Subjective Approximate Solutions for Decentralized POMDPs

Anton Chechetka
Carnegie Mellon University
antonc@cs.cmu.edu

Katia Sycara
Carnegie Mellon University
katia+@cs.cmu.edu

ABSTRACT

A problem of planning for cooperative teams under uncertainty is a crucial one in multiagent systems. Decentralized partially observable Markov decision processes (DEC-POMDPs) provide a convenient, but intractable model for specifying planning problems in cooperative teams. Compared to the single-agent case, an additional challenge is posed by the lack of free communication between the teammates. We argue, that acting close to optimally in a team involves a tradeoff between opportunistically taking advantage of agent's local observations and being predictable for the teammates. We present a more opportunistic version of an existing approximate algorithm for DEC-POMDPs and investigate the tradeoff. Preliminary evaluation shows that in certain settings oportunistic modification provides significantly better performance.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and Coordination*

General Terms

Algorithms

Keywords

Multiagent planning; Coordination, cooperation, and teamwork; Perception and action

1. INTRODUCTION

The problem of planning for cooperative teams is ubiquitous in multiagent systems, such as urban search and rescue robotics [7] and sensor networks [8]. The problem is hard, especially when uncertainty affects agents' actions and observations and the computational and communicational resources are limited. Compared to single-agent case, planning for multiple agents introduces an additional challenge: an agent in general does not know its teammates' observations and actions. It has to reason about them. A popular formalism for this problem is decentralized partially observable Markov decision processes (DEC-POMDPs). In DEC-POMDP, there are multiple players, each with its own observation model. The state of the system and team's reward (same for all agents) depend on the joint actions of the team. Solving DEC-POMDPs exactly is intractable (NEXP-complete [3]), so the bulk of research has focused on heuristics and methods for solving DEC-POMDPs with additional assumptions about the problem structure [9, 1, 11].

A fundamental problem of reasoning in DEC-POMDP framework is one of using the observations. We argue that there is a tradeoff between acting in accordance to teammates' expectations while not taking into full consideration the observations, and acting relying mostly on current agent observations while possibly ignoring teammates expectations. The former is useful because it is more likely to preserve coherent behavior of the team, while the latter may lead to better performance given the circumstances. In this paper, we examine this tradeoff. Work in [4] has produced an algorithm, called BAGA, that maintains team coherence at the expense of ignoring some information from the observations. We present its modification that relies more on the local observations at possible expense of team coherence. Preliminary experimental evaluation shows that our approach can be advantageous.

2. THE MODEL AND PRIOR WORK

2.1 DEC-POMDP

The Decentralized POMDP model is a tuple

$$\{I, S, A, O, P(s'|s,a), P(o|s,a), R(s,a), b^0, \gamma, m\},$$

where $I = \{1, \dots, n\}$ is a set of agents, S is a finite set of states, $O = \times_{i=1}^{n} O_i$ is a finite set of joint observations (O_i is the set of observations for agent i), $A = \times_{i=1}^{n} A_i$ is a finite set of joint actions, $P(s'|s,a)$ is transition model (probability of transitioning to state s' from s given joint action a), $P(o|s,a)$ is observation model (probability of joint observation o after taking action a in state s), $R(s,a) : S \times A \to \mathbb{R}$ is reward function (immediate reward for the team for taking action a in state s), b^0 is the initial state distribution, $\gamma \in [0,1)$ is discount factor, and m is planning horizon. The model is common knowledge of all agents.

A policy for player i,

$$\pi_i : \times_{t-1}(A_i \times O_i) \to A_i$$

Algorithm 1 BAGA-S main function for player i

```
Require: $b^0, N$
 1: $PDF(c) \leftarrow b^0$, $P(c) \leftarrow 1$
 2: $beliefs_i \leftarrow c$
 3: for $step = 0$ to $n$ do
 4:    $\Pi_i \leftarrow$ computeStrategies($beliefs_i$)
 5:    choose any $c \in beliefs$, execute action $\Pi_i(c)$
 6:    $o_i \leftarrow$ receive observation
 7:    $beliefs_i \leftarrow$ propagateBeliefs($beliefs_i$, $\Pi$, $o_i$)
 8:    $beliefs_i \leftarrow$ compress($beliefs_i$, $N$)
 9: end for
```

is a mapping of individual observation and action history to individual action. The goal of solving a DEC-POMDP is to find a joint policy $\pi = \{\pi_1, \dots, \pi_n\}$ that would maximize the future expected discounted reward $V^\pi = E_\pi[\sum_{t=0}^{m} \gamma^t R]$. This problem is NEXP-complete [3], so in most cases one would need to settle for an approximate solution.

