. As ρ grows, so generally will the dimensionality. We give bounds on this growth, and then show how techniques for dimensionality compactification reduce the problem to only its essential (or dominant) dimensions. Further, we show that such techniques can be used to generate new, often much smaller, shared-reward structures. These results provide firm connections between dimensionality and reward interactions in a Dec-MDP, and can reduce the complexity of the constraint structure, thus improving performance for algorithms like CSA that are highly sensitive to $|\rho|$.

An example Dec-MDP

We present a simple example Dec-MDP, to help make things clear. In this domain $\mathcal{D}$, two agents x and y must make some delivery of goods of type a and b to one of two customers, c_1 and c_2. For each agent, the individual action-outcomes and rewards are given by two MDPs, $\mathcal{M}^x$ and $\mathcal{M}^y$. The particular details are unimportant; we simply note that specifying the MDPs separately, with separate transition and reward functions, means that they can be regarded as wholly independent from the point of view of each agent. Further, the techniques we present are able to easily solve each agent's independent sub-problem, based on the transition probabilities and reward functions of the individual MDPs.

However, the delivery problem contains one important source of dependency: the first customer, c_1, is willing to (1) pay \$2 extra for receiving two items, and (2) will give an additional \$2 bonus if it actually receives two different types of items. This shared bonus can be given in terms of the following feasible set of events (writing $\langle c_i,\, dl_j\rangle^k$ for the event of agent k delivering item type j to customer c_i).

$$\begin{aligned}\rho = \big[\, &\langle\langle c_1,\, dl_a\rangle^x,\, \langle c_1,\, dl_a\rangle^y,\, 2\rangle,\\ &\langle\langle c_1,\, dl_a\rangle^x,\, \langle c_1,\, dl_b\rangle^y,\, 4\rangle,\\ &\langle\langle c_1,\, dl_b\rangle^x,\, \langle c_1,\, dl_a\rangle^y,\, 4\rangle,\\ &\langle\langle c_1,\, dl_b\rangle^x,\, \langle c_1,\, dl_b\rangle^y,\, 2\rangle\big]\end{aligned}$$

The shared-reward structure is therefore as shown in Table 1, which tracks the extra reward to be gained for the various state-action pairs for each agent. In general, any shared-reward structure can be represented in such a matrix form, where each entry corresponds to the shared-reward bonus for the corresponding pair of primitive events. Obviously, for any pair of MDPs $\mathcal{M}^x$ and $\mathcal{M}^y$, the size of this matrix representation is $|S^x|\,|A^x| \times |S^y|\,|A^y|$. Note also that this matrix only describes the shared reward for the relevant states and actions; there may be many more state-action pairs that play no role in the joint reward, but are part of the independent single-agent MDP planning problems. The policy for such a problem will involve actions for each agent in its own sequential planning problem—which might involve such things as planning local routes to various deliveries, for instance—while also maximizing overall reward based on the shared constraints.

$x\backslash y$	$\langle c_1, dl_a\rangle^y$	$\langle c_2, dl_a\rangle^y$	$\langle c_1, dl_b\rangle^y$	$\langle c_2, dl_b\rangle^y$
$\langle c_1, dl_a\rangle^x$	2	0	4	0
$\langle c_2, dl_a\rangle^x$	0	0	0	0
$\langle c_1, dl_b\rangle^x$	4	0	2	0
$\langle c_2, dl_b\rangle^x$	0	0	0	0

Table 1: The shared-reward structure for the simple delivery problem for agents x and y.

