Solving Large-Scale and Sparse-Reward DEC-POMDPs with Correlation-MDPs*

Feng Wu and Xiaoping Chen

Multi-Agent Systems Lab,Department of Computer Science,
University of Science and Technology of China,
Hefei, 230026, China
wufeng@mail.ustc.edu.cn, xpchen@ustc.edu.cn

Abstract. Within a group of cooperating agents the decision making of an individual agent depends on the actions of the other agents. A lot of effort has been made to solve this problem with additional assumptions on the communication abilities of agents. However, in some real-world applications, communication is limited and the assumptions are rarely satisfied. An alternative approach newly developed is to employ a correlation device to correlate the agents' behavior without exchanging information during execution. In this paper, we apply correlation device to large-scale and spare-reward domains. As a basis we use the framework of infinite-horizon DEC-POMDPs which represent policies as joint stochastic finite-state controllers. To solve any problem of this kind, a correlation device is firstly calculated by solving Correlation Markov Decision Processes (Correlation-MDPs) and then used to improve the local controller for each agent. By using this method, we are able to achieve a tradeoff between computational complexity and the quality of the approximation. In addition, we demonstrate that, adversarial problems can be solved by encoding the information of opponents' behavior in the correlation device. We have successfully implemented the proposed method into our 2D simulated robot soccer team and the performance in RoboCup-2006 was encouraging.

1 Introduction

Multi-Agent systems often require coordination to ensure that a multitude of agents will work together in a globally coherent manner under uncertainty. For some problems, each self-organizing agent has to cooperate to optimize a joint reward function, while having different local observations and limited communication [Kaelbling et al., 1998]. RoboCup [Kitano et al., 1997] is a good example of a cooperative multi-agent system in which the soccer-playing robots like human soccer players have to coordinate their actions by different limited messages.

The infinite-horizon Decentralized Partially Observable Markov Decision Process (DEC-POMDP) framework is one way to model these problems. And

* This work is supported by the NSFC 60275024 and the 973 programme 2003CB317000.

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 208–219, 2008.

Bounded Policy Iteration (BPI) [Bernstein et al., 2005] is currently the leading approximate algorithm which guarantees both bounded memory usage and monotonic value improvement for all initial state distributions. It defines a joint controller to be a set of local controllers along with a correlation device. 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. Namely, an iteration is guaranteed to produce a new controller with value at least as high as the old for every possible initial state distribution. A major drawback of this approach is that it scales exponentially in the number of agents. When we apply it to our soccer simulated team, it can be very slow for the major characteristic of sparse-reward structures which means the joint reward functions are zero everywhere, except for a few states.

In this paper, we just go one step further by developing an alternative approach to handle large-scale DEC-POMDPs with sparse-reward structures.Our new method which aims to reduce, as efficiently as possible, the runtime, solves these problems as follow: a correlation device is firstly calculated by solving Correlation Markov Decision Processes (Correlation-MDPs) and then used to improve the local controller for each agent. Our experimental results show its efficiency and it runs substantially faster, which achieves a tradeoff between computational complexity and the quality of the approximation.

The rest of the paper is organized as follows. The next section gives some comparison of the related works. After that, the DEC-POMDP model, and the basic idea of BPI are introduced. Then, we present our new algorithm and some experimental results in the RoboCup domain.

2 Related Work

Over the last six years, researchers have proposed a wide range of optimal and approximate algorithms for decentralized multi-agent planning. In this paper we focus on cooperation aspect. One important class of algorithms is called MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs [Zser et al., 2005], where multi-agent A* (MAA*), the first complete and optimal heuristic search algorithm for solving decentralized POMDPs with finite horizon was presented. But the algorithm runs out of time very quickly, because the search space grows double exponentially.

The previous approach that is closest in spirit to ours is called Team coordination among robotic soccer players [Matthijs et al., 2002]. It is based on the idea of dynamically distributing roles among the team members and adds the notion of a global team strategy (attack, defend and intercept). Utility functions are used for estimating how well suited a robot is for a certain role. But inconsistencies sometimes occur.

A DEC-POMDP can also be seen as a partially observable stochastic game (POSG) with common payoffs [Emery-Montemerlo et al., 2004]. In this approach, the POSG is approximated as a series of smaller Bayesian games. Interleaving planning and execution, this algorithm finds good solutions for short horizons, but it still runs out memory after horizon 10.

