Towards Computing Optimal Policies for Decentralized POMDPs

R. Nair, M. Tambe
Computer Science Department
University of Southern California
Los Angeles CA 90089
{nair,tambe}@usc.edu

M. Yokoo
Cooperative Computing Research Group
NTT Communication Science Labs
Kyoto, Japan 619-0237
yokoo@cslab.kecl.ntt.co.jp

D. Pynadath, S. Marsella
Information Sciences Institute
University of Southern California
Marina del Rey CA 90292
{pynadath, marsella}@isi.edu

Abstract

The problem of deriving joint policies for a group of agents that maximze some joint reward function can be modelled as a decentralized partially observable Markov decision process (DEC-POMDP). Significant algorithms have been developed for single agent POMDPs however, with a few exceptions, effective algorithms for deriving policies for decentralized POMDPS have not been developed. As a first step, we present new algorithms for solving decentralized POMDPs. In particular, we describe an exhaustive search algorithm for a globally optimal solution and analyze the complexity of this algorithm, which we find to be doubly exponential in the number of agents and time, highlighting the importance of more feasible approximations. We define a class of algorithms which we refer to as "Joint Equilibrium-based Search for Policies"(JESP) and describe an exhaustive algorithm and a dynamic programming algorithm for JESP. Finaly, we empirically compare the exhaustive JESP algorithm with the globally optimal exhaustive algorithm.

Introduction

Multiagent systems have moved out of the research lab into a wide range of applications areas. Systems are being developed that employ multiagent technology even for highly critical applications such as multi-satellite control. To meet the challenge of such bold applications, multiagent research will need to provide high-performing, robust designs that are as nearly optimal as feasible. Unfortunately, in practice, research on implemented systems has often fallen short in assessing the optimality of their proposed approaches.

To address this shortcoming, researchers have increasingly resorted to decision-theoretic models as a framework in which to formulate and evaluate multiagent designs. Given some group of agents, the problem of deriving separate policies for them that maximize some joint reward can be modeled as a decentralized POMDP. In particular, the DEC-POMDP model (decentralized partially observable Markov decision process) is a generalization of a POMDP to the case where there are multiple, distributed agents basing their actions on their separate observations. (POMDP is in turn a generalization of a single agent Markov decision process, or MDP, whereby the agent may make decisions on partial observations of the state.) The COMMTDP model (Pynadath & Tambe 2002) is a closely related framework that extends DEC-POMDP by explicitly modeling communication. We describe this framework in detail here to provide a concrete illustration of a decentralized POMDP model. These frameworks allow a variety of key issues to be posed and answered. Of particular interest here, these frameworks allow us to formulate what constitutes an optimal policy for a multiagent system and in principle how to derive that policy. (Nair *et al* 2002) demonstrate how a decentralized POMDP framework can be applied to a complex multiagent domain like RoboCupRescue (Kitano *et al* 1999).

However, with a few exceptions, effective algorithms for deriving policies for such decentralized POMDPS have not been developed. Significant progress has been achieved in efficient single-agent POMDP policy generation algorithms (refs, Monahan, etc). However, it is unlikely such research can be directly carried over to the decentralized case. Finding an optimal policies for decentralized POMDPs is NEXP-complete and therefore provably does not admit a polynomial time algorithm (Bernstein *et al* 2000). In contrast, solving a POMDP is PSPACE-complete (Papadimitriou & Tsitsiklis 1987). As Bernstein et al. note(Bernstein *et al* 2000) , this suggests a fundamental difference in the nature of the problems. Since the reward function is a joint one, the decentralized problem can not be treated as one of separate POMDPs in which individual policies can be generated for individual agents. (For any one action of one agent, there may be many different rewards possible, based on the actions that other agents may take.) Another possibility for solving decentralized POMDPs is to convert the decentralized problem into a single agent problem by assuming free communication among all agents. Unfortunately, in the domains of interest in this work, agents do not have such luxury of communication. Yet another possibility is to simplify the nature of the policies considered for each of the agents. For example, Chades et al. (Chadès *et al* 1994) restrict the agent policies to be memoryless (reactive) policies, thereby simplifying the problem to solving multiple MDPs. Such simplifications reduce the applicability of the approach and essentially sidestep the question of solving DEC-POMDPs. Peshkin

et al. (Peshkin *et al.* 2000) use finite-controllers to represent the partially observable state as an abstraction. However their model does not explicitly model communication.

