Point-based Dynamic Programming for DEC-POMDPs

Daniel Szer, François Charpillet

Point-based Dynamic Programming for DEC-POMDPs

Daniel Szer and **François Charpillet**
MAIA Group
INRIA Lorraine, LORIA
54506 Vandœuvre-lès-Nancy, France
{szer,charp}@loria.fr

Abstract

We introduce point-based dynamic programming (DP) for decentralized partially observable Markov decision processes (DEC-POMDPs), a new discrete DP algorithm for planning strategies for cooperative multi-agent systems. Our approach makes a connection between optimal DP algorithms for partially observable stochastic games, and point-based approximations for single-agent POMDPs. We show for the first time how relevant multi-agent belief states can be computed. Building on this insight, we then show how the linear programming part in current multi-agent DP algorithms can be avoided, and how multi-agent DP can thus be applied to solve larger problems. We derive both an optimal and an approximated version of our algorithm, and we show its efficiency on test examples from the literature.

Introduction

We present a novel approach for decentralized planning under uncertainty using Markov decision processes. We address problems that are inherently decentralized and where synchronization via communication is often either not possible, such as in nanorobotics (Shirai *et al.* 2005), or where the communication act itself constitutes the parameter to be optimized, such as in computer networks (Altman 2000). Solving these kinds of problems has been shown to be particularly hard (Bernstein *et al.* 2002), which explains the need for efficient algorithms.

On the one hand, dynamic programming and heuristic search algorithms have been proposed to address the general decentralized POMDP problem optimally (Hansen, Bernstein, & Zilberstein 2004), (Szer, Charpillet, & Zilberstein 2005). It has also been shown that the exploitation of particular structure may facilitate the computation of an optimal solution (Becker *et al.* 2004).

On the other hand, there exist several approximative approaches that are based on game theory (Emery-Montemerlo *et al.* 2004) or gradient descent (Peshkin *et al.* 2000), and that often have lower complexity. These techniques however have no strong theoretical connection with the optimal approaches mentioned before, and there is no guarantee that they will produce satisfying results in the general case. Hence the primary motivation of our work: we present the first generalized approach that approximates an optimal planning algorithm for finite-horizon DEC-POMDPs.

The aim of this paper is twofold. We first establish a formal definition of the optimal point-based dynamic programming approach for decentralized POMDPs. This includes a rigorous analysis of the possible amount of information an agent can have when operating in a distributed and partially observable multi-agent environment. We then describe how the optimal approach can be relaxed in order to allow the solution of larger problems. We present experimental results for two test problems from the literature, and we finally indicate some possible directions for future research.

Decentralized POMDPs

The decentralized POMDP formalism is an extension of classical partially observable Markov decision theory to multi-agent domains. The focus hereby lays on the execution of the system: while planning may be centralized, agents are constrained to execute their policies independently from each other.

The DEC-POMDP formalism

We review the DEC-POMDP model as it was introduced by (Bernstein *et al.* 2002). An n-agent DEC-POMDP is given as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}, T, p_0 \rangle$, where

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}(s, a_1, \dots a_n, s')$ is a function of transition probabilities
- $\mathcal{R}(s, a_1, \dots a_n)$ is a reward function
- Ω is a finite set of observations
- $\mathcal{O}(s, a_1, \dots a_n, o_1, \dots o_n, s')$ is a function of observation probabilities
- T is the problem horizon
- p_0 is the start state distribution of the system

The DEC-POMDP model is very expressive and includes in particular the communication of messages between agents as a special case of observations. Solving a DEC-POMDP for a given horizon T and start state distribution p_0 can be

seen as finding a set of n *policies* that maximize the expected joint reward $E[\sum_{t=0}^{T-1} \mathcal{R}(s_t, (a_1, \dots a_n)_t)|p_0]$.

Policies can usually be represented as decision trees. We denote q_i a policy tree and Q_i a set of possible policy trees for agent i. Q_{-i} denotes the sets of policies trees for all agents but agent i. The dynamic programming approach consists in generating incrementally the sets of useful policies for each agent.

