

Bounded Policy Iteration for Decentralized POMDPs

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

Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
shlomo@cs.umass.edu

Abstract

We present a bounded policy iteration algorithm for infinite-horizon decentralized POMDPs. Policies are represented as joint stochastic finite-state controllers, which consist of a local controller for each agent. We also let a joint controller include a correlation device that allows the agents to correlate their behavior without exchanging information during execution, and show that this leads to improved performance. The algorithm uses a fixed amount of memory, and each iteration is guaranteed to produce a controller with value at least as high as the previous one for all possible initial state distributions. For the case of a single agent, the algorithm reduces to Poupart and Boutilier’s bounded policy iteration for POMDPs.

1 Introduction

The Markov decision process (MDP) framework has proven to be useful for solving problems of sequential decision making under uncertainty. For some problems, an agent must base its decision on partial information about the system state. In this case, it is often better to use the more general partially observable Markov decision process (POMDP) framework. Though POMDPs are difficult to solve in the worst case, much progress has been made in the development of practical dynamic programming algorithms [Smallwood and Sondik, 1973; Cassandra *et al.*, 1997; Hansen, 1998; Poupart and Boutilier, 2003; Feng and Zilberstein, 2004].

Even more general are problems in which a team of decision makers, each with its own local observations, must act together. Domains in which these types of problems arise include networking, multi-robot coordination, e-commerce, and space exploration systems. To model such problems, we can use the decentralized partially observable Markov decision process (DEC-POMDP) framework. Though this model has been recognized for decades (see, e.g., [Witsenhausen, 1971]), there has been little work on efficient algorithms for it.

Recently, an exact dynamic programming algorithm was proposed for DEC-POMDPs [Hansen *et al.*, 2004]. Though the algorithm was presented in the context of finite-horizon

problems, there are various ways to extend it to the infinite-horizon case. However, in both cases, it suffers from the fact that the memory requirements grow quickly with each iteration, and in practice it has only been used to solve very small problems. It is likely that any optimal algorithm would suffer this problem, as finite-horizon DEC-POMDPs have been shown to be NEXP-complete, even for just two agents [Bernstein *et al.*, 2002].

In this paper, we present a memory-bounded dynamic programming algorithm for infinite-horizon DEC-POMDPs. The algorithm uses a stochastic finite-state controller to represent the joint policy for the agents. A straightforward approach is to use a set of independent local controllers, one for each agent. We provide an example to illustrate that higher value can be obtained through the use of shared randomness. As such, we define a joint controller to be a set of local controllers along with a *correlation device*. The correlation device is a finite-state machine that sends a signal to all of the agents on each time step. Its behavior can be determined prior to execution time, and thus it does not require that the agents exchange information after receiving local observations.

Our algorithm generalizes *bounded policy iteration* for POMDPs [Poupart and Boutilier, 2003] to the multi-agent case. On each iteration, a node is chosen from one of the local controllers or the correlation device, and its parameters are updated through the solution of a linear program. The generalization has the same theoretical guarantees as in the POMDP case. Namely, an iteration is guaranteed to produce a new controller with value at least as high for every possible initial state distribution.

In our experiments, we applied our algorithm to idealized networking and robot navigation problems. Both problems are too large for exact dynamic programming, but could be handled by our approximation algorithm. We found that the addition of a correlation device gives rise to better solutions. In addition, larger controllers most often lead to better solutions.

A number of approximation algorithms have been developed previously for DEC-POMDPs [Peshkin *et al.*, 2000; Nair *et al.*, 2003; Emery-Montemerlo *et al.*, 2004]. However, the previous algorithms do not guarantee both bounded memory usage and monotonic value improvement for all initial state distributions. Furthermore, the use of correlated stochastic policies in the DEC-POMDP context is novel. The

importance of correlation has been recognized in the game theory community [Aumann, 1974], but there has been little work on algorithms for finding correlated policies.

