Stochastic Planning for Weakly-Coupled Distributed Agents

AnYuan Guo Victor Lesser
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{anyuan,lesser}@cs.umass.edu

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

planning, game theory

1. INTRODUCTION

Partially observable stochastic games (POSGs) provide a powerful framework for modeling multi-agent interactions. While elegant and expressive, this framework has been shown to be computationally intractable [1]. An exact dynamic programming algorithm for POSGs has been developed recently, but due to high computational demands, it has only been demonstrated to work on extremely small problems. Several approximate approaches have been developed [3, 5], but they lack strong theoretical guarantees. In light of these theoretical and practical limitations, there is a need to identify special classes of POSGs that can be solved tractably.

One dimension along which computational gain can be leveraged is by exploiting the independence present in the problem dynamics. In this paper, we examine a class of POSGs where the agents only interact loosely by restricting one another's available actions. Specifically, rather than having fixed action sets, each agent's action set is a function of the global state. The agents are independent from one another otherwise, i.e. they have separate transition and reward functions that do not interact. This class of problems arises frequently in practice. Many real world domains are inhabited by self-interested agents that act to achieve their individual goals. They may not affect each other in any way, except for occasionally putting restrictions on what each other can do.

The best-known solution concepts in game theory are that of computing Nash equilibrium and performing strategy elimination. Our work addresses both of these solution concepts. First, we show that finding a Nash equilibrium where each agent achieves reward at least k is NP-hard. Then, we present the MAAE algorithm, an exact algorithm that performs iterated elimination of dominated strategies. It is able to tackle much larger problems than algorithms for the general class by exploiting the independence among agents.

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

2. FORMAL MODEL

In this section, we will present a formal definition of our model. An n-agent POSG with state-dependent action sets can be defined as $\langle\{S_i\},\{s_i^0\},\{A_i\},\{B_i\},\{P_i\},\{R_i\}\rangle$, where,

- S_i is the state space of agent i. $S = S_1 \times S_2 \times \ldots \times S_n$ denotes the joint state set.
- $s_i^0 \in \Delta(S_i)$ is the initial state distribution of agent i.
- A_i is the action set of agent i. $A = A_1 \times A_2 \times \ldots \times A_n$ denotes the joint action set.
- $B_i : S \rightarrow 2^{A_i}$ is the available action function that maps a joint state to the set of available actions for agent i. $B_i(s) \subset A_i$ for all $s \in S_i$.
- $P_i : S_i \times A_i \times S_i \rightarrow [0,1]$ is the transition function of agent i. The joint transition function is completely factored, where $P((s'_1 \ldots s'_n) \mid (s_1 \ldots s_n),(a_1 \ldots a_n)) = \prod_{i=1}^{n} P_i(s'_i|s_i,a_i)$.
- $R_i : S_i \times A_i \rightarrow \Re$ is the reward function for agent i. This states that the rewards of the agents depend only on the local state and the action taken by agent i.

DEFINITION 1. *A* local policy π_i *of agent i is a mapping from local states and available action sets $\langle s_i, B_i(s)\rangle$ to actions in the available action set $B_i(s)$. The joint state s is a tuple of local states, where s_i denotes the local state of agent i. A* joint policy, $\pi = \langle\pi_1 \ldots \pi_n\rangle$, *is defined to be a tuple of local policies, one for each agent.*

There are several key differences between our model and the general POSG model. First, the power of our model comes from the *available action function*, B, which maps joint states to available actions for each agent. Secondly, our

model does not have an observation function. Each agent has a complete view of its own local state. Partial observability arises as a result of an agent's limited view of the the states of *other* agents but not of its own. Finally, the local policy of each agent is no longer a function of the local state only but a function of the available action set as well.

Here is an example of a POSG with state-dependent action sets. Two autonomous rovers are exploring an unknown terrain and collecting rock samples. There is a river with a narrow bridge on it that only allows one rover to pass at a time. To simplify the illustration, we will assume the rovers have two states {*on land, on bridge*}, and three actions each {*move, get on bridge, pick up rock*}. The rovers receive reward for the number of rocks picked up. The state-dependent action set is used to constrain the rover's actions such that only one is allowed on the bridge at a time. For example, the available actions for rover 1 are specified as follows, $B_1(\langle s_1 = \textit{on land}, s_2 = \textit{on land}\rangle)$ = {*move, get on bridge, pick up rock*}, $B_1(\langle s_1 = \textit{on land}, s_2 = \textit{on bridge}\rangle)$ = {*move, pick up rock*}, and analogously for rover 2.