Multi-agent belief states

An important concept in partially observable domains is that of an *information state*, which summarizes the amount of information one has about the system at any given time. It has been shown that the probability distribution of the underlying state of the system constitutes a sufficient information state for single-agent POMDPs, often called a *belief state*. The POMDP value iteration algorithm thus considers the belief space to compute an optimal value function.

In decentralized control theory as well as in game theory, a belief about the underlying system state is not a sufficient information to make optimal decisions. Even if the real system state would be known with certainty, there usually remains a problem of coordination (Claus & Boutilier 1998). In general, agents have to reason about the possible future behavior of all teammates in order to chose an optimal action. Some authors have tried to address this by defining beliefs over other agents' beliefs (Gmytrasiewicz & Doshi 2005). Such an approach however might lead to infinite loops of reciprocal beliefs.

We adopt a definition that was first introduced by (Nair *et al.* 2003) and then refined by (Hansen, Bernstein, & Zilberstein 2004). A multi-agent belief state is a probability distribution over system states and possible future policies of all remaining agents. If such a distribution is given, the problem resembles a single-agent decision problem with an augmented transition function. Each agent can then determine its *best response* strategy that optimizes the teams behavior.

Definition 1 (Multi-agent belief state)**.** *A multi-agent belief state b_i for agent i is a probability distribution over system states and policies of all remaining agents: $b_i \in \Delta(\mathcal{S} \times Q_{-i})$.*

Unfortunately, it is not obvious how to estimate a concise distribution over other agents' policies in the general case. This is why the current DP approach for DEC-POMDPs has to consider the infinite space of all distributions that appear to be possible (Hansen, Bernstein, & Zilberstein 2004). We are able to show for the first time how to compute a finite set of candidate belief states, hereby excluding large parts of the belief space that are not relevant or never reachable.

Multi-agent value functions

The multi-agent value function represents the evaluation of *joint policies*. A joint policy δ is a vector of policy trees, one for each agent $\delta = (q_1, \dots q_n)$. Evaluating a joint policy can be done using dynamic programming

$$V(s, \delta) = \sum_{\mathbf{o} \in \Omega^n} P(\mathbf{o}|s, \delta) \left[\sum_{s' \in \mathcal{S}} P(s'|s, \delta, \mathbf{o}) V(s', \delta(\mathbf{o})) \right] \quad (1)$$

where $\mathbf{o} = (o_1, \dots o_n)$ is a joint observation, and $\delta(\mathbf{o})$ is the joint policy of subtrees selected by the agents after observation of $\mathbf{o}$. For a given multi-agent value function, we define agent i's individual value function as

$$V_i(b_i, q_i) = \sum_{s \in \mathcal{S}} \sum_{\delta_{-i} \in Q_{-i}} b_i(s, \delta_{-i}) V(s, \{\delta_{-i}, q_i\}) \quad (2)$$

where δ_{-i} contains policies for all agents but agent i, and $\{\delta_{-i}, q_i\}$ thus denotes a full joint policy.

Dynamic programming for DEC-POMDPs

The dynamic programming algorithm for decentralized POMDPs (Hansen, Bernstein, & Zilberstein 2004) consists of two alternating phases of incremental enumeration and iterated pruning of policies. It is shown in Figure 1. An op-

Given are sets of (t-1)-step policies Q_i^{t-1} for each agent i.

1. For each agent i, the operator first performs a backup on the set of horizon-(t-1) policies Q_i^{t-1} and produces the *exhaustive* set of policies for the next horizon Q_i^t. The policy trees in Q_i^{t-1} will thus constitute the subtrees in Q_i^t.
2. The operator then prunes *dominated* policies. A policy q_i is said to be dominated if its removal does not affect the value function of agent i. Identifying dominated policies can be done by solving the following linear program: a policy q_i for agent i can be pruned from Q_i^t if

$$(\forall b_i)(\exists \tilde{q_i} \in Q_i^t \backslash \{q_i\}) \quad \text{s.t.} \quad V_i(b_i, \tilde{q_i}) \geq V_i(b_i, q_i)$$

Return sets of t-step policies Q_i^t for each agent i.