2 Background

In this section, we present our formal framework for multi-agent decision making. A *decentralized partially-observable Markov decision process (DEC-POMDP)* is a tuple $\langle I, S, \{A_i\}, \{O_i\}, P, R \rangle$, where

- I is a finite set of agents indexed $1, \dots, n$
- S is a finite set of states
- A_i is a finite set of actions available to agent i and $\vec{A} = \times_{i \in I} A_i$ is the set of joint actions, where $\vec{a} = \langle a_1, \dots, a_n \rangle$ denotes a joint action
- O_i is a finite set of observations for agent i and $\vec{O} = \times_{i \in I} O_i$ is the set of joint observations, where $\vec{o} = \langle o_1, \dots, o_n \rangle$ denotes a joint observation
- P is a set of Markovian state transition and observation probabilities, where $P(s', \vec{o} | s, \vec{a})$ denotes the probability that taking joint action \vec{a} in state s results in a transition to state s' and joint observation \vec{o}
- $R : S \times \vec{A} \rightarrow \mathbb{R}$ is a reward function

In this paper, we consider the case in which the process unfolds over an infinite sequence of stages. At each stage, all agents simultaneously select an action, and each receives the global reward and a local observation. The objective of the agents is to maximize the expected discounted sum of rewards received. We denote the discount factor γ and require that $0 \leq \gamma < 1$.

3 Finite-State Controllers

Our algorithm uses stochastic finite-state controllers to represent policies. In this section, we first define a type of controller in which the agents act independently. We then provide an example demonstrating the utility of correlation, and show how to extend the definition of a joint controller to allow for correlation among agents.

3.1 Local Finite-State Controllers

In a DEC-POMDP, each agent must select an action based on its history of local observations. Finite-state controllers provide a way to represent local policies using a finite amount of memory. The state of the controller is based on the observation sequence, and the agent's actions are based on the state of its controller. We allow for stochastic transitions and stochastic action selection, as this can help to make up for limited memory. This type of controller has been used previously in the single-agent context [Platzman, 1980; Meuleau *et al.*, 1999; Poupart and Boutilier, 2003].

Formally, we define a *local finite-state controller* for agent i to be a tuple $\langle Q_i, \psi_i, \eta_i \rangle$, where Q_i is a finite set of controller nodes, $\psi_i : Q_i \rightarrow \Delta A_i$ is an action selection function, and $\eta_i : Q_i \times A_i \times O_i \rightarrow \Delta Q_i$ is a transition function. The functions ψ_i and η_i parameterize the conditional distribution $P(a_i, q'_i | q_i, o_i)$.

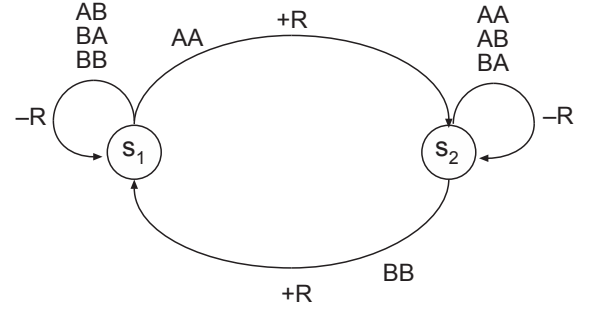


Figure 1: This figure shows a DEC-POMDP for which the optimal memoryless joint policy requires correlation.

Taken together, the agents' controllers determine the conditional distribution $P(\vec{a}, \vec{q}' | \vec{q}, \vec{o})$. This is denoted an *independent joint controller*. In the following subsection, we show that independence can be limiting.

3.2 The Utility of Correlation

The joint controllers described above do not allow the agents to correlate their behavior via a shared source of randomness. We will use a simple example to illustrate the utility of correlation in partially observable domains where agents have limited memory. This example generalizes the one given in [Singh *et al.*, 1994] to illustrate the utility of stochastic policies in single-agent partially observable settings.