Bilinear Programs and Dec-MDPs

Petrik and Zilberstein (2007) have demonstrated how this class of Dec-MDPs can be represented and solved as separable bilinear programs (for more details on separability, see Horst & Tuy (2003)). We have simplified that presentation somewhat here. For Dec-MDP $\mathcal{D} = \langle \mathcal{M}^x,\, \mathcal{M}^y,\, \rho\rangle$, we define the equivalent bilinear program:

$$\begin{aligned}\text{maximize}\quad & r_1^T x + x^T R y + r_2^T y\\ \text{subject to}\quad & A_x x = \Delta_{S^x} \qquad x \geq 0\\ & A_y y = \Delta_{S^y} \qquad y \geq 0\end{aligned}\tag{1}$$

Such a program is defined similarly to the dual linear program form for single-agent MDPs (see Puterman (2005)). The vectors x and y are composed of variables corresponding to the possible state-action pairs from the two MDPs; we write $x(s, a)$ for the state-action pair corresponding to $s \in S^x$ and $a \in A^x$, for instance. Each linear reward-vector r_i in the objective function is simply the individual reward, taken from $R^i \in \mathcal{M}^i$. The matrices A_i encode state-visitation information, so that the multiplication in the constraints generates the original state distribution Δ_{S^i}, preserving total flow in the system for each state; for instance, multiplying vector x by A_x yields, for any $s \in S^x$,

$$\sum_{a\in A^x} x(s,\, a) - \sum_{s'\in S^x}\sum_{a'\in A^x} P(s\,|\,s', a')x(s', a') = \Delta_{S^x}(s).$$

Note that all elements so far are linear. However, we get generally non-linear behavior in the objective function via the matrix R, encoding the shared-reward structure of the Dec-MDP. This leads to NP-hardness in solving the overall problem, although methods have been found that work quite well in practice. Once the mathematical program has been solved, the agent policies for agent x can be extracted by letting $\pi^x(s) = a$ iff $x(s, a) > 0$, and similarly for y.

While any general Dec-MDP can be represented bilinearly in principle, it is only practical for either very small general problems, or for the special nearly-independent form given here. Koller and Megiddo (1992; 1996) consider the representation of extensive-form games in the form of linear complementarity problems (LCP, see (Cottle, Pang, & Stone 1992)). Mangasarian (1995) shows how such LCPs can in turn be represented as a separable bilinear program. Unfortunately, for the general problem class, this two-stage reduction is of little practical use: variables take the form of possible *action-observation sequences* for each agent, and thus the resulting bilinear formulation is exponentially large in the size of the original Dec-MDP. (This is not a failure of the method, per se; evidently, given the NEXP-hardness of the original general class, this is unavoidable by any method in the worst case). Still, such reductions are possible in principle, and may lead to useful methods in some general cases. A similar approach is used by Aras *et al.* (2007), who perform a similar sequence-form reduction in order to solve Dec-MDPs via mixed integer programs. Similarly, Amato *et al.* (2006; 2007) employ quadratically-constrained linear and non-linear methods to solve the general problem. These methods extend the ability to solve some general-form Dec-MDPs (and Dec-POMDPs), but are still limited by their inherent complexity.

The bilinear approach is particularly useful in the special case described here, where R is simply a reward matrix on state-action pairs. Especially interesting is the possibility for dimensionality reduction. As we show, this technique allows us to develop automated methods for reducing the size of reward-constraint formulations of Dec-MDPs, providing new hope for methods like CSA that scale poorly.

Dimensionality Reduction

We will refer to the *dimensionality* of a bilinear program for a Dec-MDP as in (1), by which we mean n, the size of the y-dimension of shared-reward matrix R. As we will describe, this dimensionality has been observed to dominate the complexity of solving such programs, and we will prove that it is tightly bound to the shared reward constraint structure. Note that in what follows, we assume that the original matrix R is a square $(n \times n)$ matrix, i.e. that x and y are both of length n; for two MDPs with differently sized state or action-sets, this can be enforced by padding out the smaller MDP with null actions and null states. This is trivial and convenient. Note also that we could as easily perform all described operations along the x-dimension of R; nothing depends upon y.

Petrik and Zilberstein (2007) prove that a given Dec-MDP can easily and automatically be reduced to its essential dimensions, based on shared-reward matrix R. That is, we can perform the following elementary matrix operations to eliminate all constant dimensions of y (along which the best response for agent y is the same for anything x does):

Eigenvector Generation: Generate the $(n \times n)$ matrix $R^T R$, and calculate the eigenvectors of $R^T R$. Since $R^T R$ is always a symmetric square matrix, these eigenvectors can be written in their orthonormal form.