Figure 1: The multi-agent DP operator

timal joint policy $\delta^* = (q_1^T, \dots q_n^T)$ for a given start state distribution p_0 can then be extracted as follows:

$$\delta^* = \underset{\delta^T \in Q_1^T \times \dots \times Q_n^T}{\arg\max} \sum_{s \in \mathcal{S}} p_0(s) V(s, \delta^T) \quad (3)$$

Unlike for POMDPs, pruning policies for DEC-POMDPs reduces the dimensionality of the belief space of all other agents.

Point-based multi-agent DP

The point-based multi-agent dynamic programming approach addresses two major drawbacks of the DP approach presented in (Hansen, Bernstein, & Zilberstein 2004), namely the computationally expensive linear programming part necessary to identify dominated policies, and the difficulty to exclude those regions of the belief space that are never reachable. We will begin by reviewing the point-based dynamic programming algorithm for single-agent POMDPs.

Point-based DP for POMDPs

Lovejoy was the first one to show that the computation of the optimal POMDP value function can be approximated by discretizing the belief space (Lovejoy 1991) into a regular grid. Since the value function is determined at the grid points only, there is no need for extra pruning as in classical POMDPs: the linear programming operator becomes a simple *max*-operator. The value of any belief state that is not a grid point can be obtained by linear interpolation. The quality of the approximation obviously depends on the resolution of the grid.

More recently, Pineau *et al.* pointed out that a set of relevant *belief points* can be determined more accurately by sampling stochastic trajectories (Pineau, Gordon, & Thrun 2003). This assures that the belief points are distributed in those regions of the belief space that are effectively likely to be visited. Their *point-based value iteration* algorithm also explicitly considers the gradient of the value, and thus generalizes better to unexplored regions of the belief space.

Exact point-based multi-agent DP

The determination of the relevant multi-agent belief points constitutes the basis of our multi-agent dynamic programming algorithm. As opposed to the single-agent POMDP case, where a belief state can simply be obtained by sampling actions and observations, and Bayes' theorem, defining a multi-agent belief state also includes determining adequate distributions over the possible policies of all remaining agents. We denote $b_i^{\mathcal{S},t}$ agent i's belief over states, and $b_i^{j,t}$ agent i's belief over agent j's strategies at time t. $b_i^t = (b_i^{\mathcal{S},t}, b_i^{1,t}, \ldots, b_i^{i-1,t}, b_i^{i+1,t}, \ldots b_i^{n,t})$ then denotes a complete belief state for agent i at time t.

Distributions over policies, so called *mixed strategies*, are often encountered in game theory, and it is known that the usefulness of a single policy usually depends on the entire space of possible mixed strategies of all other agents. It is therefore important to emphasize the particularity of the cooperative multi-agent planning problem, where only some of these distributions will actually be encountered during execution time. The same cannot be guaranteed for the learning case, where the policy of one agent may evolve in some way that is unpredictable to the remaining agents, nor for any competitive scenario, where policies are usually hidden from each other.

Dynamic programming for decentralized POMDPs is a variant of backward induction, in the sense that the set Q_i^t contains those sub-policies that might be selected during the last t steps of execution. Determining the usefulness of a policy $q_i^t \in Q_i^t$ thus translates to the question: what mixed strategies over Q_{-i}^t does agent i have to consider, given that $(T-t)$ time steps have already been passed? As mentioned above, each agent has full knowledge about the entire joint policy δ^T distributed to the agents prior to execution. Given $\mathcal{P}$, $\mathcal{O}$, the first $(T-t)$ steps of joint policy δ^T, and the history of its own observations $h_i^{T-t} = (o_i^1, \ldots o_i^{T-t})$, agent i can therefore compute $P(h_j^{T-t}|h_i^{T-t}, \delta^{T-t})$, a probability distribution over the histories of observations encountered by each remaining agent j. Given such a distribution, and given the complete joint policy δ^T, it is straightforward to compute a distribution over the remaining policies of all agents for the last t time steps. If we denote $\delta^T(h_j^{T-t})$ the remaining sub-policy of agent j after observation of h_j^{T-t}, then the belief of agent i about a particular strategy of agent j, given an observation history h_i^{T-t}, can be determined as

$$P(q_j^t|h_i^{T-t}, \delta^{T-t}) = \begin{cases} P(h_j^{T-t}|h_i^{T-t}, \delta^{T-t}), \\ \qquad \text{if } \delta^T(h_j^{T-t}) = q_j^t \\ 0, \qquad \text{otherwise} \end{cases} \quad (4)$$

and the belief over agent j's policies thus becomes

$$b_i^{j,T-t}(q_j^t) = \sum_{h_i^{T-t}} P(q_j^t|h_i^{T-t}, \delta^{T-t}) P(h_i^{T-t}|\delta^{T-t}) \quad (5)$$