3 The DEC-POMDP Model and BPI Algorithm

The family of Markov decision processes describes discrete stochastic systems that evolve under the influence of one or multiple controllers. With each transition of the system is associated a reward value, the objective of the controller is to select precisely a sequence of actions that maximizes the collection of rewards in the long run. For the case of several distributed but cooperative controllers, their objective is to act selfishly as to maximize the reward collected by the team.

3.1 The DEC-POMDP Model

We base our work on the DEC-POMDP framework introduced by Bernstein [Bernstein et al., 2002], although alternative definitions are equally allowed.

Definition 1 (DEC-POMDP). An n-agent DEC-POMDP is given as a tuple $\langle I, S, \{A_i\}, \{O_i\}, P, R\rangle$, where

- I is a finite set of agents indexed $1, ..., n$
- S is a finite set of states
- A_i is a finite set of actions available to agent i and $\overrightarrow{A} = \times_{i \in I} A_i$ is the set of joint actions, where $\overrightarrow{a} = \langle a_i, ..., a_n\rangle$ denotes a joint action
- O_i is a finite set of observations for agent i and $\overrightarrow{O} = \times_{i \in I} O_i$ is the set of joint observations, where $\overrightarrow{o} = \langle o_1, ..., o_n\rangle$ denotes a joint observation
- P is a set of Markovian state transition and observation probabilities, where $P(s', \overrightarrow{o}|s, \overrightarrow{a})$ denotes the probability that taking joint action $\overrightarrow{a}$ in state s results in a transition to state s' and joint observation $\overrightarrow{o}$
- $R : S \times \overrightarrow{A} \rightarrow \mathcal{R}$ 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 $0 \leq \gamma < 1$. In order to be optimal, the Markov assumption requires a policy to depend on the whole information available to the agent at time t, namely its complete history of past observations and actions. For infinite horizon problems however, this would require a controller to have infinite memory, which is not always possible. Therefore, our algorithm uses stochastic finite-state controllers (FSCs) to represent policies.

Definition 2 (FSC). A stochastic finite-state controller (FSC) is a policy graph, defined as 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
- $\eta_i : Q_i \times A_i \times O_i \rightarrow \Delta Q_i$ is a transition function

The functions ψ_i and η_i parameterize the conditional distribution $P(a_i, q_i' | q_i, o_i)$. For the case of decentralized problems with multiple controllers, the goal is it to find a set of FSCs, one for each agent, such that their concurrent execution maximizes the expected discounted sum of rewards received.The agents' controllers determine the conditional distribution $P(\overrightarrow{a}, \overrightarrow{q}', | \overrightarrow{q}, \overrightarrow{o})$.

Recently, a memory-bounded dynamic programming algorithm was proposed for infinite-horizon DEC-POMDPs [Bernstein et al., 2005] . It extends a joint controller to allow for correlation among the agents. To do this, an additional finite-state machine, called a correlation device is introduced, which provides 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. By using correlated joint controllers, higher value can be achieved than with independent joint controllers of the same size.

Definition 3 (Correlation Device). A correlation device is a tuple $\langle C, \psi \rangle$, where

- C is a set of states
- $\psi : C \rightarrow \Delta C$ is a state transition function

To improve a correlated joint controller, either the correlation device or one of the local controllers can be changed. Both improvements can be done via a bounded backup, which involves solving a linear program.

Following an improvement, the controller can be reevaluate through the solution of a set of linear equations. It has been proofed that performing either of two updates cannot lead to a decrease in value for any initial state distribution. The runtime is polynomial in the sizes of the DEC-POMDP and the joint controller, but exponential in the number of agents.

However, in large-scale and sparse-reward domains, improving the correlation device is very difficult because of the characteristic of sparse reward structures. It can take a very long time for rewards to propagate to distinct states. Thus, it is often possible to get no improvement for just a few of steps. Long steps of search is rather inefficient for large scale problems such as RoboCup. It is obviously worse if multiple choices exist at each state. For example, in the decision making of the RoboCup 2D Simulation League, agents gain non-zero reward for their joint defense actions only when some agent of the team kicks or tackles the ball. Currently, most of the opponents process ball with high quality. Thus, it usually takes thousands of time steps to steal the ball for opponents.