Divide the Eigenvectors Let F be the matrix with columns formed by all eigenvectors of $R^T R$ with non-zero eigenvalues; let G be the zero-value eigenvectors. Let $[F; G]$ be the matrix of all eigenvectors, with all of F first (otherwise order of columns does not matter). Note that since $R^T R$ is symmetric and $(n \times n)$, $[F; G]$ is also an $(n \times n)$ matrix.

Generate the Inverse: Let $D = [F; G]^{-1}$. It is an elementary fact about the collection of eigenvectors of symmetric, square $R^T R$ that such an inverse exists. Let k be the number of columns in F (i.e., the number of non-zero eigenvectors of $R^T R$), let matrix D_k^T be the first k rows of D^T (i.e., the transposed inverse corresponding to those non-zero eigenvectors), and let matrix D_{k+1}^T be the remaining rows.

Separate Dimensions: Let $y_1 = D_k^T y$ and $y_2 = D_{k+1}^T y$. This separates out those dimensions of y that "matter" in our problem (y_1), from those that do not (y_2). Let $\langle y_1, y_2 \rangle$ be the vector composed of y_2 appended to y_1. (Note that the size of $[y_1; y_2]$ is just the same as the original, $n = |y|$.)

It is now elementary that the following is equivalent to the original mathematical program (1):

$$\begin{aligned} \text{maximize} \quad & r_1^T x + x^T R F y_1 + r_2^T [F; G] \langle y_1, y_2 \rangle \\ \text{subject to} \quad & A_x x = \Delta_{S^x} \\ & A_y [F; G] \langle y_1, y_2 \rangle = \Delta_{S^y} \\ & \quad x \geq 0 \quad y_1 \geq 0 \quad y_2 \geq 0. \end{aligned} \tag{2}$$

It is easy to verify by the construction of y_1 and y_2 as a result of inverse multiplication that $[F; G] \langle y_1, y_2 \rangle = y$, and so this formulation respects the original individual reward function r_2 for agent y, and the original constraints on distribution of states, Δ_{S^y}. What is interesting, however, is that we can replace the original $(n \times n)$ joint-reward matrix R in (1) with the $(n \times k)$ matrix RF here; when k, the dimensionality of F, is small, and $R^T R$ has few non-zero eigenvectors, this can be a substantial savings. Furthermore, we can then go back to the original problem formulation, and replace the reward-constraint structure with a new one, often smaller, as we describe below. This means that we can preserve the often more intuitive structure, based on events, and use algorithms exploiting this sort of structure.

Application to Our Example Problem

To see how this works in practice, let us consider again our simple delivery problem, with a 4-dimensional shared-reward matrix as found in Table 1:

$$R = \begin{bmatrix} 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad R^T R = \begin{bmatrix} 20 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 \\ 16 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The non-zero eigenvectors of $R^T R$ are thus the columns of:

$$F = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix}$$

In this case, further, the inverse-row-matrix $D_k^T = F^T$; it is important to note that this will not hold in general, although D_k^T always exists and is easily calculated. Thus, we have the following (using $y(i, k)$ to abbreviate the event of y giving item k to customer i):

$$RF = \begin{bmatrix} 3\sqrt{2} & -\sqrt{2} \\ 0 & 0 \\ 3\sqrt{2} & \sqrt{2} \\ 0 & 0 \end{bmatrix} \quad y_1 = D_k^T y = \begin{bmatrix} \frac{y(1,a)+y(1,b)}{\sqrt{2}} \\ \frac{y(1,a)-y(1,b)}{\sqrt{2}} \end{bmatrix}$$

Our new joint-reward matrix RF is now 2-dimensional, and has only two y_1-variables, each a linear combination of pre-existing variables. One can easily confirm that the minimized reward function is identical to the original (that is, $x^T R y = x^T R F y_1$), and so the resulting objective function is equivalent to the original. It is also easy to generate remaining components G and y_2, and confirm that all other operations preserve equivalent problem input and output.

