Dynamic Programming for Partially Observable Stochastic Games

Daniel S. Bernstein
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
bern@cs.umass.edu

Eric A. Hansen
Dept. of CS and Engineering
Mississippi State University
Mississippi State, MS 39762
hansen@cse.msstate.edu

Shlomo Zilberstein, Christopher Amato
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
{shlomo,cjamato}@cs.umass.edu

Abstract

We develop an exact dynamic programming algorithm for partially observable stochastic games (POSGs). The algorithm is a synthesis of dynamic programming for partially observable Markov decision processes (POMDPs) and iterative elimination of dominated strategies in normal form games. We prove that it iteratively eliminates very weakly dominated strategies without first forming the normal form representation of a finite-horizon POSG. This is the first dynamic programming algorithm for iterative strategy elimination in these types of games. For the special case in which agents share the same payoffs, the algorithm can be used to find an optimal solution. We present preliminary empirical results and discuss ways to further exploit POMDP theory in solving POSGs.

Introduction

The theory of stochastic games forms the basis of much recent work on multi-agent planning and learning (Littman 1994; Boutilier 1999; Brafman & Tennenholtz 2002). A stochastic game can be viewed as an extension of a Markov decision process (MDP) in which there are multiple agents with possibly conflicting goals, and the joint actions of agents determine state transitions and rewards. Much of the literature on stochastic games assumes that agents have complete information about the state of the game; in this respect, it generalizes work on completely observable MDPs. In fact, exact dynamic programming algorithms for stochastic games closely resemble exact dynamic programming algorithms for completely observable MDPs (Shapley 1953; Filar & Vrieze 1997).

Although there is considerable literature on the partially observable Markov decision process (POMDP), corresponding results for the partially observable stochastic game (POSG) are very sparse, and no exact dynamic programming algorithm for solving POSGs has been previously described. Algorithms for POSGs could prove to be useful in solving complex, multi-agent problems, such as those arising from the coordination of robot teams. In particular, they could help in extending previous work on applying POMDPs to single robot problems (Pineau, Gordon, & Thrun 2003).

In this paper, we show how to generalize the dynamic programming approach to solving POMDPs in order to develop a dynamic programming algorithm for POSGs. The difficulty in developing this generalization is that there is not a shared belief state that can be used to define a dynamic programming recursion; each agent can have a different belief state. Our approach is to combine dynamic programming for POMDPs with iterative elimination of dominated strategies in normal form games. We generalize the notion of belief state to include uncertainty about the other agents' future plans. This allows us to define a *multi-agent dynamic programming operator*.

We show that the resulting dynamic programming algorithm corresponds to a type of iterative elimination of dominated strategies in the normal form representation of finite-horizon POSGs. This is the first dynamic programming algorithm for iterative strategy elimination. For the special case where all agents share the same payoff function, our dynamic programming algorithm can be used to find an optimal solution.

Related work

A finite-horizon POSG can be viewed as a type of extensive game with imperfect information (Kuhn 1953). Though much work has been done on such games, very little of it is from a computational perspective. This is understandable in light of the negative worst-case complexity results for POSGs (Bernstein *et al.* 2002). A notable exception is the work in (Koller, Megiddo, & von Stengel 1994). In this work, the authors take advantage of the *sequence form* representation of two-player games to find mixed strategy Nash equilibria efficiently. In contrast to their work, ours applies to any number of players. Furthermore, our algorithms are focused on eliminating dominated strategies, and do not make any assumptions about which of the remaining strategies will be played.

For the special case of cooperative games, several algorithms have been proposed. However, none of the previous algorithms guarantee optimality in general. In (Hsu & Marcus 1982), a dynamic programming algorithm is presented for the case in which the agents observe each others' private information after a one-step delay. Becker et al. (2003) present an algorithm that works for problems in which the agents' dynamics are loosely coupled. Finally, the algorithms of Peshkin et al. (2000) and Nair et al. (2003) can be applied to the general problem, but are only guaranteed to converge to local optima.

Background

We first review the POSG model and two algorithms that we generalize to create a dynamic programming algorithm for POSGs: dynamic programming for POMDPs and elimination of dominated strategies in solving normal form games.