Thus, there remains a critical need for new efficient algorithms for generating optimal policies in distributed POMDPs. In this paper, we take a first step, by presenting new algorithms for solving decentralized POMDPs, restricted to the case of discrete belief states. First, we present an exact algorithm in the sense that it derives the globally optimal policy via a full search of the space of policies. This exact algorithm is of course expensive to compute which limits its applicability to problems where there is sufficient time to off-line pre-compute such an exact solution. Therefore, we also present two approximate algorithms that search the space of policies incrementally. These algorithms iterate through the agents, finding an optimal policy for each agent assuming the policies of the other agents are fixed. These algorithms, which we refer to as Joint Equilibrium-based Search for Policies (JESP) algorithms, terminate when no improvements to the joint reward is achieved, thus achieving a local optimum similar to a Nash Equilibrium. We conclude with an empirical evaluation of the algorithms.

Background

The COM-MTDP model (Pynadath & Tambe 2002) is a formal model based on multiple communicating POMDPs with joint transition, observation and reward functions. This model allows us to explicitly reason about the cost of communication. This is especially relevant in the search for an optimal solution to a decentralized POMDP because communicating enables agents to synchronize their beliefs allowing for a more optimal solution.

The COM-MTDP Model

Given a team of selfless agents, α, a COM-MTDP (Pynadath & Tambe 2002) is a tuple, $\langle S, A_\alpha, \Sigma_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R\rangle$. S is a set of world states. $A_\alpha = \prod_{i\in\alpha} A_i$ is a set of combined domain-level actions, where A_i is the set of actions for agent i. $\Sigma_\alpha = \prod_{i\in\alpha} \Sigma_i$ is a set of combined messages, where Σ_i is the set of messages for agent i. $P(s_b, \mathbf{a}, s_e) = Pr(S^{t+1}{=}s_e | S^t{=}s_b, A^t_\alpha{=}\mathbf{a})$ governs the domain-level action's effects. $\Omega_\alpha = \prod_{i\in\alpha} \Omega_i$ is a set of combined observations, where Ω_i is the set of observations for agent i. Observation function, $O_\alpha(s, \mathbf{a}, \boldsymbol{\omega}) = Pr(\Omega^t_\alpha{=}\boldsymbol{\omega} | S^t{=}s, A^{t-1}_\alpha{=}\mathbf{a})$ specifies a probability distribution over the agents' joint observations and may be classified as:

Collective Partial Observability: No assumptions are made about the observability of the world state.

Collective Observability: Team's combined observations uniquely determines world state: $\forall \boldsymbol{\omega} \in \Omega_\alpha$, $\exists s \in S$ such that $\forall s' \neq s$, $\Pr({\Omega_\alpha}^t = \boldsymbol{\omega} | S^t = s') = 0$.

	Ind. Obs.	Coll. Obs.	Coll. Part. Obs.
No Comm.	P-Comp.	NEXP-Comp.	NEXP-Comp.
Gen. Comm.	P-Comp.	NEXP-Comp.	NEXP-Comp.
Free Comm.	P-Comp.	P-Comp.	PSPACE-Comp.

Table 1: Computational Complexity

Individual Observability: Each individual's observation uniquely determines the world state: $\forall \omega \in \Omega_i$, $\exists s \in S$ such that $\forall s' \neq s$, $\Pr(\Omega^t_i = \omega | S^t = s') = 0$.

Agent i chooses its actions and communication based on its belief state, $b^t_i \in B_i$, derived from the observations and communication it has received through time t. $B_\alpha = \prod_{i\in\alpha} B_i$ is the set of possible combined belief states. Like the Xuan-Lesser model(Xuan *et al* 2001), each decision epoch t consists of two phases. In the first phase, each agent i updates its belief state on receiving its observation, $\omega^t_i \in \Omega_i$, and chooses a message to send to its teammates. In the second phase, it updates its beliefs based on communication received, Σ^t_α, and then chooses its action. The agents use separate *state-estimator* functions to update their belief states: initial belief state, $b^0_i = SE^0_i()$; pre-communication belief state, $b^t_{i\bullet\Sigma} = SE_{i\bullet\Sigma}(b^{t-1}_{i\Sigma\bullet}, \omega^t_i)$; and post-communication belief state, $b^t_{i\Sigma\bullet} = SE_{i\Sigma\bullet}(b^t_{i\bullet\Sigma}, \Sigma^t_\alpha)$.