The estimation of the observation histories h_j^{T-t} can be carried out explicitly and at any time t during execution, because the system proceeds forward in time. The same however is not true during planning, since the planning process is oriented backwards. This means in particular that the *prior policy* δ^{T-t} for the first $(T-t)$ time steps has not been determined at iteration step t.

Evaluating a policy $q_i^t \in Q_i^t$ thus involves (a) determining a possible prior joint policy of actions δ^{T-t} that has been executed before, (b) selecting a history of observations h_i^{T-t} for agent i, (c) determining a distribution over the possible histories of observations h_{-i}^{T-t} for all other agents, consistent with both q_i^t and h_i^{T-t}, and (d) determining a multi-agent belief state, which is a consistent distribution over the remaining policies of all agents, given the policy sets Q_{-i}^t, already computed by the algorithm.

The general point-based multi-agent DP operator is summarized in Figure 2. We call *exact point-based multi-agent dynamic programming* the version of the algorithm that repeats the steps (a) to (d) for all possible configurations. The exhaustive generation of belief states is given in Figure 3. The exact point-based multi-agent dynamic programming approach usually keeps fewer policies for the next iteration than the DP based on linear programming, since it only considers those regions of the belief space that are physically possible. It is however constrained to consider exponentially many configurations in order to establish the exhaustive set of belief points.

Approximate point-based multi-agent DP

We now derive an approximate version of the point-based DP algorithm by avoiding the exhaustive generation of policies and belief states given by the two **all** statements in Figure 2.

Generation of joint policies Generating the exhaustive set of prior policies is usually impractical. We adopt a strategy that has already been used in the single-agent case in order to determine candidate belief states (Pineau, Gordon, & Thrun 2003): we sample from the set of possible prior policies. Those policies that are most likely to be spread out far away from each other are retained. The Manhattan distance can be used as a simple metric between policy trees.

Given are sets of (t-1)-step policies Q_i^{t-1} for each agent i.

1. Generate **all** possible joint policies δ^{T-t}.
2. For each agent i:
   (a) For each possible history h_i^{T-t}:
      i. $\overline{Q}_i^t$ holds the set of horizon-t candidate policies created by exhaustive backup from Q_i^{t-1}.
      ii. Generate the set of **all** possible belief states $B(h_i^{T-t}, \delta^{T-t}, \overline{Q}_{-i}^t)$.
      iii. For each $b_i^t \in B(h_i^{T-t}, \delta^{T-t}, \overline{Q}_{-i}^t)$:

$$Q_i^t \leftarrow Q_i^t \bigcup \left\{ \arg\max_{q_i^t \in \overline{Q}_i^t} V(b_i^t, q_i^t) \right\}$$

   (b) Set $\overline{Q}_i^t \leftarrow Q_i^t$.

Return sets of t-step policies Q_i^t for each agent i.

Figure 2: The exact point-based multi-agent DP operator

Generation of belief states The complexity in generating all possible belief states is due to the exponentially many possible assignments of the policies in $\overline{Q}_{-i}^t$ to the leaves of δ^{T-t}. Some leaves however are much less likely to be visited than others, and we can specify the error we possibly commit if we avoid testing all possible policy subtrees at that leaf node, but rather assign a subtree at random. For a given history h_i of agent i, we denote γ the probability of some compatible set of histories h_{-i} for all agents but i:

$$\gamma = P(h_{-i}|h_i, \delta^{T-t}) \tag{6}$$

The maximal loss of rewards we can expect when the worst possible policies are assigned to the leaves corresponding to h_{-i} is $\gamma(T-t)(R_{max} - R_{min})$. If we allow an error of ϵ, then we can avoid considering all possible assignments of policies to the leaves h_{-i} if

$$\gamma \leq \frac{\epsilon}{(T-t)(R_{max} - R_{min})} \tag{7}$$

Our main theorem states that the exact point-based multi-agent DP algorithm is indeed optimal.