For an example like this, the dimensionality reduction is not very surprising; clearly, in the original specification of R, deliveries to the second customer play no role in maximizing the shared reward. No extra reward is received for deliveries to c_2, and the columns $y(2,*)$ and rows $x(2,*)$ are all empty (0). This is not generally the case, however; the method does not amount to simply ignoring columns that are all 0. There will be many cases in which no columns or rows of the original R are empty, and yet we can still compactify.

Furthermore, this example shows an important, and less obvious, new feature of the dimension reduction process, namely the compactification of the overall reward-constraint structure. While prior work has been interested in converting event-based Dec-MDPs into the bilinear formulation solely as a means of solving them, we can now go a step further. That is, we can convert the reduced bilinear form *back into* the reward-constraint formulation, with the possibility of a substantive savings in the size of the structure ρ.

Even though our original problem was very small, we are still able to re-write it using a smaller ρ than was possible before. Comparing the two matrices from the two presentations of the problem:

$$R = \begin{bmatrix} 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad RF = \begin{bmatrix} 3\sqrt{2} & -\sqrt{2} \\ 0 & 0 \\ 3\sqrt{2} & \sqrt{2} \\ 0 & 0 \end{bmatrix}$$

we see that in original matrix R, no non-zero column or row contains any repeated values. Thus, the original shared-reward structure ρ is minimal with respect to its elementary events. That is, ρ needs four distinct entries ρ to specify R. In the case of RF, however, this is not true, since the first column, corresponding to new variable $(y(1,a) + y(1,b))/\sqrt{2} \in y_1$, contains only a single value, $3\sqrt{2}$. It follows that we can write a new reward-constraint structure in terms of the new, compound event-variables in y_1, featuring only 3 entries. Even on this small, nearly minimal example, then, we have reduced the minimum number of constraints necessary to describe the joint-reward structure. For algorithms like CSA, exponential in this value, this can significantly improve performance.

Constraints and Dimensionality

As we now show, these reductions, in both overall dimensionality and size of the minimal reward-structure are not accidental: we can in fact relate the basic properties of the minimal constraint structure for a problem to the dimensionality of its reduced bilinear form, to establish that the reduced form will always be no larger than the original.

We first establish an upper bound upon the *essential dimensionality* of a Dec-MDP, by which we mean the dimensionality of the reward matrix RF in the compactified bilinear form; we write $k[\mathcal{D}]$ for the essential dimensionality, with $k = \#$ columns of RF.

Theorem 1. *Let* $\mathcal{D} = \langle \mathcal{M}^x,\, \mathcal{M}^y, \rho\rangle$ *with*

$$\rho = \left[\, \langle \mathcal{E}_1^x,\, \mathcal{E}_1^y,\, r_1\rangle, \langle \mathcal{E}_2^x,\, \mathcal{E}_2^y,\, r_2\rangle, \ldots \langle \mathcal{E}_m^x,\, \mathcal{E}_m^y,\, r_m\rangle \right]$$

and let

$$Y_\rho = \cup_{i=1}^m \mathcal{E}_i^y.$$

Then $k[\mathcal{D}] \leq |Y_\rho|$.

That is, the essential dimensionality of the problem is bounded on top by the size of the set formed from the union of all y-events in ρ. While this bound may be loose, it can often provide a good guide to the overall essential complexity of a given Dec-MDP $\mathcal{D}$.