Partially observable stochastic games

A *partially observable stochastic game* (POSG) is a tuple $\langle \mathcal{I}, \mathcal{S}, \{b0\}, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \mathcal{P}, \{R_i\} \rangle$, where

- $\mathcal{I}$ is a finite set of agents (or controllers) indexed $1, \ldots, n$
- $\mathcal{S}$ is a finite set of states
- $b0 \in \Delta(\mathcal{S})$ represents the initial state distribution.
- $\mathcal{A}_i$ is a finite set of actions available to agent i and $\vec{\mathcal{A}} = \times_{i \in \mathcal{I}} \mathcal{A}_i$ is the set of joint actions (i.e., action profiles), where $\vec{a} = \langle a_1, \ldots, a_n \rangle$ denotes a joint action
- $\mathcal{O}_i$ is a finite set of observations for agent i and $\vec{\mathcal{O}} = \times_{i \in \mathcal{I}} \mathcal{O}_i$ is the set of joint observations, where $\vec{o} = \langle o_1, \ldots, o_n \rangle$ denotes a joint observation
- $\mathcal{P}$ is a set of Markovian state transition and observation probabilities, where $\mathcal{P}(s', \vec{o} | s, \vec{a})$ denotes the probability that taking joint action $\vec{a}$ in state s results in a transition to state s' and joint observation $\vec{o}$
- $R_i : \mathcal{S} \times \vec{\mathcal{A}} \rightarrow \Re$ is a reward function for agent i

A game unfolds over a finite or infinite sequence of stages, where the number of stages is called the *horizon* of the game. In this paper, we consider only finite-horizon POSGs; the challenges involved in solving the infinite-horizon case are discussed further on in the paper. At each stage, all agents simultaneously select an action and receive a reward and observation. The objective, for each agent, is to maximize the sum of rewards it receives during the game.

Whether agents compete or cooperate in seeking reward depends on their reward functions. The case in which the agents share the same reward function is called a *decentralized partially observable Markov decision process (DEC-POMDP)* (Bernstein *et al.* 2002).

Dynamic programming for POMDPs

A POSG with a single agent corresponds to a POMDP. We briefly review an exact dynamic programming algorithm for POMDPs that provides a foundation for our exact dynamic programming algorithm for POSGs. We use the same notation for POMDPs as for POSGs, but omit the subscript that indexes an agent.

The first step in solving a POMDP by dynamic programming (DP) is to convert it into a completely observable MDP with a state set $\mathcal{B} = \Delta(\mathcal{S})$ that consists of all possible beliefs about the current state. Let $b^{a,o}$ denote the belief state that results from belief state b, after action a and observation o. The DP operator can be written in the form,

$$V^{t+1}(b) = \max_{a \in \mathcal{A}} \left\{ \sum_s R(s,a) + \sum_{s',o} \mathcal{P}(s', o | s, a) V^t(b^{a,o}) \right\}, \quad (1)$$

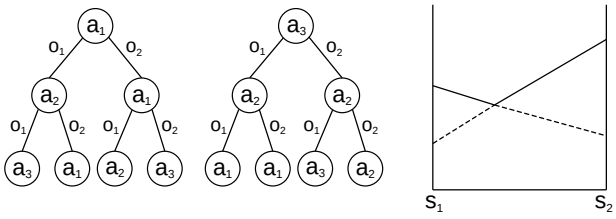

Figure 1: Two policy trees, along with their value function for a POMDP with two states.

where the updated value function is computed for all belief states $b \in \mathcal{B}$. Exact DP algorithms for POMDPs rely on Smallwood and Sondik's (1973) proof that the DP operator preserves the piecewise linearity and convexity of the value function. This means that the value function can be represented exactly by a finite set of $|\mathcal{S}|$-dimensional value vectors, denoted $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$, where

$$V(b) = \max_{1 \leq j \leq k} \sum_{s \in \mathcal{S}} b(s) v_j(s). \quad (2)$$

As elucidated by Kaelbling et al. (1998), each value vector corresponds to a complete conditional plan that specifies an action for every sequence of observations. Adopting the terminology of game theory, we sometimes refer to a complete conditional plan as a *strategy*. We use this interchangeably with *policy tree*, because a conditional plan can be viewed as a tree. A set of policy trees along with their value function over belief-state space are shown in Figure 1.