Theorem 1. *The exact point-based multi-agent dynamic programming algorithm produces a complete set of useful policies for each agent.*

Proof. We first emphasize that the belief set generated by the algorithm in Figure 3 is exhaustive, which means that it generates all possible beliefs for a given local history of observations and the policy sets of all remaining agents. The point-based multi-agent DP operator in Figure 2 then generates the exhaustive sets of optimal policies for the given belief states. The proof is by induction: if each Q_i^{t-1} contains a complete set of useful policies for horizon ($t-1$), then the belief set generation and the point-based multi-agent DP operator guarantee that each Q_i^t contains a complete set of useful policies for horizon t. The induction hypothesis is satisfied if $(\forall i)(Q_i^0 = \emptyset)$, and computations begins with $t = 1$. □

Given are $h_i^{T-t}, \delta^{T-t}, \overline{Q}_{-i}^t$.

1. Compute the belief over states $b_i^{\mathcal{S}}$. For all $s \in \mathcal{S}$ and all possible h_{-i}^{T-t}:

$$b_i^{\mathcal{S}}(s) \mathrel{+}= P(h_{-i}^{T-t}|h_i^{T-t}, \delta^{T-t}) P(s|h_i^{T-t}, h_{-i}^{T-t})$$

2. Compute the beliefs over policies $b_i^1, \ldots b_i^n$. For all possible assignments of the policies in $\overline{Q}_{-i}^t$ to the leaves of δ_{-i}^{T-t}:
   (a) If policy q_j is associated with the leaf node corresponding to history h_j^{T-t} of tree δ_j^{T-t}, then:

$$b_i^j(q_j) = P(h_{-i}^{T-t}|h_i^{T-t}, \delta^{T-t})$$

   (b) Build a belief state $b_i = (b_i^{\mathcal{S}}, b_i^1, \ldots b_i^n)$.
   (c) $B(h_i^{T-t}, \delta^{T-t}, \overline{Q}_{-i}^t) \leftarrow B(h_i^{T-t}, \delta^{T-t}, \overline{Q}_{-i}^t) \bigcup b_i$

Return the belief set $B(h_i^{T-t}, \delta^{T-t}, \overline{Q}_{-i}^t)$.

Figure 3: Belief set generation

We now give a strong theoretical result about the optimality of the approximate point-based multi-agent DP algorithm.

Theorem 2. *The policy sets produced by the approximate point-based multi-agent dynamic programming algorithm may be incomplete, and they may contain dominated policies that do not appear in the resulting sets of the optimal algorithm.*

Proof. We first argue that the resulting policy sets may be incomplete. This is essentially due to the way belief states are sampled. We then show that the resulting policies may be dominated. This is due to the incompleteness of the policy sets: given an incomplete set of policies Q_i^{t-1}, the set of candidate policies $\overline{Q}_i^t$ for the next horizon is necessarily incomplete as well. This might result in a policy $q_i^t \in \overline{Q}_i^t$ that appears to be optimal at some given belief state, but that is actually dominated by another policy $\tilde{q}_i^t \notin \overline{Q}_i^t$ which could not be considered. □

Experimental results

We implemented both the exact and the approximate point-based multi-agent DP algorithm. The approximate version samples a single prior joint policy for the first **all** statement, but considers the exhaustive set of belief points for the second **all** statement (cf. Figure 2). The values were averaged over 10 trials. We now present initial results of our point-based multi-agent DP approach on two different test problems. The first one is a simplified version of a multi-access broadcast channel problem, introduced and described in (Hansen, Bernstein, & Zilberstein 2004). It simulates a network of two agents that have to maximize the amount of messages that can be exchanged via a shared channel. Collisions occur if both agents attempt to access the channel at the same time. The second problem is the multi-agent version of Kaelbling's Tiger problem, presented in (Nair *et al.*

2003). It simulates a cooperative gambling task, where two agents have to guess independently the location of a common reward, but where uncoordinated behavior is quickly penalized.