4 Dynamic Programming for Correlation-MDPs

In this section, we describe the Dynamic Programming (DP) algorithm to calculate the correlation device as an approximate alternative to BPI. This method, analogous to the belief propagation, operates by solving a MDP, which can be regarded as a rewards propagation process. For sparse reward structures, each distinct state will have non-zero reward after the DP algorithm.

We can translate the above method to our multi-agent decision making problem by giving the correlation device a concrete meaning.

Definition 4 (Correlation Device State). A correlation device state $c \in C$ is a set of joint actions[1], where $\forall \overrightarrow{a_1}, \overrightarrow{a_2} \in c, |\bar{R}(\overrightarrow{a_1}) - \bar{R}(\overrightarrow{a_2})| \leq \varepsilon$ and $\bar{R}(\overrightarrow{a})$ is the reward function. The reward function for c is $R(c) = max_{\overrightarrow{a} \in c} \bar{R}(\overrightarrow{a})$.

Thus, how to compute the correlation device states is the main job of our method. In RoboCup 2D defensive decisions, the majority of the joint actions has no immediate rewards, or in other wards, it is very difficult in the immediate direct rewards given when the proceeds are sparse reward structures. Only when all opponents can not process the ball any longer, our agents make that defensive effectiveness and have non-zero reward. Running for a better opponent team, it may need to spend tens, hundreds or even more of the cycles to reach this ultimate goal. So far, solving this type of DEC-POMDP problems with the existing methods is not very satisfied.

In the DEC-POMDP model, the reward function $R(s, \overrightarrow{a})$ indicates that joint actions should link to a particular state for the need to obtain rewards. Thus, the pairs of specific state and joint actions which have the maximum rewards can be certainly established. In RoboCup 2D defensive decisions, the structure of rewards is sparse. In order to assess the states of those with zero reward, first is to be done with the goal of state to propagate the reward to distinct states. The propagation process can be described by the following definition of Correlation-MDPs.

Definition 5 (Correlation-MDP). A Correlation-MDP is given as a tuple $\langle \bar{S}, \bar{A}, \bar{P}, \bar{R}, \gamma \rangle$, where

- $\bar{S}$ is the set of states of the DEC-POMDP model
- $\bar{A} = \times_{i \in I} A_i$ is a set of joint actions
- $\bar{R} : S \rightarrow \Re$ and $\bar{R} = max_{a \in \bar{A}} \{R(s, \overrightarrow{a})\}$, where $R(s, \overrightarrow{a})$ is the reward function of the DEC-POMDP model
- $\bar{P} : S \times S \rightarrow [0, 1]$ can be defined:

$$\bar{P}(s'|s) = \frac{P(s') \cdot P(s|s')}{\sum_{s_i \in S} P(s_i) \cdot P(s|s_i)}^2 \tag{1}$$

  where $P(s|s') = max_{\overrightarrow{a} \in \bar{A}} \{P(s|\overrightarrow{a}, s')\}$, $P(s|s_i) = max_{\overrightarrow{a} \in \bar{A}} \{P(s|\overrightarrow{a}, s_i)\}$, $P(s')$ and $P(s_i)$ are the probability of s' and s_i
- γ is the discount factor

A Correlation-MDP compared to the original DEC-POMDP can be viewed as a reverse model. What Correlation-MDP considers is the shift from the target

[1] Note that a correlation device is a finite-state machine, any reasonable definition of the states is allowed.

[2] Known as Bayes formula.

state to the initial one. However, in light of a DEC-POMDP model, it can only provide such a probability $P(s', \overrightarrow{o}|s, \overrightarrow{a})$, which is from the initial state to the target after the execution of joint actions. And the equation (1) is one of the possible solutions (from $P(s|s')$ to $\bar{P}(s'|s)$). In our model, $P(s')$ and $P(s_i)$ are the probability of s' and s_i, which can be used to control the emergence of a particular state (for example, in the area of inside and outside penalty, the strategy for both is different in the RoboCup 2D simulation league). It is easy to encode the information of opponents behavior into the correlation device in this way. Solving a Correlation-MDP means finding a policy $\pi : \bar{S} \rightarrow \bar{A}$ that maximizes the expected reward for each state $s \in \bar{S}$. In fact, the whole process is to build a tree from a root which is the target of the strategy. For the sake of better understanding this process, let us consider a simplified example of a soccer defense situation in Fig. 1. and the tree built by the DP algorithm is in Fig. 2.