The DP operator of Equation (1) computes an updated value function, but can also be interpreted as computing an updated set of policy trees. In fact, the simplest algorithm for computing the DP update has two steps, which are described below.

In the first step, the DP operator is given a set Q^t of depth-t policy trees and a corresponding set $\mathcal{V}^t$ of value vectors representing the horizon-t value function. It computes Q^{t+1} and $\mathcal{V}^{t+1}$ in two steps. First, a set of depth $t+1$ policy trees, Q^{t+1}, is created by generating every possible depth $t+1$ policy tree that makes a transition, after an action and observation, to the root node of some depth-t policy tree in Q^t. This operation will hereafter be called an *exhaustive backup*. Note that $|Q^{t+1}| = |\mathcal{A}||Q^t|^{|\mathcal{O}|}$. For each policy tree $q_j \in Q^{t+1}$, it is straightforward to compute a corresponding value vector, $v_j \in \mathcal{V}^{t+1}$.

The second step is to eliminate policy trees that need not be followed by a decision maker that is maximizing expected value. This is accomplished by eliminating (i.e., pruning) any policy tree when this can be done without decreasing the value of any belief state.

Formally, a policy tree q_j is considered dominated and can be removed from the current set Q_i^{t+1} if for all $b \in \mathcal{B}$ there exists a $v_k \in \mathcal{V}_i^{t+1} \setminus v_j$ such that $b \cdot v_k \geq b \cdot v_j$. When a policy tree is removed, its corresponding value vector is also removed. The test for dominance can be performed using linear programming. The dual linear program corresponding to this test is to prune any policy tree q_j for which there is

a probability distribution p over the other policy trees, such that

$$\sum_{k \neq j} p(k) v_k(s) \geq v_j(s), \forall s \in S. \quad (3)$$

This dual LP plays a role in iterative strategy elimination, as we will see in the next section, and was recently applied in the context of POMDPs (Poupart & Boutilier 2004).

Iterative elimination of dominated strategies

Techniques for eliminating dominated strategies in solving a POMDP are very closely related to techniques for eliminating dominated strategies in solving games in normal form. A game in normal form is a tuple $G = \{\mathcal{I}, \{D_i\}, \{V_i\}\}$, where $\mathcal{I}$ is a finite set of agents, D_i is a finite set of strategies available to agent i, and $V_i : \vec{D} \rightarrow \Re$ is the value (or payoff) function for agent i. Unlike a stochastic game, there are no states or state transitions in this model.

Every strategy d_i in D_i is a *pure strategy*. Let $\delta_i \in \Delta(D_i)$ denote a *mixed strategy*, that is, a probability distribution over the pure strategies available to agent i, where $\delta_i(d_i)$ denotes the probability assigned to strategy $d_i \in D_i$. Let d_{-i} denote a profile of pure strategies for the other agents (i.e., all the agents except agent i), and let δ_{-i} denote a profile of mixed strategies for the other agents. Since agents select strategies simultaneously, δ_{-i} can also represent agent i's belief about the other agents' likely strategies. If we define $V_i(d_i, \delta_{-i}) = \sum_{d_{-i}} \delta_{-i}(d_{-i}) V(d_i, d_{-i})$, then

$$B_i(\delta_{-i}) = \{d_i \in D_i | V_i(d_i, \delta_{-i}) \geq V_i(d'_i, \delta_{-i}) \; \forall d'_i \in D_i\} \quad (4)$$

denotes the *best response function* of agent i, which is the set of strategies for agent i that maximize the value of some belief about the strategies of the other agents. Any strategy that is not a best response to some belief can be deleted.

A dominated strategy d_i is identified by using linear programming. The linear program identifies a probability distribution σ_i over the other strategies such that

$$V_i(\sigma_i, d_{-i}) > V_i(d_i, d_{-i}), \forall d_{-i} \in D_{-i}. \quad (5)$$

This test for dominance is very similar to the test for dominance used to prune strategies in solving a POMDP. It differs in using strict inequality, which is called *strict dominance*. Game theorists also use *weak dominance* to prune strategies. A strategy d_i is weakly dominated if $V_i(\sigma_i, d_{-i}) \geq V_i(d_i, d_{-i})$ for all $d_{-i} \in D_{-i}$, and $V_i(\sigma_i, d_{-i}) > V_i(d_i, d_{-i})$ for some $d_{-i} \in D_{-i}$. The test for dominance which does not require any strict inequality is sometimes called *very weak dominance*, and corresponds exactly to the test for dominance in POMDPs.