The major obstacle for the dynamic programming algorithm in (Hansen, Bernstein, & Zilberstein 2004) is memory. On the channel problem, the algorithm runs out of memory at horizon 5, and on the tiger problem, it does so at horizon 4. Figure 4 presents comparison results with both the exact and the approximate point-based multi-agent DP algorithm for the channel problem. Hansen *et al.*'s algorithm keeps 300 dominant policies at iteration 4 and cannot enumerate the exhaustive set of policies necessary for iteration 5. The exact point-based approach already keeps much fewer policies in memory, which means that most of the 300 policies identified by Hansen *et al.*'s approach lay in regions of the belief space that will never be visited. The approximate point-based approach is even less memory demanding. The number of policies kept by the point-based approaches also depends on the problem horizon, since a different horizon implies different prior policies and thus different belief sets.

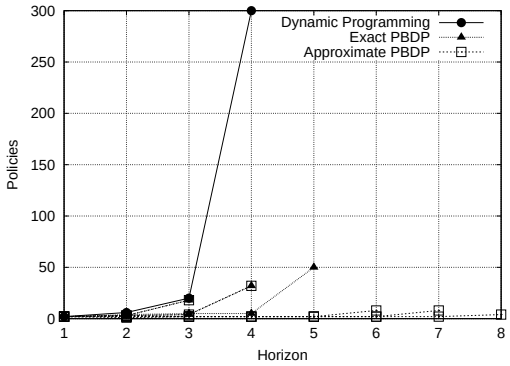

Figure 4: The number of useful policies evaluated by Hansen *et al.*'s DP approach, the exact point-based DP approach, and the approximate point-based DP approach on the multi-access channel problem. The size of the policy set for the point-based approaches depends on the horizon: the exact approach was tested up to horizon 5, and the approximate approach up to horizon 8, which means that there are actually 5 (resp. 8) different curves for each case.

Since the point-based algorithms have to consider much fewer policies, and because they do not include the linear programming part necessary to identify dominated policies, their runtime is usually expected to be shorter. Figure 5 shows a runtime comparison on the multi-access channel problem for all three approaches. Our exact point-based DP approach is indeed faster than the previous DP approach, and our approximate point-based DP approach is even more competitive. The runtime of the point-based approximation surprisingly goes down for horizons 6 and 7. This might be explained by the fact that a different horizon results in a different prior policy, and thus in a different set of belief points. It is possible that a smaller set of policies is sufficient to cover up a larger set of belief points. Note that the ordinate is scaled logarithmically.

We finally give a comparative overview of the solutions that can be obtained with the DP algorithm from (Hansen, Bernstein, & Zilberstein 2004), the heuristic search algorithm MAA* from (Szer, Charpillet, & Zilberstein 2005), and our point-based approaches. On the broadcast channel problem, the point-based DP approaches are definitely the most competitive algorithms, since they produce optimal or nearly optimal policies and enable to solve for larger problems. On the more difficult tiger problems, the point-based approaches are more competitive than the previous DP algorithm, but they do not permit to outperform the heuristic search algorithm.

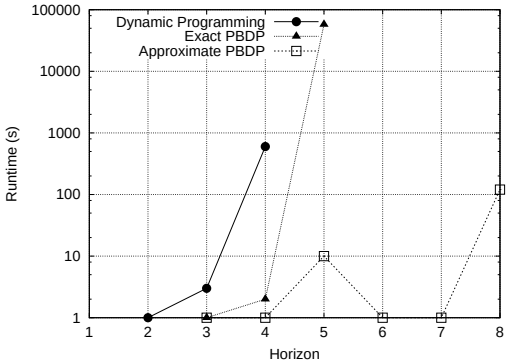

Figure 5: The effective runtime of the three approaches on the multi-access channel problem, measured in seconds and plotted on a logarithmic scale. The runtime of the approximate point-based DP approach is averaged over 10 trials.