Consider the DEC-POMDP shown in Figure 1. This problem has two states, two agents, and two actions per agent (A and B). The agents each have only one observation, and thus cannot distinguish between the two states. For this example, we will consider only memoryless policies.

Suppose that the agents can independently randomize their behavior using distributions $P(a_1)$ and $P(a_2)$, and consider the policy in which each agent chooses either A or B according to a uniform distribution. This yields an expected reward of $-\frac{R}{2}$ per time step, which results in an expected long-term reward of $\frac{-R}{2(1-\gamma)}$. It is straightforward to show that no independent policy yields higher reward than this one for all states.

Next, let us consider the larger class of policies in which the agents may act in a correlated fashion. In other words, we consider all joint distributions $P(a_1, a_2)$. Consider the policy that assigns probability $\frac{1}{2}$ to the pair AA and probability $\frac{1}{2}$ to the pair BB . This yields an average reward of 0 at each time step and thus an expected long-term reward of 0. The difference between the rewards obtained by the independent and correlated policies can be made arbitrarily large by increasing R .

3.3 Correlated Joint Controllers

In the previous subsection, we established that correlation can be useful in the face of limited memory. In this subsection, we extend our definition of a joint controller to allow for correlation among the agents. To do this, we introduce an additional finite-state machine, called a correlation device, that provides

Variables: $\epsilon, x(c, a_i), x(c, a_i, o_i, q'_i)$
 Objective: Maximize ϵ
 Improvement constraints:

$$\forall s, q_{-i}, c \quad V(s, \vec{q}, c) + \epsilon \leq \sum_{\vec{a}} P(a_{-i}|c, q_{-i}) [x(c, a_i) R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} x(c, a_i, o_i, q'_i) P(q'_{-i}|c, q_{-i}, a_{-i}, o_{-i}) P(s', \vec{o}|s, \vec{a}) P(c'|c) V(s', \vec{q}', c')]$$

Probability constraints:

$$\forall c \quad \sum_{a_i} x(c, a_i) = 1, \quad \forall c, a_i, o_i \quad \sum_{q'_i} x(c, a_i, o_i, q'_i) = x(c, a_i) \\ \forall c, a_i \quad x(c, a_i) \geq 0, \quad \forall c, a_i, o_i, q'_i \quad x(c, a_i, o_i, q'_i) \geq 0$$

Table 1: The linear program used to find new parameters for agent i 's node q_i . The variable $x(c, a_i)$ represents $P(a_i|q_i, c)$, and the variable $x(c, a_i, o_i, q'_i)$ represents $P(a_i, q'_i|c, q_i, o_i)$.

extra signals to the agents at each time step. The device operates independently of the DEC-POMDP process, and thus does not provide the agents with information about the other agents' observations. In fact, the random numbers necessary for its operation could be determined prior to execution time.

Formally, a *correlation device* is a tuple $\langle C, \psi \rangle$, where C is a set of states and $\psi : C \rightarrow \Delta C$ is a state transition function. At each step, the device undergoes a transition, and each agent observes its state.

We must modify the definition of a local controller to take the state of the correlation device as input. Now, a local controller for agent i is a conditional distribution of the form $P(a_i, q'_i|c, q_i, o_i)$. The correlation device together with the local controllers form a joint conditional distribution $P(c', \vec{a}, \vec{q}'|c, \vec{q}, \vec{o})$. We will refer to this as a *correlated joint controller*. Note that a correlated joint controller with $|C| = 1$ is effectively an independent joint controller.