There are a couple other interesting differences between the tests for dominance in Equations (3) and (5). First, there is a difference in beliefs. In normal-form games, beliefs are about the strategies of other agents, whereas in POMDPs, beliefs are about the underlying state. Second, elimination of dominated strategies is iterative when there are multiple agents. When one agent eliminates its dominated strategies, this can affect the best-response function of other agents (assuming common knowledge of rationality). After all agents take a turn in eliminating their dominated strategies, they can consider eliminating additional strategies that may only have been best responses to strategies of other agents that have since been eliminated. The procedure of alternating between agents until no agent can eliminate another strategy is called *iterative elimination of dominated strategies*.

In solving normal-form games, iterative elimination of dominated strategies is a somewhat weak solution concept, in that it does not (usually) identify a specific strategy for an agent to play, but rather a set of possible strategies. To select a specific strategy requires additional reasoning, and introduces the concept of a Nash equilibrium, which is a profile of strategies (possibly mixed), such that $\delta_i \in B_i(\delta_{-i})$ for all agents i. Since there are often multiple equilibria, the problem of *equilibrium selection* is an important one. (It has a more straightforward solution for cooperative games than for general-sum games.) In this paper, we focus on the issue of elimination of dominated strategies.

Dynamic programming for POSGs

We now describe a DP algorithm for POSGs that is a synthesis of DP for POMDPs and iterative elimination of very weakly dominated strategies in normal-form games. To motivate the algorithm, we first point out that every horizon-t POSG can be given a normal form representation. The strategy sets include all depth-t policy trees, and the value of a strategy profile is the t-step reward achieved from the start state distribution. If a horizon-t POSG is represented this way, iterative elimination of dominated strategies could be used in solving the game.

The problem is that this representation can be much larger than the original representation of a POSG. In fact, the size of the strategy set for each agent i is greater than $|\mathcal{A}_i|^{|\mathcal{O}_i|^t}$, which is doubly exponential in t. Because of the large sizes of the strategy sets, it is usually not feasible to work directly with the normal form representation of a POSG. The dynamic programming algorithm we present addresses this problem by performing iterative elimination of dominated strategies without first constructing the normal form representation.

In the dynamic programming approach, a POSG is solved in stages. The key step of our algorithm is a *multi-agent dynamic programming operator* that generalizes the DP operator for POMDPs, and is defined as follows. First, the DP operator is given a set of depth-t policy trees Q_i^t, and a corresponding set of value vectors $\mathcal{V}_i^t$ for each agent i. The operator performs an exhaustive backup on each of the sets of trees, to form Q_i^{t+1} for each agent i. It then recursively computes the value vectors in $\mathcal{V}_i^{t+1}$ for each agent i.

The second step of the multi-agent DP operator consists of pruning dominated strategies. As in the single agent case, an agent i policy tree can be pruned if its removal does not decrease the value of any belief state for agent i. As with normal form games, removal of a policy tree reduces the dimensionality of the other agents' belief state spaces, and it can be repeated until no more policy trees can be pruned from any agent's set. (Note that different agent orderings may lead to different sets of policy trees and value vectors.

Input: Sets of depth-t policy trees Q_i^t and corresponding value vectors $\mathcal{V}_i^t$ for each agent i.

1. Perform exhaustive backups to get Q_i^{t+1} for each i.
2. Recursively compute $\mathcal{V}_i^{t+1}$ for each i.
3. Repeat until no more pruning is possible:
   (a) Choose an agent i, and find a policy tree $q_j \in Q_i^{t+1}$ for which the following condition is satisfied: $\forall b \in \Delta(\mathcal{S} \times Q_{-i}^{t+1}), \exists v_k \in \mathcal{V}_i^{t+1} \setminus v_j$ s.t. $b \cdot v_k \geq b \cdot v_j$.
   (b) $Q_i^{t+1} \leftarrow Q_i^{t+1} \setminus q_j$.
   (c) $\mathcal{V}_i^{t+1} \leftarrow \mathcal{V}_i^{t+1} \setminus v_j$.