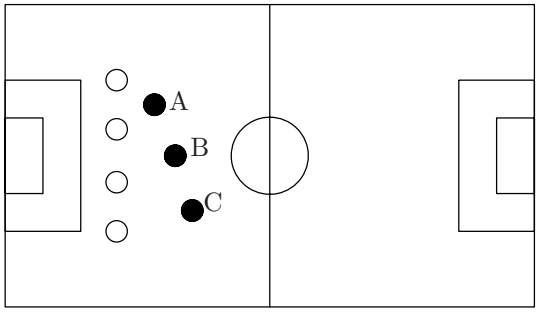

Fig. 1. White circles represent our members, and black ones represent opponents A (the top one), B (the middle one), C (the bolttom one). Each agent only has four type of actions: (formation), (mark, A), (mark, B), (mark, C). The target is to make all the opponents marked. Our side is left in the pitch.

In order to control the time complexity and precision we define *maxChildren* to limit the maximum number of children for each parent. **Algorithm 1** shows the pseudo-code of the DP algorithm.

Proposition 1 (Algorithm 1). *The **Algorithm 1** has a linear space complexity with respect to the maxChildren, and the worst case time complexity is* $O(maxChildren \times |\bar{S}|^2)$.

Proof. The main loop of the algorithm (line 5-14) depends linearly on the size of $\bar{S}$. Inside this loop, the remaining critical operation is in line 9, where the children of each node is updated. Once the variable *maxChildren* is fixed, the number of the children per node is not more than it. In every iteration of the algorithm, no more than *maxChildren* subtrees are constructed. In the worst case, each node has to search all of the states to update its children, therefor, the upper limit

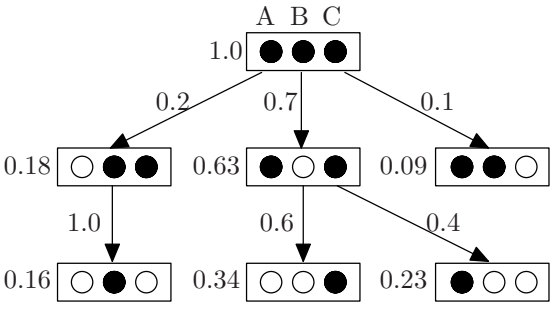

Fig. 2. The value of nodes is $\bar{R}(s)$, and the value of edges is $\bar{P}(s'|s)$. The discount factor is 0.9. The black circle means the opponent is marked, while the white one means not. Note that, in Fig. 1, the easiest action is to mark B, then is to mark A, and to mark C is the most difficult. Obviously the best defense strategy (Marking all the opponent successfully at the same time is usually impossible) should be marking C first, then A, and B finally. And the worst one may be marking A first, then B, and C finally, because the final step is much difficult by following this strategy: Opponent C may control the ball at that time and pass it to A or B, which is very dangerous in some case.

Algorithm 1. Compute the policy tree (correlation tree) Q_{t+1}

$Q_0 \leftarrow$ initialize all states in $\bar{S}$ as a tree
for each $s \in \bar{S}, V_0(s) \leftarrow \bar{R}(s)$
$maxChildren \leftarrow$ max number of children for the tree
$t \leftarrow 0$
loop:
$t \leftarrow t + 1$
for each $s \in \bar{S}$, do {
  $Q_{t+1} \leftarrow$fullBackup(Q_t)
  find a set S' from $\bar{S}$ to satisfy:

  - $\forall s'' \in \bar{S} - \bar{S}', s' \in \bar{S}'\ P(s|s'') \leq P(s|s')$
  - $\forall s' \in \bar{S}' R(s') + V_t(s) \cdot \bar{P}(s'|s) \leq V_t(s')$
  - $|\bar{S}'| \leq maxChildren$

  $V_{t+1}(s') \leftarrow R(s') + V_t(s) \cdot \bar{P}(s'|s)$ for each $s' \in \bar{S}'$
  $V_{t+1}(s'') \leftarrow V_t(s'')$ for each $s'' \in \bar{S} - \bar{S}'$
  $Q_{t+1} \leftarrow$ set s as the parent of each $s' \in \bar{S}'$, set each $s' \in \bar{S}'$ as the children of s.
}
until $max_s|V_{t+1}(s) - V_t(s)| < \varepsilon$
return Q_{t+1}

for the construction is equal to $maxChildren|\bar{S}|$. By choosing $maxChildren$ appropriately, the desired upper limit of the tree length can be per-set, no more than $|\bar{S}|$. Thus, the amount of space grows linearly with $maxChildren$. For the worst case, this is $O(maxChildren \times |\bar{S}|^2)$.