Proof. Let the length of our original y-vector be n (so the dimensionality of the unreduced R matrix is also n). Now consider any primitive event $\langle s, a\rangle_j^y \notin Y_\rho$; since this event is not featured in any constraint in ρ, we have that column j of R is entirely 0. Thus, column j of R^TR is entirely 0, and so we have that there exists a unitary vector

$$v_j^0 = [0_1\; 0_2\; \cdots\; 0_{j-1}\; 1_j\; 0_{j+1}\; \cdots\; 0_{n-1}\; 0_n]^T$$

(i.e. 0's in all places, and 1 in place j), which is an eigenvector of R^TR with eigenvalue = 0. For each such $\langle s, a\rangle_j^y \notin Y_\rho$, such a distinct v_j will exist; each will be orthogonal, and the entire collection can be put into orthonormal form. Thus the size of the set G of all 0-value eigenvectors of R^TR will be at least the size of the complement of Y_ρ, $|G| \geq (n - |Y_\rho|)$. Therefore, since $|F| = (n - |G|)$, $k[\mathcal{D}] = |F| \leq |Y_\rho|$ ☐

Thus, for any Dec-MDP $\mathcal{D}$, we can bound the dimensionality of the reduced form in advance. In the worst case, when $Y_\rho = \{y(s,a) \,|\, s \in S^y,\; a \in A^y\}$, this bound will simply be n. Of course, the worst case for compactification is that all eigenvalues of R^TR are non-zero, and dimensionality is in fact n. (This is equivalent to invertibility of R).

A more interesting result concerns the opposite direction, namely bounding the size of the minimal constraint structure for Dec-MDP $\mathcal{D}$ based upon the reduced problem representation. As noted, our example problem can be written using 3 constraints once in compact bilinear form, rather than the original 4. This point can easily be made general.

Fact 1. *Let $\mathcal{D}$ be a Dec-MDP written in reduced form* (2), *with compactified shared-reward matrix RF. For any column i of RF, let $u_i[RF]$ be the number of unique values occurring in that column. Then $\mathcal{D}$ can be written in an equivalent form $\mathcal{D}^-$, using reward structure ρ^- with size:*

$$\left|\rho^-\right| = \sum_{i=1}^{|RF|} u_i[RF].$$

We can see this from our example problem, where the reward structure will be:

$$\rho = \Big[\, \langle \{x(1,a),\, x(1,b)\},\, \frac{y(1,a) + y(1,b)}{\sqrt{2}},\, 3\sqrt{2}\rangle,$$
$$\langle x(1,a),\, \frac{y(1,a) - y(1,b)}{\sqrt{2}},\, -\sqrt{2}\rangle,$$
$$\langle x(1,b),\, \frac{y(1,a) - y(1,b)}{\sqrt{2}},\, \sqrt{2}\rangle \Big]$$

In general, for any column of RF corresponding to a compound event variable $y^- \in y_1$, and any unique value u in that column, reward structure ρ^- requires one constraint. Each such constraint will be of the form

$$c = \langle \mathcal{E}^x,\, y^-,\, u \rangle$$

where $\mathcal{E}^x$ is the set of all state-action pairs $x(s,a)$ corresponding to rows of RF in which value u appears. This allows us to easily bound the general size of the reduced shared-reward structure.

Fact 2. *Let Dec-MDP $\mathcal{D} = \langle \mathcal{M}^x,\, \mathcal{M}^y,\, \rho \rangle$, be written in equivalent form $\mathcal{D}^-$ as just described. We have an upper bound on the size of the reduced shared-reward structure:*

$$\left|\rho^-\right| \leq |S^x|\,|A^x|\,k[\mathcal{D}].$$

Proof. This is a straightforward application of Fact 1. Since the size of the structure is $|\rho^-| = \sum_{i=1}^{|RF|} u_i[RF]$, and the number of unique values in any column of RF is at most $n = |S^x|\,|A^x|$ (i.e., simply the number of rows in RF, equal to the size of vector x). The result is then obvious, since the number of columns in RF is simply the number of columns in F, i.e. the essential dimensionality $k[\mathcal{D}]$. □

Along with these basic bounds, we can also prove something far more significant about the shared-reward structure of a compactified Dec-MDP $\mathcal{D}$. In particular, we can show that by putting $\mathcal{D}$ in the reduced form (2), and then rewriting it in terms of the induced reward structure, we can only reduce the number of constraints required.