Output: Sets of depth-$t+1$ policy trees Q_i^{t+1} and corresponding value vectors $\mathcal{V}_i^{t+1}$ for each agent i.

Table 1: The multi-agent dynamic programming operator.

The question of order dependence in eliminating dominated strategies has been extensively studied in game theory, and we do not consider it here.) Pseudocode for the multi-agent DP operator is given in Table 1.

The algorithm depends crucially on how a belief state is defined. Our definition of a belief state is a simple synthesis of the definition of a belief state for POMDPs (a distribution over possible states) and the definition of a belief state in iterative elimination of dominated strategies (a distribution over the possible strategies of the other agents). For each agent i, a belief state is defined to be a distribution over $\mathcal{S} \times Q_{-i}^t$. From this perspective, each agent treats the other agents' current policy trees as part of its state space. A value vector in $\mathcal{V}_i^t$ assigns value to each element of $\mathcal{S} \times Q_{-i}^t$, and the set of value vectors is a piecewise linear and convex representation of the value function for the agent's belief space.

There is an important difference between this algorithm and the single agent algorithm, in terms of implementation. In the single agent case, only the value vectors need to be kept in memory. At execution time, an optimal policy can be extracted using one-step lookahead. We do not currently have a way of doing this when there are multiple agents. Thus, the policy tree sets must also be remembered. Of course, some memory savings is possible by realizing that the policy trees for an agent share subtrees.

The dynamic programming algorithm can be viewed as performing strategy elimination in the POSG without first forming the normal-form representation. On iteration $k \leq t$, when a depth-k policy tree is pruned by the multi-agent DP operator, every depth-t policy tree containing it as a subtree is effectively eliminated. In fact we can prove that the algorithm performs iterative elimination of *very weakly dominated* strategies.

Theorem 1 *Dynamic programming applied to a finite-horizon POSG corresponds to iterative elimination of very weakly dominated strategies in the normal form of the POSG.*

Proof: Let t be the horizon of the POSG, and suppose that iteration $k \leq t$ of dynamic programming has thus far produced a policy tree set Q_i^k for each agent i. Equivalently, it has pruned strategies in the normal form of the game down to all those whose only depth-k subtrees are in Q_i^k. Suppose further that the next policy tree to be pruned is $q_j \in Q_i^k$. According to the dual formulation of the pruning rule, there exists a distribution p over policy trees in $Q_i^k \setminus q_j$ such that $\sum_{k \neq j} p(k) v_k(s, q_{-i}) \geq v_j(s, q_{-i})$ for all $s \in \mathcal{S}$ and $q_{-i} \in Q_{-i}^k$.

Consider any depth-t policy tree q' with q_j as a subtree. We can replace instances of q_j in q' with the distribution p to get a *behavioral strategy*, which is a stochastic policy tree. From the pruning rule given above, it follows that the value of this behavioral strategy is at least as high as that of q', regardless of the initial state distribution and strategies executed by the other agents. Since a POSG is a game with perfect recall, every behavioral strategy can be represented by a distribution over pure strategies (Kuhn 1953). Thus, in the normal form representation of the POSG, q' is very weakly dominated. □

In the case of cooperative games, removing very weakly dominated strategies preserves at least one optimal strategy profile. Thus, the multi-agent DP operator can be used to solve finite-horizon DEC-POMDPs optimally. When dynamic programming reaches step t, we can simply extract the highest-valued strategy profile from the start state distribution.

Corollary 1 *Dynamic programming applied to a finite-horizon DEC-POMDP yields an optimal strategy profile.*

In general-sum games, it may be undesirable to remove certain very weakly dominated strategies. It is still an open question whether we can define DP operators that prune only weakly dominated or strongly dominated strategies. In the strongly dominated case, it may seem reasonable simply to make the inequality in the pruning rule strict. However, sometimes a strategy that is not strongly dominated will have a strongly dominated, but unreachable subtree. This is referred to as an *incredible threat* in the game theory literature.

Example

We ran initial tests on a cooperative game involving control of a multi-access broadcast channel (Ooi & Wornell 1996). In this problem, nodes need to broadcast messages to each other over a channel, but only one node may broadcast at a time, otherwise a collision occurs. The nodes all share the common goal of maximizing the throughput of the channel.