2.2 Prior work

An exact algorithm for solving finite-horizon DEC-POMDPs is presented in [12]. It uses pruning of very weakly dominated policies to improve performance. Approaches exists to solve problems with extra assumption about structure [1], or with restricted policy space [2]. Local search in policy space was also explored [9]. A significant amount of work has focused on finding good communication policies [10, 9].

Few of the above approaches scale beyond small problems. Algorithm BAGA [4, 5] is a scalable approximate approach. Unlike the dynamic programming in [6], it constructs policies from the first steps forward instead of from the last steps backward. As soon as the first step policy is computed, the agents can take the corresponding action, before the rest of the policy length is computed. In our algorithm, we take an approach similar to that of BAGA, but modify the way the observations affect the handling of beliefs by the agents.

3. THE ALGORITHM

The main function of our algorithm (called BAGA-S, or BAGA-subjective) is presented in Alg. 1. It is executed independently by all the agents of the team. The main function of BAGA-S is the same as one of BAGA, up to few implementational details, - all the meaningful differences are in the subfunctions. Execution (line 5:) and planning (4:) are interleaved.

3.1 Beliefs representation and tracking

Each player i maintains a set $beliefs_i$ of particles. Each particle corresponds to a set of possible joint histories. A particle c consists of two fields: $PDF(c)$ is the probability distribution over S and $P(c)$ is the probability of the particle itself. The cumulative PDF over S represented by $belief_i$ is

$$P(S) = \sum_{c \in belief_i} P(c) \times PDF(c)$$

The beliefs for the next step are constructed from current beliefs using Bayesian updates. Assume that Π_i provides us with a mapping from particles of $beliefs_i$ to joint actions (**computeStrategies** computes Π_i). Player i also assumes that whenever the joint observation history corresponds to particle c, the rest of the team performs joint action $\Pi_i(c)$.

Algorithm 2 propagateBeliefs function for player i

```
Require: $beliefs_i, \Pi_i, o_i$
 1: $result \leftarrow \emptyset$
 2: for all $c \in beliefs_i$ do
 3:    $a \leftarrow \Pi_i(c)$
 4:    for all $o \in O$ such that $o$ is consistent with $o_i$ do
 5:       create particle $d$
 6:       $P(d) \leftarrow P(o|S,a)\, PDF(c)$
 7:       $\forall s \in S:\ PDF(d,s) \leftarrow \frac{P(o|s,a)P(s|S,a)^T PDF(c)}{\sum_{s \in S} P(o|s,a)P(s|S,a)^T PDF(c)}$
 8:       $result \leftarrow result \cup d$
 9:    end for
10: end for
11: {normalize the probabilities of particles}
12: $P_o \leftarrow \sum_{d \in result} P(d)$
13: $\forall d \in result\ \ P(d) \leftarrow \frac{P(d)}{P_o}$
14: return $result$
```

It does not hold in general, but can be thought of as a first-order approximation of the infinite recursion of agents' mutual beliefs. Using Bayesian updates with this assumption, we arrive at **propagateBeliefs** function in Alg. 2.

Obviously, because different agents get different observations, the results of executing **propagateBeliefs** will not be the same across the team. This is a major difference of BAGA-S from BAGA: BAGA-S players only consider joint observations $o \in O$ *consistent with local observation*, while in BAGA each player considers *all possible* observations $o \in O$, even those *inconsistent* with local history.

3.2 Beliefs approximation

At each step, the procedure **propagateBeliefs** creates $\frac{|O|}{|O_i|}$ new particles for each particle from $beliefs_i$. This leads to memory requirements exponential in planning horizon. To keep the memory requirements of the algorithm down, we use weighted K-means clustering of particles in the space of probability distributions over $|S|$ with fixed maximum number N of clusters. This part of the algorithm is denoted as **compressBeliefs**. It is similar to the minimal-distance clustering used in [5], but uses different distance metric.

3.3 Policy computation

Given its beliefs, the player needs to be able to compute a policy Π_i that would assign an (approximately) optimal joint action to each particle so as to maximize the expected utility. We assume existence of a heuristic $EV(a, pdf)$ for approximately computing expected discounted reward of taking action a in a distribution $pdf(S)$ over the states and acting optimally afterwars. Denote a_{-i} the action of all players except i. Then, using this heuristic and $beliefs_i$, player i chooses action

$$a_i^* = \arg\max_{a_i \in A_i} \sum_{c \in beliefs_i} P(c) EV((a_i, a_{-i}^*(c)), PDF(c)) \quad (1)$$

where

$$a_{-i}^*(c) = \arg\max_{a_{-i} \in A_{-i}} \left(\max_{a_i} EV((a_i, a_{-i}), PDF(c)) \right). \quad (2)$$

The action a_i^* is the same for all particles, because player i's observation history is consistent with all joint histories of all particles, so i cannot distinguish between the particles. Optimal actions a^* can be computed by enumerat-

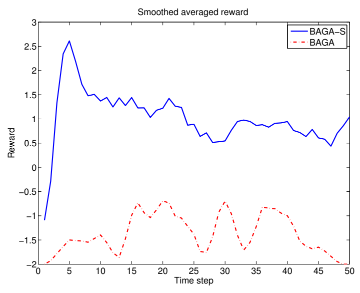

Figure 1: Smoothed average rewards versus time.

ing all possible joint actions, calculating the expectations $EV(a, PDF(c))$, and taking the maxima in (1, 2).