Theorem 2. *Let $\mathcal{D}$ be a Dec-MDP with $|\rho| = n$; let $\mathcal{D}^-$ be the compactified bilinear form of the problem, and ρ^- be the resulting constraint structure, as described above. Then we have the following:*

$$\left|\rho^-\right| \leq |\rho|$$

The full proof of Theorem 2 requires two parts. We must show that the original formulation of $\mathcal{D}$ must contain at least one distinct constraint for (i) every column of RF, and (ii) every distinct value in that column. The first is easily shown; here, we prove the second, since it is more interesting.

Proof. Consider any column c of RF, and suppose it contains two distinct values $c_i \neq c_j$. Let $v_c \in F$ be the column eigenvector of F that generated column $c \in RF$ (i.e., $c = Rv_c$). Thus, since

$$c_i = \sum_{k=1}^{n} r_{ik} v_c \quad \text{and} \quad c_j = \sum_{k=1}^{n} r_{jk} v_c$$

there must exist column k^* of R such that $r_{ik^*} \neq r_{jk^*}$ (else $c_i = c_j$). Therefore, in the original problem formulation of $\mathcal{D}$, the specification of R in terms of events will require two separate and distinct constraints:

$$c_1 = \langle \mathcal{E}_1^x = \{\langle s,a\rangle_i^x, \ldots\},\, \mathcal{E}_1^y = \{\langle s,a\rangle_{k^*}^y, \ldots\},\, r_{ik^*}\rangle,$$
$$c_2 = \langle \mathcal{E}_2^x = \{\langle s,a\rangle_j^x, \ldots\},\, \mathcal{E}_1^y = \{\langle s,a\rangle_{k^*}^y, \ldots\},\, r_{jk^*}\rangle.$$

Thus, each distinct value in any column of RF corresponds to at least one constraint in the original problem. □

Thus, the reduction in number of necessary constraints observed for our example problem is no accident. Rather, the three-stage process of (1) conversion into bilinear form, (2) dimensionality reduction, and (3) re-conversion into event-based reward-constraint form, will never increase the size of the problem specification (since it only ever shrinks ρ, and leaves all else alone).

Practical Applications

These techniques are of more than formal interest. Our ongoing research has applied the presented techniques to a number of domains, including the multiagent broadcast-channel and tiger problems, standard benchmarks used, for example, in recent work by Aras *et al.* (2007), and a common Dec-MDP formulation of a Mars rover robot exploration problem, used in the work of Becker, Lesser & Zilberstein (2004). The reduction method has been shown to reduce the number of events necessary to specify a wide range of these domains. We found that in the broadcast domain, dimensionality (and the number of necessary events) is reduced to 3 no matter what the original problem size, providing a potentially very large reduction from the event-based specification. In the rover case, many irrelevant events are eliminated, reducing to one for each site that two rovers both explore, out of many initial events involving all possible sites; additionally, in particular instances the number of events may further be reduced even more significantly, with very little resulting error. Finally, when applied to instances of the decentralized tiger problem, the number of events is reduced by about a factor of 5, from 108 to 20, with a reward loss of at most 2%. Since even linear reductions in the number of events provides exponential possible speed-ups for algorithms like the CSA, this transforms such problem instances from ones that are simply infeasible to those that can be practically solved after all.

Conclusions and Discussion

As we have shown, the reduction process allows us to potentially eliminate constant dimensions for one of agent's actions, and also rewrite the problem in terms of a smaller reward structure. While the method of converting into bilinear program and doing dimensionality reduction was already known, this work is the first to consider how to move back to the original form, and how that affects problem size. This is of theoretical and practical interest.

In analytical terms, this method allows us to reveal the essential structure of dependencies between agents in a Dec-MDP. By converting to the reduced form, find a more minimal set of events suitable for representing a domain. The event-based formulation is very convenient and intuitive, but can be highly inefficient. While simple techniques for merging events exist, they are limited. In fact, as we have shown, problems can be such that there is simply no way of reducing the size of the event formulation, so long as we use state-action pairs. This poses a serious roadblock to the use of methods like the Coverage Set Algorithm, which explicitly iterates based on separate constraints. Our process of reduction allows problems to reduce this size, often dramatically.