The process proceeds in discrete time steps. At the start of each time step, each node decides whether or not to send a message. The nodes receive a reward of 1 when a message is successfully broadcast and a reward of 0 otherwise. At the end of the time step, each node receives a noisy observation of whether or not a message got through.

The message buffer for each agent has space for only one message. If a node is unable to broadcast a message, the message remains in the buffer for the next time step. If a node i is able to send its message, the probability that its buffer will fill up on the next step is p_i. Our problem has two nodes, with $p_1 = 0.9$ and $p_2 = 0.1$. There are 5 states, 2 actions per agent, and 2 observations per agent.

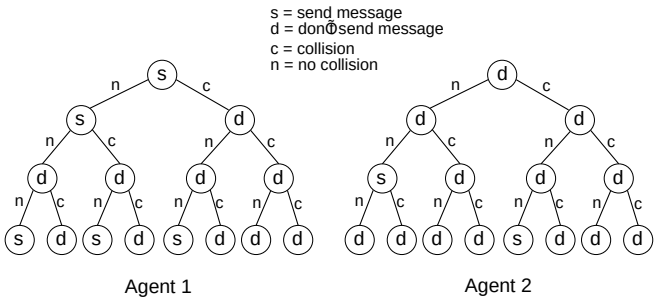

Figure 2: A pair of policy trees that is optimal when both message buffers start out full. The policy trees achieve a total reward of 3.89.

We compared our dynamic programming algorithm with a brute force algorithm, which also builds sets of policy trees, but never prunes any of them. On a machine with 2 gigabytes of memory, the brute force algorithm was able to complete iteration 3 before running out of memory, while the dynamic programming algorithm was able to complete iteration 4. At the end of iteration 4, the number of policies for the dynamic programming algorithm was only about 1% of the number that would have been produced by the brute force algorithm, had it been able to complete the iteration. This shows that the multi-agent DP operator can prune a significant number of trees. However, even with pruning, the number of policy trees grows quickly with the horizon. This issue is discussed in the next section.

Figure 2 shows a pair of depth-4 trees constructed by the dynamic programming algorithm. In the case where the message buffers both start out full, this pair is optimal, yielding a total reward of 3.89.

Future work

Development of an exact dynamic programming approach to solving POSGs suggests several avenues for future research, and we briefly describe some possibilities.

Improving efficiency

A major scalability bottleneck is the fact that the number of policy trees grows rapidly with the horizon and can quickly consume a large amount of memory. We may be able to avoid this if we are willing to prune strategies that are *almost* very weakly dominated. Aggressive pruning has already been tried with POMDPs, with some success (Feng & Hansen 2000). Furthermore, we may be able to reduce the number of policy trees if we allow for stochastic policies, as in (Poupart & Boutilier 2004). We may also be able to extend work on compactly represented POMDPs and value functions to the multi-agent case (Boutilier & Poole 1996; Hansen & Feng 2000).

There may also be ways to reduce the amount of time that the multi-agent DP backup takes. There has been a lot of work in the POMDP literature on pruning policy trees incrementally, so that an exhaustive backup never has to be done (Cassandra, Littman, & Zhang 1997). Whether this can be extended to the multi-agent case remains an open problem.

Finally, there exist POMDP algorithms that leverage a known start state distribution for greater efficiency. These algorithms perform a forward search from the start state and are able to avoid unreachable belief states. Whether some kind of forward search can be done in the multi-agent case is an important open problem.

Eliminating Pareto dominated strategies

The pruning technique that we described deals with only one agent at a time. By considering multiple agents at the same time, we may be able to eliminate more strategies. In particular, we can eliminate strategies that are part of a *Pareto-dominated* strategy profile. A strategy profile is Pareto dominated when there is another strategy profile that yields at least as much reward for all agents. In some cases—most notably the fully cooperative case—it is sensible to eliminate such strategy profiles.

We would like to define a multi-agent DP operator that eliminates only Pareto dominated strategy profiles. The following is an intuitive explanation of how this might be done. Consider iteration t of dynamic programming for a two-agent POSG. First, an exhaustive backup is performed, and the resulting policy trees are evaluated. Next, the algorithm identifies two pairs of policy trees, (q_1, q_2) and (q_1', q_2'), satisfying the following two conditions. First, for all states, q_1' yields at least as much reward as q_1 over t steps when paired with any agent 2 tree, with the possible exception of q_2. Second, for all states, q_2' yields at least as much reward as q_2 over t steps when paired with any agent 1 tree, with the possible exception q_1. At this point, q_1 and q_2 can be eliminated because replacing them with q_1' and q_2' does not lead to a decrease in value for either agent in any situation.