The value function for a correlated joint controller can be computed by solving the following system of linear equations, one for each $s \in S, \vec{q} \in \vec{Q}$, and $c \in C$:

$$V(s, \vec{q}, c) = \sum_{\vec{a}} P(\vec{a}|c, \vec{q}) [R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(s', \vec{o}|s, \vec{a}) P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) \cdot P(c'|c) V(s', \vec{q}', c')].$$

We sometimes refer to the value of the controller for an initial state distribution. For a distribution δ , this is defined as

$$V(\delta) = \max_{\vec{q}, c} \sum_s \delta(s) V(s, \vec{q}, c).$$

It is assumed that, given an initial state distribution, the controller is started in the joint node which maximizes value from that distribution.

4 Bounded Policy Iteration

We now describe our bounded policy iteration algorithm for improving correlated joint controllers. To improve a corre-

lated joint controller, we can either change the correlation device or one of the local controllers. Both improvements can be done via a *bounded backup*, which involves solving a linear program. Following an improvement, the controller can be reevaluated through the solution of a set of linear equations. Below, we describe how a bounded backup works, and prove that it always produces a new controller with value at least as high for all initial state distributions.

4.1 Improving a Local Controller

We first describe how to improve a local controller. To do this, we choose an agent i , along with a node q_i . Then, we search for new parameters for the conditional distribution $P(a_i, q'_i|c, q_i, o_i)$.

The search for new parameters works as follows. We assume that the original controller will be used from the second step on, and try to replace the parameters for q_i with better ones for just the first step. In other words, we look for the best parameters satisfying the following inequality:

$$V(s, \vec{q}, c) \leq \sum_{\vec{a}} P(\vec{a}|c, \vec{q}) [R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) \cdot P(c'|c) V(s', \vec{q}', c)]$$

for all $s \in S, q_{-i} \in Q_{-i}$, and $c \in C$. Note that the inequality is always satisfied by the original parameters. However, it is often possible to get an improvement.

Finding new parameters can be done using linear programming, as shown in Table 1. We note that this linear program is the same as that of Poupart and Boutilier [2003] for POMDPs, with the nodes of the other local controllers and correlation device considered part of the hidden state. Its size is polynomial in the sizes of the DEC-POMDP and the joint controller, but exponential in the number of agents.

4.2 Improving the Correlation Device

The procedure for improving the correlation device is very similar to the procedure for improving a local controller. We

Variables: $\epsilon, x(c')$
 Objective: Maximize ϵ
 Improvement constraints:

$$\forall s, \vec{q} \quad V(s, \vec{q}, c) + \epsilon \leq \sum_{\vec{a}} P(\vec{a}|c, \vec{q}) [R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) x(c') V(s', \vec{q}', c')]$$

Probability constraints:

$$\forall c' \quad \sum_{c'} x(c') = 1, \quad \forall c' \quad x(c') \geq 0$$

Table 2: The linear program used to find new parameters for the correlation device node c . The variable $x(c')$ represents $P(c'|c)$.

first choose a device node c , and consider changing its parameters for just the first step. We look for the best parameters satisfying the following inequality:

$$\begin{aligned} V(s, \vec{q}, c) \leq & \sum_{\vec{a}} P(\vec{a}|c, \vec{q}) [R(s, \vec{a}) + \\ & \gamma \sum_{s', \vec{o}, \vec{q}', c} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) \\ & \cdot P(c'|c) V(s', \vec{q}', c')] \end{aligned}$$

for all $s \in S$ and $\vec{q} \in \vec{Q}$.

As in the previous case, the search for parameters can be formulated as a linear program. This is shown in Table 2. This linear program is also polynomial in the sizes of the DEC-POMDP and joint controller, but exponential in the number of agents.

4.3 Monotonic Improvement

We have the following theorem, which says that performing either of the two updates cannot lead to a decrease in value for any initial state distribution.