The COM-MTDP reward function represents the team's joint utility (shared by all members) over states and actions, $R{:}S{\times}\Sigma_\alpha{\times}A_\alpha{\rightarrow}\mathbb{R}$, and is the sum of two rewards: a domain-action-level reward, $R_A{:}S{\times}A_\alpha{\rightarrow}\mathbb{R}$, and a communication-level reward, $R_\Sigma{:}S{\times}\Sigma_\alpha{\rightarrow}\mathbb{R}$. COM-MTDP (and likewise R-COM-MTDP) domains can be classified based on the allowed communication and its reward:

General Communication: no assumptions on Σ_α nor R_Σ.

No Communication: $\Sigma_\alpha = \emptyset$.

Free Communication: $\forall \sigma \in \Sigma_\alpha$, $R_\Sigma(\sigma) = 0$.

Analyzing the extreme cases, like free communication (and others in this paper) helps to understand the computational impact of the extremes. Table 1 from (Pynadath & Tambe 2002) shows the worst case computational complexities for various classes of COM-MTDP domains. In the case of general communication and partially observable states, the decision problem of determining if there exists a joint policy with expected reward at least k is NEXP-Complete. The following sections illustrate algorithms that find a joint policy for this case.

Optimal Joint Policy

When agents do not fully communicate, they must coordinate instead by selecting policies that depend on their entire histories of observations. Thus, each agent, i, follows a deterministic policy of behavior, $\pi_i : \Omega^*_i \rightarrow A$, that maps a sequence of its individual observations into an action. We define a joint policy, π_α, for a team, α,

as a combination of such individual policies for each member of α. The problem facing the team is finding the optimal joint policy—i.e., the joint policy that maximizes the team's expected reward.

One sure-fire method for finding the optimal joint policy is to simply search the entire space of possible joint policies, evaluate the expected reward of each, and select the policy with the highest such value. To perform such a search, we must first be able to determine the expected reward of a joint policy. In computing this expectation, we must consider all of the branches, for both different world states and different observations, that occur at each time step. In general, for a team $\alpha = \{1, \ldots, n\}$ and for a horizon T, we can compute its expected reward when following a joint policy, $\boldsymbol{\pi}_\alpha$, by expanding all possible paths of execution. In the initial time step, we can express the value of the policy as follows:

$$\begin{aligned} V_{\boldsymbol{\pi}_\alpha} =& \sum_{s_0 \in S} P(\cdot, \cdot, s_0) \cdot \sum_{\omega_{10} \in \Omega_1} \cdots \sum_{\omega_{n0} \in \Omega_n} \boldsymbol{O}_\alpha(s_0, \cdot, \langle \omega_{10}, \ldots, \omega_{n0} \rangle) \\ & \cdot [R(s_0, \langle \pi_1(\langle \omega_{10} \rangle), \ldots, \pi_n(\langle \omega_{n0} \rangle) \rangle) \\ & + V^1_{\boldsymbol{\pi}_\alpha}(s_0, \langle \langle \omega_{10} \rangle, \ldots, \langle \omega_{n0} \rangle \rangle)] \end{aligned} \tag{1}$$

This expression represents the expected immediate reward earned at time 0 and the future value of executing policy, $\boldsymbol{\pi}_\alpha$, from time 1 onward. We represent the latter as $V^1_{\boldsymbol{\pi}_\alpha}$, which is a function of the current state of the world and the individual agents' observations. For all times $t \leq T$, we can express this function as:

$$\begin{aligned} & V^t_{\boldsymbol{\pi}_\alpha}(s_{t-1}, \langle \langle \omega_{10}, \ldots, \omega_{1(t-1)} \rangle, \ldots, \langle \omega_{n0}, \ldots, \omega_{n(t-1)} \rangle \rangle) \\ =& \sum_{s_t \in S} P\left(s_{t-1}, \langle \pi_1\left(\langle \omega_{10}, \ldots, \omega_{1(t-1)} \rangle\right), \ldots, \right. \\ & \left. \pi_n\left(\langle \omega_{n0}, \ldots, \omega_{n(t-1)} \rangle\right) \rangle, s_t\right) \\ & \cdot \sum_{\omega_{1t} \in \Omega_1} \cdots \sum_{\omega_{nt} \in \Omega_n} \boldsymbol{O}_\alpha\left(s_t, \langle \pi_1\left(\langle \omega_{10}, \ldots, \omega_{1(t-1)} \rangle\right), \ldots, \right. \\ & \left. \pi_n\left(\langle \omega_{n0}, \ldots, \omega_{n(t-1)} \rangle\right) \rangle, \langle \omega_{1t}, \ldots, \omega_{nt} \rangle\right) \\ & \cdot [R(s_t, \langle \pi_1(\langle \omega_{10}, \ldots, \omega_{1t} \rangle), \ldots, \pi_n(\langle \omega_{n0}, \ldots, \omega_{nt} \rangle) \rangle) \\ & + V^{t+1}_{\boldsymbol{\pi}_\alpha}(s_t, \langle \langle \omega_{10}, \ldots, \omega_{1t} \rangle, \ldots, \langle \omega_{n0}, \ldots, \omega_{nt} \rangle \rangle)] \end{aligned} \tag{2}$$

We terminate this recursion at the end of the finite horizon, where:

$$\begin{aligned} & V^{T+1}_{\boldsymbol{\pi}_\alpha}(s_T, \langle \langle \omega_{10}, \ldots, \omega_{1T} \rangle, \ldots, \langle \omega_{n0}, \ldots, \omega_{nT} \rangle \rangle) \\ =& 0 \end{aligned} \tag{3}$$

At each time step, the computation of $V^t_{\boldsymbol{\pi}_\alpha}$ performs a summation over all possible world states and agent observations, so the time complexity of this algorithm is $O\left(|S| \cdot |\boldsymbol{\Omega}_\alpha|^{T+1}\right)$. The overall search performs this computation for each and every possible joint policy. Since each policy specifies different actions over possible histories of observations, the number of possible policies for an individual agent i is $O\left(|A_i|^{\frac{|\Omega_i|^T - 1}{|\Omega_i| - 1}}\right)$. The number of possible *joint* policies is thus $O\left(\left(|A_*|^{\frac{|\Omega_*|^T - 1}{|\Omega_*| - 1}}\right)^{|\alpha|}\right)$, if we take $|A_*|$ and $|\Omega_*|$ to be the largest such state sizes over all the agent team members. The time complexity for finding the optimal joint policy by searching this space is thus:

$$O\left(\left(|A_*|^{\frac{|\Omega_*|^T - 1}{|\Omega_*| - 1}}\right)^{|\alpha|} \cdot \left(|S| \cdot |\boldsymbol{\Omega}_\alpha|\right)^T\right)$$

Thus, the time complexity of this exhaustive method for finding an optimal joint policy is exponential in the number of possible observations, $|\Omega_*|$, exponential in the number of agents, $|\alpha|$, and *doubly* exponential in the time horizon, T.

Approximate Algorithms

Given the complexity of exhaustively searching for the optimal joint policy, it is clear that such methods will not be successful when the amount to time to generate the policy is restricted. In this section, we will present any-time approximation algorithms that are guaranteed to find a locally optimal joint policy. We refer to this category of algorithms as "Joint ESP" (Joint Equilibrium-Based Search for Policies). The solution obtained is closely related to (i) PBP(Person-by-Person) in team theory; (ii) Nash equilibrium from cooperative and non-cooperative game theory. The key idea is to find the policy for one agent at a time, keeping the policies of all the other agents fixed, that maximizes the joint expected reward. This process is repeated until an equilibrium is reached (local optimum is found).

Exhaustive approach for Joint ESP

This algorithm below describes an exhaustive algorithm for Joint ESP. Here we consider that there are n cooperative agents. We modify the policy of one agent at a time keeping the policies of the other $n-1$ agents fixed. The function bestPolicy, returns the joint policy that maximizes the expected joint reward, obtained by keeping $n-1$ agents' policies fixed and exhaustively searching in the entire policy space of the agent whose policy is free. At each iteration, the value of the modified joint policy will either increase or remain unchanged (within some threshold ϵ of the previous joint policy's value). This is repeated until an equilibrium is reached, i.e., the policies of all n agents remains unchanged. This policy is guaranteed to be a local maximum since the value of the new joint policy at each iteration is non-decreasing.