There are several heuristics available to be used as EV. One example is a POMDP heuristic (assuming that agents communicate at every step and reducing the problem to a POMDP). Another is k-step lookahead.

3.4 Discussion and complexity

BAGA always tries to minimize approximation error of the *a priori* distribution over joint histories, regardless of where in the joint history space the players are likely to be in a given run of the algorithm. BAGA-S aims to approximate *a posteriori* distribution. The algorithms trade off *consistency of beliefs across the team* for *relevance of beliefs to the actual outcomes of actions* differently.

There are 3 major elements to consider when determining the complexity of BAGA-S: propagation of beliefs, compression of beliefs, and computation of strategies. It is straightforward to sum up their respective impacts to obtain

$$Time = O\left(mN\left(\frac{|O|}{|O_i|}(|S|^2 + N_{KM}N) + |A|T_{EV}\right)\right)$$

$$Space = O\left(N\left(\frac{|O|}{|O_i|}|S| + T_{EV}\right)\right)$$

where m is the planning (and execution) horizon, N_{KM} is the limit on the number of K-means iterations, N is the upper limit on the number of clusters.

4. EVALUATION

We evaluate BAGA-S and compare it with BAGA using combination of a well-known "multiagent lady and tiger" [9] problem and its single-agent version. Each player had to select an action for the multiagent game as well as for its own instance of a single-agent game. The reward was a sum of individual rewards of the three games.

4.1 Results

As a performance metric for the algorithms, we investigated the evolution of the average reward per step over time. Our hypothesis was that BAGA-S is more robust in the long term, while BAGA has advantage in shorter horizon planning. The maximum number of particles for both algorithms was fixed to 10, the number of random restarts in the local search for Bayes-Nash equilibria in BAGA was 50. As a heuristic for computing expected optimal reward EV, we used 2-step lookahead for both algorithms.

The results for the average reward per step for 50 steps, averaged over 500 runs, are presented in Fig. 1. BAGA-S has better performance even in the initial stages. We attribute this to the fact that the algorithms need to perform clustering as early as on the second time step and in this memory-limited setting BAGA-S has an advantage of concentrating on the relevant parts of the joint history space.

5. CONCLUSION

We have presented an algorithm, BAGA-S, for solving DEC-POMDPs, an important formalism for problems of cooperative multiagent planning under uncertainty. The algorithm is a modification of an existing BAGA algorithm, the main difference being the way of handling local observations by individual agents. BAGA-S sacrifices cross-team coherence of agents' beliefs in attempt to make better use of the observations. Although it does not provide unifom improvement over BAGA, in certain settings, for large-scale problems with long planning horizons, it may be the right choice. Preliminary empirical evaluation is encouraging and shows that making the most use of the local observations by individual agents is important to the team performance.

6. ACKNOWLEDGEMENTS

This research has been sponsored by AFOSR grants FA9550-07-1-0039 and FA9620-01-1-0542 and NSF ITR grant IIS-0205526.

7. REFERENCES

[1] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *JAIR*, 22:423–455, 2004.
[2] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *IJCAI*, 2005.
[3] 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):819–840, 2002.
[4] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. AAMAS*, 2004.
[5] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Game theoretic control for robot teams. In *Proc. ICRA*, 2005.
[6] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proc. AAAI*, 2004.
[7] M. Koes, I. Nourbakhsh, and K. Sycara. Constraint optimization coordination architecture for search and rescue robotics. In *Proc. ICRA*, pages 3977–3982, May 2006.
[8] V. Lesser, C. Ortiz, and M. Tambe, editors. *Distributed Sensor Networks: A Multiagent Perspective*, 2003.
[9] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: towards efficient policy computation for multiagent settings. In *Proc. IJCAI*, 2004.
[10] M. Roth, R. Simmons, and M. Veloso. Decentralized communication strategies for coordinated multi-agent policies. *Multi-Robot Systems: From Swarms to Intelligent Automat*, III, 2005.
[11] M. T. J. Spaan, G. J. Gordon, and N. A. Vlassis. Decentralized planning under uncertainty for teams of communicating agents. In *Proc. AAMAS*, 2006.
[12] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: a heuristic search algorithm for solving decentralized POMDPs. In *Proc. UAI*, 2005.