Although the worst case time complexity is $O(maxChildren \times |\bar{S}|^2)$, the average time complexity is much smaller. Increasing the value of $maxChildren$ generally leads to both higher accuracy and time complexity on average. In practice, $maxChildren$ is usually necessary to achieve a tradeoff between computational complexity and the quality of the approximation. When $maxChildren$ is equal to $|\bar{S}|$, every iteration of **Algorithm 1** needs to consider all the possible states, which is the same as a linear program of BPI. The size of these states is exponential in the number of agents. A more detailed analysis of the DEC-POMDPs shows that most of the states are useless, especially with sparse reward structures.

According to the tree computed by **Algorithm 1**, the correlation device can be calculated easily. In general, the assumption below holds: Only when an agent finds a better, or when it finds higher reward cooperation strategy, the current one is changed. It means a rational agent will not choose the worse forms of cooperation from its own local observation. In the RoboCup 2D defensive decision making, an agent in the next step would be impossible to choose a strategy, although in accordance with their own local observation such a strategy might be possible, from the perspective of cooperation it is not likely to exist such a big leap. An upper bound estimate for the reward can be established by the tree calculated above.

Proposition 2 (Algorithm 2). *The **Algorithm 2** returns the near-optimal value for $P(c'|c)$, which is proportional to $\frac{maxChildren}{|S|}$.*

Proof. The usage of $P(c'|c)$ is to correlate the joint controllers in BPI. According to BPI, the procedure for improving the correlation device works by looking for the best parameters satisfying the following inequality:

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

$$P(s', \overrightarrow{o}|s, \overrightarrow{a}) \cdot P(c'|c)V(s', \overrightarrow{q}', c')] \quad (2)$$

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

Note that if $R(c') > R(c)$, the value of $P(c'|c)$ is definitely high by the concrete meaning of c' and c, since the inequality (2) implies that $V(s, \overrightarrow{q}, c) > V(s', \overrightarrow{q}', c')$ is not allowed. The basis of the computational process of **Algorithm 2** is an ideal tree constructed by **Algorithm 1** in which each state has the largest reward propagated from the target. For each node of the tree, the value of its parent is the upper bound of the possible rewards for the next step. Therefore, the rewards which could be calculated for the next step during execution time will range between current rewards and the upper bound. And the process of **Algorithm 2** is just on the premise that the information about agents local observations is unavailable and gives $P(c'|c)$ a reasonable approximation of the distribution while ensuring the inequality (1) non-reducing. This guarantees the near-optimality of the solution. In the process of **Algorithm 2**, if $maxChildren = |\bar{S}|$, all possible states need to be examined, which is the same

Algorithm 2. Compute the state transition function $p(c'|c)$

1: $Q \leftarrow$ a pre-computed correlation tree by **Algorithm 1**;
2: $S_0 \leftarrow$ all of the nodes in Q with the value $V(s) : |V(s) - R(c)| \leq \varepsilon$
3: $S_1 \leftarrow \emptyset, S_2 \leftarrow \emptyset$
4: $\delta \leftarrow 0, \gamma \leftarrow 0$
5: for each $s \in S_0$ {
6: $\delta \leftarrow \delta + V(s) \cdot 1$
7: if($|V(s) - R(c')| \leq \varepsilon$) $\gamma \leftarrow \gamma + V(s) \cdot 1$
8: $s' \leftarrow$the parent of s
9: $S_1 \leftarrow S_1 \cup \{s'\}$
10: $\delta \leftarrow \delta + V(s') \cdot \bar{P}(s|s')$
11: if($|V(s') - R(c')| \leq \varepsilon$) $\gamma \leftarrow \gamma + V(s') \cdot \bar{P}(s|s')$
12: $S' \leftarrow$all the children of s' except s
13: for each $s'' \in S'${
14: if($R(c) - V(s'') \leq \varepsilon$){
15: $S_1 \leftarrow S_1 \cup \{s''\}$
16: $\delta \leftarrow \delta + V(s'') \cdot \frac{\bar{P}(s''|s)}{\bar{P}(s|s')}$
17: if($|V(s'') - R(c')| \leq \varepsilon$) $\gamma \leftarrow \gamma + V(s'') \cdot \frac{\bar{P}(s''|s')}{\bar{P}(s|s')}$
18: }
19: }
20: }
21: return $\frac{\gamma}{\delta}$