3. COMPLEXITY

We state our decision problem, denoted as POSG-NE, as follows: given a POSG with state-dependent action sets, $G = \langle \{S_i\}, \{s_i^0\}, \{A_i\}, \{B_i\}, \{P_i\}, \{R_i\}\rangle$, does there exist a Nash equilibrium where all agents have expected utility at least k?

THEOREM 1. *POSG-NE is NP-hard even in problems involving 2 agents and a horizon of 2.*

Proof: We will present a reduction from NFG-NE, a known NP-hard problem [2]. This is the problem of determining the existence of a Nash equilibrium where all agents have expected utility at least k in a 2-agent normal form game, $N = \langle \Sigma_1, \Sigma_2, U_1, U_2\rangle$.

The basis for the reduction rests on the observation that normal form games can be thought of as degenerate POSGs in which each agent only has one state and is allowed to take only one action from that state. When encoding the utility function of the normal form game using constructs in our problem, the challenge lies in the fact that the utility is specified as a function of the actions of *both* agents, whereas, in our model, both the transition and the reward function only have access to the *local* state and action.

To solve this problem, we add auxiliary states to memorize the action taken at each state. We also add composite actions to the action set that corresponds to all possible pairs of actions by each agent. The state-dependent action set restricts the legal actions from the auxiliary states to be the single composite action that corresponds to the pair of actions taken by the agents. The reward for taking the composite action simply corresponds to the utility of the pair of component primitive actions in the normal form game. Therefore, there exists a Nash equilibrium where all agents have expected utility at least k in the POSG if and only if the same is true in the normal form game. □

4. THE MAAE ALGORITHM

We present an algorithm that performs iterated elimination of dominated strategies. The algorithm is able to work directly with the compact representation of a POSG without first converting it to the double exponentially large normal form representation. In this algorithm, first, we first fix the action sets of each agent at each state to the optimistic and pessimistic action sets. The idea behind this is that although an agent cannot predict exactly what actions will be available at each state, it can find out what actions would be available in the best and worst case scenario. In the best case, none of the actions that can be restricted by the state of the other agents are, and in the worst case, all of the actions that can be restricted are in fact unavailable.

For each agent i:

1. For each state s_k, compute the optimistic and pessimistic action sets. Here, $D = \{s \mid s^i = s_k^i\}$.

$$A_{opt}(s_k^i) = \bigcup_{s \in D} B_i(s)$$

$$A_{pes}(s_k^i) = \bigcap_{s \in D} B_i(s)$$

2. Compute the value functions of the 2 MDPs that correspond to alternately fixing the available action sets to the A_{opt} and A_{pes}.

$$U(s) = \max_{a \in A_{opt}(s)} \left[R(s,a) + \sum_{s'} P(s'|s,a) U(s') \right]$$

$$L(s) = \max_{a \in A_{pes}(s)} \left[R(s,a) + \sum_{s'} P(s'|s,a) L(s') \right]$$

3. For each action in each state, derive upper and lower bounds on the action value from the value function bounds $U(s)$ and $L(s)$.

$$Q_U(s,a) = R(s,a) + \sum_{s'} P(s'|s,a) U(s)$$

$$Q_L(s,a) = R(s,a) + \sum_{s'} P(s'|s,a) L(s)$$

4. At each state, eliminate all actions a' whose action value is dominated by another action a, $Q_L(s,a) \geq Q_U(s,a')$.

5. Repeat steps 1 to 4 until no more actions can be eliminated at any state.

Table 1: The iterated action elimination algorithm.

Once we fix the available action sets of each agent at all the states, the agents no longer depend on each other in any way. Our model decomposes to a set of MDPs. For each agent, we solve two MDPs, one using the optimistic action sets and the other using the pessimistic action sets. The solutions will provide us with an upper and a lower bound on the actual value function of each agent. With these upper and lower bounds on the expected values of each state, we can now derive bounds on the expected action values. At each state, dominated actions are pruned. We iterate the action elimination procedure until no more actions can be pruned. Since the number of actions at each state is finite, the procedure will eventually converge.