Theorem 1 *Performing a bounded backup on a local controller or the correlation device produces a correlated joint controller with value at least as high for every initial state distribution.*

Proof. Consider the case in which some node q_i of agent i 's local controller is changed. Let V_o be the value function for the original controller, and let V_n be the value function for the new controller. Recall that the new parameters for $P(a_i, q'_i|c, q_i, o_i)$ must satisfy the following inequality for all $s \in S, q_{-i} \in Q_{-i}$, and $c \in C$:

$$\begin{aligned} V_o(s, \vec{q}, c) \leq & \sum_{\vec{a}} P(\vec{a}|c, \vec{q}) [R(s, \vec{a}) + \\ & \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) \\ & \cdot P(c'|c) V_o(s', \vec{q}', c)] \end{aligned}$$

Notice that the formula on the right is the Bellman operator for the new controller, applied to the old value function. Denoting this operator T_n , the system of inequalities implies that $T_n V_o \geq V_o$. By monotonicity, we have that for all $k \geq 0$,

$T_n^{k+1}(V_o) \geq T_n^k(V_o)$. Since $V_n = \lim_{k \rightarrow \infty} T_n^k(V_o)$, we have that $V_n \geq V_o$. Thus, the value of the new controller is higher than that of the original controller for all possible initial state distributions.

The argument for changing nodes of the correlation device is almost identical to the one given above. \square

4.4 Local Optima

Although bounded backups give nondecreasing values for all initial state distributions, convergence to optimality is not guaranteed. There are a couple of factors contributing to this. First is the fact that only one local controller, or the correlation device, is improved at once. Thus, it is possible for the algorithm to get stuck in a suboptimal Nash equilibrium in which each of the controllers and the correlation device is optimal with the others held fixed. It is an open problem whether there is a linear program for updating more than one controller at a time.

Of course, a bounded backup does not find the *optimal* parameters for one controller with the others held fixed. Thus, a sequence of such updates may converge to a local optimum without even reaching a Nash equilibrium. For POMDPs, Poupart and Boutilier [2003] provide a characterization of these local optima, and a heuristic for escaping from them. This could be applied in our case, but it would not address the suboptimal Nash equilibrium problem.

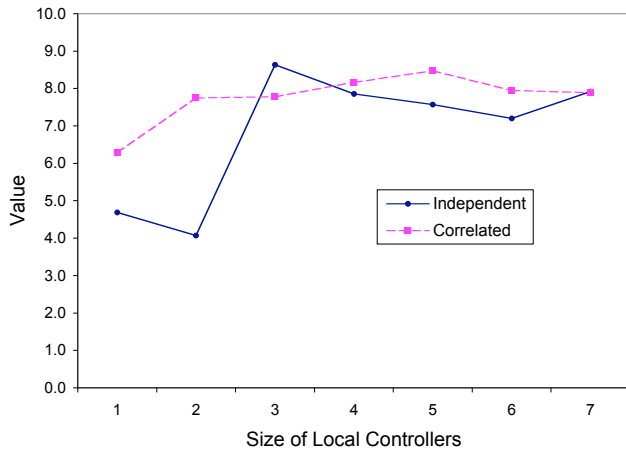
5 Experiments

We implemented bounded policy iteration and tested it on two different problems, an idealized networking scenario and a problem of navigating on a grid. Below, we describe our experimental methodology, the specifics of the problems, and our results.

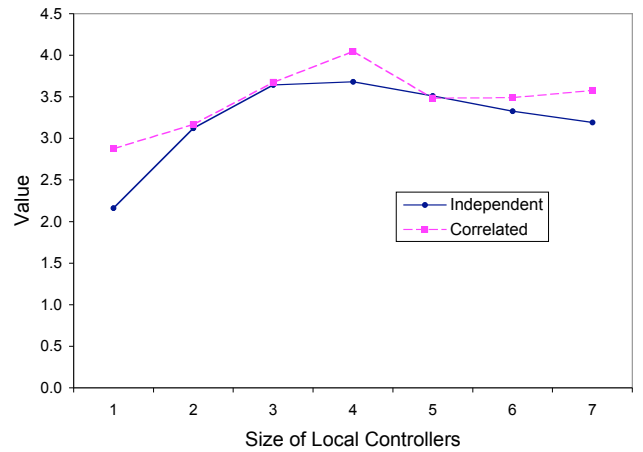
5.1 Experimental Setup

Although our algorithm guarantees nondecreasing value for all initial state distributions, we chose a specific distribution to focus on for each problem. Experiments with different distributions yielded qualitatively similar results.