as a linear program for BPI, and if $maxChildren < |\bar{S}|$, a lot of states which are useless for the target will be eliminated. Thus, the algorithm works efficiently and the precision is proportional to $\frac{maxChildren}{|\bar{S}|}$.

In **Algorithm 2**, by choosing the value of $maxChildren$, the states which make small contribution to the goal have not been carried out in order to reduce the amount of calculation while guarantee high accuracy. This technique is effective, especially for the sparse reward domain. Further analysis of the RoboCup 2D defense problem shows that many joint actions for a special state are useless. BPI algorithm gives these useless joint actions the same needs of assessment, while our algorithm takes full account of this characteristic, thereby maintaining the high accuracy with a substantial amount of the reduction of the runtime. The parameters $maxChildren$ present an important trade-off: its increase generally increases both precision and runtime. Thus, we can examine this trade-off and identify the best parameters for concrete problems. Though we are not able to give theoretically strict proof on this issue and the exact solution for the accuracy of the two algorithm in this paper, the following experiments proved that our approach is more effective.

Although Correlation-MDPs are very usefull for our soccer robot team, it has two limitations when extended to other applications. Firstly, it requires initial state distribution as input. Secondly,the joint actions of agents and their effort should be easily modeled.

5 Experiments

We performed an experimental feasibility study in RoboCup domain that compares our algorithm and BPI [Bernstein et al., 2005], currently the leading algorithm for solving infinite-horizon DEC-POMDPs with quality guarantees. Below, we describe our experimental methodology, the specifies of the problems, and our results.

As noted above, the correlation device operates independently of the DEC-POMDP process. Thus, the experiments took place in two phases. First, either our algorithm or BPI was run to calculate the correlation device. Secondly, the correlation devices were used to improve local controllers with the some improving procedure.We applied both the BPI and our algorithm to compute the correlation device offline. In BPI we first chose a device node c, and considered changing its parameters for just the first step. New parameters must yield value at least as high for all states and nodes of the other local controllers. For our algorithm, we applied a Correlation-MDP model for all states, and calculated it with some high reward states fixed[3] . Then the correlation device was born by **Algorithm 2**. The computation is performed offline in a centralized way and the final solution is a correlation device which can then be executed by multiple agents in a decentralized way.

In order to encode the information of opponents behavior, we use some learning methods [Ubbo Visser et al., 2003] to determine the value of P(s) in Equation (1). For example, some opponents prefer attacking from the midway, then the probability of midway defending states will be increased; while some rivals like attacking from the sideway, then the probability of sideway defending states will be increased. But a discussion of the learning algorithm is beyond the scope of this paper and thus the following performance comparison does not include results for it.

Experiment 1: We determined how our algorithm and BPI trade off between the number of agents and runtime for the RoboCup Simulation 2D League with the fixed threshold reward (0.8). Our results show that our algorithm is faster than BPI when the number of agents is bigger than 3. For example, our algorithm needed 458.2ms and BPI needed 799.5ms to compute a solution that is only 8 agents under consider (there are 11 agents for each team in the RoboCup Simulation 2D League). Fig. 3 presents the performance comparison.

Experiment 2: We then determined how our algorithm and BPI trade off between runtime and the reward for the RoboCup Simulation 2D League when the number of agents is fixed (7 agents). Our results show that our algorithm is still faster than BPI with the same reward value, by fixing the number of agents. For example, our algorithm needed 533.2ms and BPI needed 1111.9ms to compute a solution that the reward is 0.9. Fig. 4 presents the performance comparison.[4]

[3] The key parameter is the maximum number of children for each node, $maxChildren$, which is related to the runtime and precision. In our experimental domain, $maxChildren = 3$ is sufficient to produce the best solution.

[4] All results are generated on a 2.2GHz/1GB machine using a C++ implementation.