Channel	**T=3**	**T=4**	**T=5**	**T=6**	**T=7**	**T=8**
DP	2.99	3.89				
MAA*	2.99	3.89				
ex. PBDP	2.99	3.89	4.79			
ap. PBDP	2.99	3.89	4.70	5.69	6.59	7.40

Tiger A / B	**T=2**	**T=3**	**T=4**
DP	-4.00 / 20.0	5.19 / 30.0	
MAA*	-4.00 / 20.0	5.19 / 30.0	4.80 / 40.0
ex. PBDP	-4.00 / 20.0	5.19 / 30.0	
ap. PBDP	-4.00 / 20.0	5.19 / 30.0	4.80 / 40.0

Figure 6: Values of the resulting policies for Hansen *et al.*'s DP approach, Szer *et al.*'s MAA* approach, and our point-based multi-agent DP algorithms. The test scenarios were the multi-access broadcast problem, and two versions of the multi-agent tiger problem. Blanks indicate that the algorithms have either run out of memory or out of time.

Future work

The point-based multi-agent DP algorithm is a first attempt in exploiting the particular structure of the multi-agent belief

space. There is however much place for future work, and we want to point out some possible directions.

While Equation (7) establishes an error bound for excluding unlikely policy distributions, it is an open question if this result can be generalized to bound the error in the approximate case, where the exhaustive set of prior policies and belief states is replaced with a small set of samples.

We have so far dealt with finite-horizon problems. It is known from MDP and POMDP theory that solving infinite horizon problems using value-based techniques basically translates to an infinite application of the same operators as in the finite horizon case. Policy-based approaches require in addition to restrict the class of policies one wants to consider, since infinite-horizon (DEC-)POMDPs are otherwise not computable. The theory of *finite state controllers* has recently been applied to solve infinite-horizon DEC-POMDPs (Bernstein, Hansen, & Zilberstein 2005), (Szer & Charpillet 2005), and we are confident that point-based multi-agent dynamic programming can be extended in a similar way.

Although the point-based multi-agent DP approach already restricts the computation of the value functions to those regions of the belief space that are effectively possible, there are often still too many belief points that have to be considered. Varakantham *et al.* recently showed how the belief space can be partitioned such that a dominant policy can be determined for each belief region (Varakantham *et al.* 2006). Their algorithm is only guaranteed to produce locally optimal solutions, but it might give an insight on how relevant belief regions for a point-based approach could be determined efficiently.

Discussion

We have presented point-based multi-agent dynamic programming, a new discrete planning algorithm for solving finite-horizon DEC-POMDPs. Our algorithm gives new insights into the computation of relevant multi-agent belief states, and enables the class of decentralized DP algorithms to solve larger problems. We believe this to be a major step towards more efficient approximation techniques that are able to take into account the particular structure of the belief space. Experimental results show that our algorithm produces optimal or near optimal policies, and we are confident that it can be used as a basis for a wide range of interesting research.

Acknowledgments

The authors would like to thank Dan Bernstein for making available his DP code for solving POSGs.

References

Altman, E. 2000. Applications of markov decision processes in communication networks: A survey. Technical Report RR-3984, INRIA Sophia-Antipolis.

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

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.

Bernstein, D. S.; Hansen, E. A.; and Zilberstein, S. 2005. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*.

Claus, C., and Boutilier, C. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, 746–752.

Emery-Montemerlo, R.; Gordon, G.; Schneider, J.; and Thrun, S. 2004. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the 3rd AAMAS*.

Gmytrasiewicz, P. J., and Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research* 24:49–79.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artificial Intelligence*.

Lovejoy, W. S. 1991. Computationally feasible bounds for partially observed markov decision processes. *Operations Research* 39(1):162–175.

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 18th International Joint Conference on Artificial Intelligence*.

Peshkin, L.; Kim, K.-E.; Meuleau, N.; and Kaelbling, L. 2000. Learning to cooperate via policy search. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.

Shirai, Y.; Osgood, A. J.; Zhao, Y.; Kelly, K. F.; and Tour, J. M. 2005. Directional control in thermally driven single-molecule nanocars. *Nano Letters* 5(11).

Szer, D., and Charpillet, F. 2005. An optimal best-first search algorithm for solving infinite horizon dec-pomdps. In *Proceedings of the 16st European Conference on Machine Learning*.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

Varakantham, P.; Nair, R.; Tambe, M.; and Yokoo, M. 2006. Winning back the cup for distributed pomdps: Planning over continuous belief spaces. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*.