We define a *trial run* of the algorithm as follows. At the start of a trial run, a size is chosen for each of the local controllers and the correlation device. The action selection and transition functions are initialized to be deterministic, with



(a)



(b)

Figure 2: Average value per trial run plotted against the size of the local controllers, for (a) the multi-access broadcast channel problem, and (b) the robot navigation problem. The solid line represents independent controllers (a correlation device with one node), and the dotted line represents a joint controller including a two-node correlation device.

the outcomes drawn according to a uniform distribution. A step consists of choosing a node uniformly at random from the correlation device or one of the local controllers, and performing a bounded backup on that node. After 50 steps, the run is considered over. In practice, we found that values usually stabilized within 15 steps.

We varied the sizes of the local controllers from 1 to 7 (the agents' controllers were always the same sizes as each other), and we varied the size of the correlation device from 1 to 2. Thus, the number of joint nodes ranged from 1 to 98. Memory limitations prevented us from using larger controllers. For each combination of sizes, we performed 20 trial runs. We recorded the highest value obtained across all runs, as well as the average value over all runs.

5.2 Multi-Access Broadcast Channel

Our first domain is an idealized model of control of a multi-access broadcast channel [Ooi and Wornell, 1996]. In this problem, nodes need to broadcast messages to each other over a channel. Only one node may broadcast at a time, otherwise a collision occurs. The nodes share the common goal of maximizing the throughput of the channel.

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 observes its own buffer, and whether the previous step contained a collision, a successful broadcast, or nothing attempted.

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 4 states, 2 actions per agent, and 5 observations per agent. The discount factor is 0.9. The start state distribution is deterministic, with

the buffer for agent 1 containing a message and the buffer for agent 2 being empty.

5.3 Meeting on a Grid

In this problem, we have two robots navigating on a two-by-two grid with no obstacles. Each robot can only sense whether there are walls to its left or right, and the goal is for the robots to spend as much time as possible on the same square. The actions are to move up, down, left, or right, or to stay on the same square. When a robot attempts to move to an open square, it only goes in the intended direction with probability 0.6, otherwise it either goes in another direction or stays in the same square. Any move into a wall results in staying in the same square. The robots do not interfere with each other and cannot sense each other.

This problem has 16 states, since each robot can be in any of 4 squares at any time. Each robot has 4 observations, since it has a bit for sensing a wall to its left or right. The total number of actions for each agent is 5. The reward is 1 when the agents share a square, and 0 otherwise, and the discount factor is 0.9. The initial state distribution is deterministic, placing both robots in the upper left corner of the grid.

5.4 Results

For each combination of controller sizes, we looked at the best solutions found across all trial runs. The values for these solutions were the same for all controller sizes except for the few smallest.

It was more instructive to compare average values over all trial runs. Figure 2 shows graphs of average values plotted against controller size. We found that, for the most part, the average value increases when we increase the size of the correlation device from one node to two nodes (essentially moving from independent to correlated).

For small controllers, the average value tends to increase with controller size. However, as the controllers get larger,

there is no clear trend. This behavior is somewhat intuitive, given the way the algorithm works. For new node parameters to be acceptable, they must not decrease the value for any combination of states, nodes for the other controllers, and nodes for the correlation device. This becomes more difficult as controllers get larger, and thus it is easier to get stuck in a local optimum.

Improving multiple controllers at once would help to alleviate the aforementioned problem. As mentioned earlier, we do not currently have a way to do this using linear programming, and it thus remains an interesting topic for future work.