RoboCup-2006 has provided an ideal test bed for our algorithm which implemented in our 2D simulation robot team. This team used the framework and algorithm described in the previous section to improve its highlevel strategy. The main motivation was to improve upon the coordination during defense. Since many different factors contribute to the overall performance of the team, it is difficult to measure the actual effect of the coordination with our new algorithm clearly. However, using this approach, we won all the matches except one ended in a draw 0-0 in the RoboCup-2006 in Bremen, German.

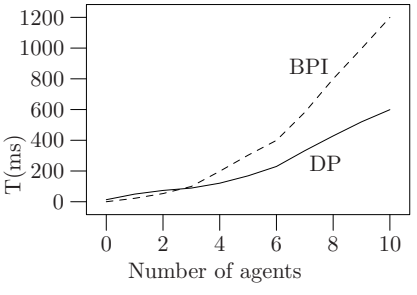

Fig. 3. The result of Experiment 1 with 0.8 as the threshold reward and the discount rate is $\gamma = 0.9$

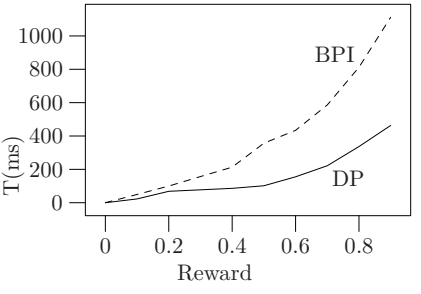

Fig. 4. The result of Experiment 2 with 7 agents and he discount rate is $\gamma = 0.9$

6 Conclusion and Future Work

The decision making in the RoboCup 2D Simulation League can be modeled with DEC-POMDPs. Despite recent advances in solving DEC-POMDPs, state-of-the art solution methods are still either inefficient [Bernstein et al., 2005] or cannot provide guarantees on the quality of the resulting policy. In this paper, we presented a solution method, that avoids both of these shortcomings. Our experimental results show that the algorithm performs very well. The aim of this paper was to provide a first experimental feasibility study to demonstrate its potential. It is future work to study the theoretical properties of this method in more depth (for example, analyze its complexity or extend its error analysis), and extend it (for example, to handle more adversarial problems). The algorithm and representations used in this work open up multiple research avenues for developing effective approximation algorithms for the DEC-POMDP model in the RoboCup domain.

Acknowledgments

We thank Changjie Fan and Benjamin Johnston for helpful discussions of this work and other members of our team (Jiliang Wang, Jingnan Cai) for their contribution to WE2006 on which the experiments of this paper based. We also thank the anonymous reviewers for their valuable comments on the early version of this paper.

References

[Kitano et al., 1997] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: RoboCup: A Challenge problem for AI. AI Magazine

[Kaelbling et al., 1998] Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101, 101–134 (1998)

[Bernstein et al., 2002] Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Mathematics of Operations Research 27(4), 819–840 (2002)

[Bernstein et al., 2005] Bernstein, D.S., Hansen, E.A., Zilberstein, S.: Bounded Policy Iteration for Decentralized POMDPs. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence(IJCAI), Edinburgh, Scotland, pp. 1287–1292 (July 2005)

[Ubbo Visser et al., 2003] Visser, U., Weland, H.-G.: Using online learning to analyze the opponents behavior. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 78–93. Springer, Heidelberg (2003)

[Rabinovich et al., 2003] Rabinovich, Z., Goldman, C.V., Rosenschein, J.S.: The Complexity of Multiagent Systems: The Price of Silence. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems(AAMAS), Melbourne, Australia, pp. 1102–1103 (2003)

[Matthijs et al., 2002] Spaan, M.T.j., Groen, F.C.A.: Team coordination among robotic soccer players. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752. Springer, Heidelberg (2003)

[Emery-Montemerlo et al., 2004] Emery-Montemerlo, R., Gordon, G., Schneider, J., Thrun, S.: Approximate solutions for partially observable stochastic games with common payoffs. In: Kudenko, D., Kazakov, D., Alonso, E. (eds.) AAMAS 2004. LNCS (LNAI), vol. 3394, pp. 136–143. Springer, Heidelberg (2005)

[Zser et al., 2005] Szer, D., Charpillet, F., Zilberstein, S.: MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs. In: Proceedings of the 21st Conference on UAI (2005)