It should be possible to generalize the reasoning above to more than two agents and more than two policy trees at a time. The first step in this line of work will be to derive a general condition for pruning policy trees, and the second step will be to develop an efficient algorithm for checking whether the condition is satisfied.

Extension to infinite-horizon POSGs

It should be possible to extend our dynamic programming algorithm to apply to infinite-horizon, discounted POSGs, and we are currently exploring this possibility. In this case, the multi-agent DP operator is applied to infinite trees. One possibility for representing a set of infinite trees is as a finite-state controller (FSC). In the case of single-agent POMDPs, some interesting algorithms have been based on this representation (Hansen 1998; Poupart & Boutilier 2004), and we believe that these can be extended.

Many questions remain regarding the infinite-horizon case, however. It seems more difficult to describe what an infinite-horizon version of the algorithm would be doing in game-theoretic terms. In a sense, it would still be eliminating dominated strategies, but unlike in the finite-horizon case, this process would be heavily biased by the choice of initial FSCs.

Restricting attention to the cooperative case clears up the picture quite a bit. In that case, we would want to be able to claim that an algorithm returns a set of controllers that is

ϵ-optimal in a finite number of iterations. This gives rise to interesting questions regarding the existence of some type of Bellman equation and the difficulty of computing error bounds.

Conclusion

We presented an algorithm for solving POSGs that generalizes both dynamic programming for POMDPs and iterative elimination of very weakly dominated strategies for normal form games. The key component of the algorithm is a generalization of belief states to include uncertainty about the other agents' future behavior. This makes it possible to define a multi-agent dynamic programming operator for POSGs that performs stage-wise elimination of dominated strategies. There are many avenues for future research, in both making the algorithm more time and space efficient and extending it beyond finite-horizon POSGs.

Acknowledgements

This work was supported in part by the National Science Foundation under grants IIS-0219606 and IIS-9984952, and by NASA under cooperative agreement NCC-2-1311 and grant NAG-2-1463. Daniel Bernstein was supported by a NASA GSRP Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the NSF or NASA.

References

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2003. Transition-independent decentralized Markov decision processes. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems*.

Bernstein, D.; 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., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1168–1175.

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

Brafman, R., and Tennenholtz, M. 2002. R-MAX–a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3:213–231.

Cassandra, A.; Littman, M. L.; and Zhang, N. L. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, 54–61.

Feng, Z., and Hansen, E. 2000. Approximate planning for factored POMDPs. In *Proceedings of the Sixth European Conference on Planning*.

Filar, J., and Vrieze, K. 1997. *Competitive Markov Decision Processes*. Springer-Verlag.

Hansen, E., and Feng, Z. 2000. Dynamic programming for POMDPs using a factored state representation. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, 130–139.

Hansen, E. 1998. Solving POMDPs by searching in policy space. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 211–219.

Hsu, K., and Marcus, S. I. 1982. Decentralized control of finite state Markov processes. *IEEE Transactions on Automatic Control* AC-27(2):426–431.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Koller, D.; Megiddo, N.; and von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, 750–759.

Kuhn, H. 1953. *Contributions to the Theory of Games II*. Princeton University Press. chapter Extensive Games and the Problem of Information, 193–216.

Littman, M. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, 157–163.

Nair, R.; Pynadath, D.; Yokoo, M.; Tambe, M.; 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*.

Ooi, J. M., and Wornell, G. W. 1996. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proceedings of the 35th Conference on Decision and Control*, 293–298.

Peshkin, L.; Kim, K.-E.; Meuleau, N.; and Kaelbling, L. P. 2000. Learning to cooperate via policy search. In *Proceedings of the Sixteenth International Conference on Uncertainty in Artificial Intelligence*, 489–496.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Policy-contingent abstraction for robust robot control. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*.

Poupart, P., and Boutilier, C. 2004. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*. Vancouver, Canada: MIT Press.

Shapley, L. 1953. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America* 39:1095–1100.

Smallwood, R., and Sondik, E. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21:1071–1088.