THEOREM 2. *The iterated action elimination algorithm corresponds to the iterated elimination of very weakly dominated strategies.*

5. EXPERIMENTS

We tested the algorithm on two domains involving machine maintenance and autonomous rover exploration. In manufacturing settings, factories frequently have to make maintenance and repair decisions. The question is how to weigh these two decisions at different stages of the machine life-cycle so as to minimize the likelihood of a machine malfunction.

The detailed specifications for the problem are as follows. There are ten factories, or agents, each owns and operates a machine. Routine maintenance is performed on site. When the machine becomes very old, they need to be serviced by the machine's manufacturer who could only take on one job at a time. This resource constraint is modeled by our state dependent action sets. The state space of each agent is characterized by the current life-cycle of its machine {*new, worn, old, broken*}. Each year, the factory examines the current state of the machine and makes one of the following decisions: *do nothing*, *perform maintenance* or *make repairs*.

Furthermore, each factory has its own reward function. A sample reward function is as follows. There is a high cost, $100,000 associated with a catastrophe (machine breaking while in operation). The factory makes a profit of $5,000 each year when the machine is in operation. Machine maintenance cost $1,000 while repairs cost $10,000. The goal is to maximize the factory's profit.

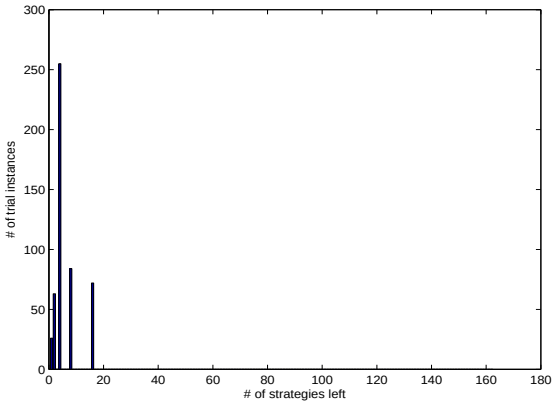

Figure 1: Machine maintenance problem: the distribution of the size of the policy set after the MAAE algorithm is run.

We performed 500 trials by randomly choosing the reward functions of each of the agents. The maintenance costs are chosen in the ranges of $1,000 to $10,000; the cost of a catastrophe ranges from $50,000 to $100,000; finally, the cost of repairs ranges from $10,000 to $50,000. The qualitative trends we found actually correspond well to our intuition. When the machine is relatively new, most of the policies would recommend to do nothing. Unless the penalty for a machine breaking is very high, in which case it adopts a more conservative approach of performing maintenance.

As the machine ages, the algorithm tries to weight the cost of maintenance and repairs. Overall, the MAAE algorithm drastically reduces the size of the policy space. The number of policies before pruning is 162 per agent, while the the number of policies after the algorithm is run, averaged over all trials, is 6. The distribution of the number of policies over all trial runs can be seen in Figure 1 below.

Aside from the large policy space reduction, another key advantage of this algorithm is that it scales well with the number of agents. There are ten agents in this experiment while the general POSG algorithm, due to its high computational demands, has so far only been shown to be effective on 2 agent problems [4]. Next, we tested the MAAE algorithm on a problem with much a larger state space. We examine how the size of the state space and the level of interaction affect the efficiency of the algorithm.

# states	before pruning	after pruning
6	972,000	13
8	34,992,000	168
9	2.1×10^8	69
12	4.5×10^{10}	774
16	5.9×10^{13}	2,198

Table 2: Average size of the policy space before and after action elimination for problems with 3 constrained states.

The second test domain is a simplified 2-agent autonomous rover exploration scenario. The size of each agent's local state space varies from 6 to 16. We introduced up to three constrained states in each agent's state space. 100 trials were run for each size of state space and number of constraints. Table 2 shows detailed results for problems with three constrained states. For problems with one and two constrained states, the final policy space ranges from 3 to 106 and 8 to 252 respectively. In all three cases, the iterated action elimination algorithm is able to reduce the size of the policy space by several orders of magnitude.

6. REFERENCES

[1] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:4, November 2002.

[2] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[3] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the Third Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[4] E. Hansen and D. Bernstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

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