Of course, it may be hard to look at a linear combination of state-action pairs, as generated by our method, and see how this relates to the structure of the original problem. That is, it is difficult to interpret the weighted combination of elementary events produced by compactification. Our ongoing work concerns Dec-MDPs for which this problem of interpretation is much easier. In such cases, the partial inverse matrix D_k^T is of a special form, and our new reduced problem can be expressed in terms of simple events from the original problem, while still reducing the maximum number of constraints generated. These sorts of extensions have many possible practical applications, since they can provide ways of automatically reconfiguring large and complex multiagent system specifications, eliminating unnecessary events and reward-constraints from consideration.

Finally, we note that is straightforward to extend this approach to problems with more than two agents, if rewards depend on pairs of agents and the dependency graph is bipartite. In this case, the problem is again formulated bilinearly. An extension to general multiagent problems is more problematic. A possible approach may rely on a multilinear program formulation, and then applying a tensor version of singular value decomposition (SVD). The problem is that in general, these methods are often NP-complete, unlike two-dimensional SVD, which can be done in polynomial time. Clearly, this is an important next step.

Acknowledgments

This work was supported in part by the Air Force Office of Scientific Research under Grant No. FA9550-05-1-0254 and by the National Science Foundation under Grants No. IIS-0328601. The findings and views expressed in this paper are those of the authors and do not necessarily reflect the positions of the sponsors.

References

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2006. Optimal fixed-size controllers for decentralized POMDPs. In *Proceedings of the Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*. Held in conjunction with the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2007. Solving POMDPs using quadratically constrained linear programs. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2418–2424.

Aras, R.; Dutech, A.; and Charpillet, F. 2007. Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 18–25.

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2004. Solving transition independent decentralized MDPs. *Journal of Artificial Intelligence Research* 22:423–455.

Becker, R.; Lesser, V.; and Zilberstein, S. 2004. Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 302–309.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27(4):819–840.

Boutilier, C. 1999. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 478–485.

Cottle, R. W.; Pang, J.-S.; and Stone, R. E. 1992. *The Linear Complementarity Problem*. Academic Press.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 709–715.

Horst, R., and Tuy, H. 2003. *Global Optimization: Deterministic Approaches*. Springer.

Kim, Y.; Nair, R.; Varkantham, P.; Tambe, M.; and Yokoo, M. 2006. Exploiting locality of interaction in networked distributed POMDPs. In *Proceedings of the AAAI Spring Symposium on Distributed Planning and Scheduling*.

Koller, D., and Megiddo, N. 1992. The complexity of zero-sum games in extensive form. *Games and Economic Behavior* 4(4):528–552.

Koller, D., and Megiddo, N. 1996. Finding mixed strategies with small supports in extensive form games. *International Journal of Game Theory* 25(1):73–92.

Mangasarian, O. L. 1995. The linear complementarity problem as a separable bilinear program. *Journal of Global Optimization* 12:1–7.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 705–711.

Petrik, M., and Zilberstein, S. 2007. Anytime coordination using separable bilinear programs. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, 750–755.

Puterman, M. L. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley & Sons, Inc.

Rabinovich, Z.; Goldman, C. V.; and Rosenschein, J. S. 2003. The complexity of multiagent systems: The price of silence. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 1102–1103.

Seuken, S., and Zilberstein, S. 2005. Formal models and algorithms for decentralized control of multiple agents. Technical Report UM-CS-2005-68, University of Massachusetts, Amherst, Department of Computer Science.

Shen, J.; Becker, R.; and Lesser, V. 2006. Agent Interaction in Distributed MDPs and its Implications on Complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 529–536.

Szer, D., and Charpillet, F. 2005. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *Proceedings of the Sixteenth European Conference on Machine Learning*, 389–399.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 576–583.