6 Conclusion and Future Work

We have presented a bounded policy iteration algorithm for DEC-POMDPs. Besides the fact that it uses finite memory, the algorithm has a number of other appealing theoretical guarantees. First, by using correlated joint controllers, we can achieve higher value than with independent joint controllers of the same size. Second, assuming a constant number of agents, each iteration of the algorithm completes in polynomial time. Finally, monotonic value improvement is guaranteed for all states on each iteration.

Our empirical results are encouraging. By bounding the size of the controller, we are able to achieve a tradeoff between computational complexity and the quality of the approximation. Up to a point, increasing the sizes of the local controllers leads to higher values on average. After this point, average values tend to level off or decrease. Increasing the size of the correlation device leads to higher value, which is consistent with our theoretical results.

For future work, there are many more experiments that can be done with bounded policy iteration. For instance, in moving to a larger controller, we could use the previous controller as a starting point, rather than starting over with a random controller. Poupart and Boutilier's [2003] escape technique could be useful here. Also, rather than choosing nodes uniformly at random for updating, we could develop a principled way to order the nodes.

We are also looking into ways of extending the algorithm to handle problems with large numbers of agents. In many problems, each agent interacts with only a small subset of the other agents. This additional structure can be exploited to reduce the size of the problem representation, and it should be possible to extend our algorithm to take advantage of these local interactions.

Finally, it would be interesting to extend bounded policy iteration to the noncooperative setting, where each agent has a separate reward function. One approach is to require that a change in parameters does not lead to a decrease in value for *any* agent. Another approach is to consider just the value function for the agent whose node is being updated. This should move the joint controller towards a Nash equilibrium.

7 Acknowledgments

We thank Martin Allen and Özgür Şimşek for helpful discussions of this work. This work was supported in part by the National Science Foundation under grants IIS-0219606 and IIS-9984952, by NASA under cooperative agreement NCC

2-1311, and by the Air Force Office of Scientific Research under grant F49620-03-1-0090.

References

- [Aumann, 1974] Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1:67–96, 1974.
- [Bernstein *et al.*, 2002] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.
- [Cassandra *et al.*, 1997] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of UAI*, pages 54–61, 1997.
- [Emery-Montemerlo *et al.*, 2004] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of AAMAS*, 2004.
- [Feng and Zilberstein, 2004] Zhengzhu Feng and Shlomo Zilberstein. Region-Based incremental pruning for POMDPs. In *Proceedings of UAI*, pages 146–153, 2004.
- [Hansen *et al.*, 2004] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of AAAI*, pages 709–715, 2004.
- [Hansen, 1998] Eric Hansen. Solving POMDPs by searching in policy space. In *Proceedings of UAI*, pages 211–219, 1998.
- [Meuleau *et al.*, 1999] Nicolas Meuleau, Kee-Eung Kim, Leslie Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of UAI*, pages 417–426, 1999.
- [Nair *et al.*, 2003] Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of IJCAI*, 2003.
- [Ooi and Wornell, 1996] James M. Ooi and Gregory W. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proceedings of the 35th Conference on Decision and Control*, pages 293–298, 1996.
- [Peshkin *et al.*, 2000] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings of UAI*, pages 489–496, 2000.
- [Platzman, 1980] Loren K. Platzman. A feasible computational approach to infinite-horizon partially-observed Markov decision processes. Technical report, Georgia Institute of Technology, 1980. Reprinted in *Working Notes of the 1998 AAAI Fall Symposium on Planning Using Partially Observable Markov Decision Processes*.
- [Poupart and Boutilier, 2003] Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In *Proceedings of NIPS*, 2003.
- [Singh *et al.*, 1994] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of ICML*, 1994.
- [Smallwood and Sondik, 1973] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.
- [Witsenhausen, 1971] Hans S. Witsenhausen. Separation of estimation and control for discrete time systems. *Proceedings of the IEEE*, 59(11):1557–1566, 1971.