ExhaustiveJointESP()
n: number of agents
ϵ: threshold for convergence
prev: previous joint policy, initially randomly set

convCount: count to indicate if convergence has taken place, initially 0
1. while convCount $\neq n-1$
2. for $i \leftarrow 1$ to n
3. fix policy of all agents except i
4. policySpace$\leftarrow$ list of all possible policies for i
5. new $\leftarrow$ bestPolicy(i,policySpace,prev)
6. prev $\leftarrow$ new
7. if new.value - prev.value $< \epsilon$
8. convCount$\leftarrow$convCount + 1
9. else
10. convCount$\leftarrow$0
11. if convCount $= n-1$
12. break
13. return new

This algorithm has the same worst case complexity as the exhaustive search for a globally optimal policy. However, this algorithm could end up in a local optimum. Techniques like *simulated annealing* can be applied to perturb the solution found to see if it settles on a different higher value. In the following section we describe a dynamic programming alternative to this exhaustive approach for doing Joint ESP.

Introducing Dynamic Programming for Finding an Optimal Policy

Basic Ideas In this section, we describe a dynamic-programming like method for finding an optimal policy for agent 1, assuming that agent 2's policy π_2 is determined. As discussed in previous sections, we cannot describe a policy of an agent as a function of its belief state. Therefore, we cannot use standard value iteration methods. However, we can still utilize a dynamic-programming like method since the *principle of optimality* holds. More specifically, assume we have a k-step optimal policy. If we take a part of this policy for last j steps, then this policy must be the optimal policy for remaining j steps, given the history of first $k-j$ steps. In this section, we show how to define a value function, which is defined on possible histories agent 1 might encounter.

Notations

- Assume we are designing the policy of agent 1, while the policy of agent 2, π_2 is fixed.
- S is a set of world states $\{1, 2, \ldots, m\}$
- A_1, A_2, are set of action for agent 1 and 2. a joint action is represented as (a_1, a_2).
- Ω_1, Ω_2 are set of observations for agent 1 and 2.
- Observation functions: O_1 and O_2. $O_i(s, (a_1, a_2), \omega_i)$ represents the probability of observing ω_i, if the current state is s and the previous joint action is (a_1, a_2). For notation simplicity, we define another observation function O_1' and O_2', where $O_i'(s, (a_1, a_2), \omega_i)$ represents the probability of observing ω_i, after performing joint action (a_1, a_2) at state s.
- A reward function: $R(s, (a_1, a_2))$ is the immediate reward for performing joint action (a_1, a_2) at state s.
- transition function: $P(s_i, (a_1, a_2), s_f)$ represents the probability of the current state is s_f, if the previous state is s_i and the previous joint action is (a_1, a_2).
- history: history h^t is a sequence of t pairs of an action and an observation, i.e., $h^t =< (a^1, w^1), (a^2, w^2), \ldots, (a^t, w^t) >$
- A state transition history $\mathbf{S^t}$, which is a sequence of t states $< S_1, S_2, \ldots, S_t >$.
- A belief state over state transition histories $\mathbf{BS^t}$. Let us denote the probability that $\mathbf{S^t}$ occurs as $B(\mathbf{S^t})$

Details of Dynamic Programming Method

- For a given history h^{t-1} and a state transition history $\mathbf{S^t}$, we can calculate the probability agent 2 has observed a sequence of observations o^{t-1}, $P(o^{t-1}|h^{t-1}, \mathbf{S^t})$.

  Also, given an observation history o^{t-1}, agent 2's t-th action is determined. We denote this as $a_2(o^{t-1})$.

$$P(o^{t-1}|h^{t-1}, \mathbf{S^t}) = \prod_{1 \leq i \leq t-1} O_2'(S_i, (a_1^i, a_2(o^{i-1})), \omega_i)$$

  where S_i is the i-th state in $\mathbf{S^t}$, a_1^i is the i-th action in h^{t-1}, and o^{i-1} is the first $i-1$ observations in o^{t-1}.

- For a given history h^{t-1} and initial synchronized belief state b^{init}, we can calculate the current belief state over state transition histories $\mathbf{BS^t}$.

  $B(\mathbf{S^t})$ can be incrementally calculated as follows.

$$B(\mathbf{S^1}) = b_{init}(S_1)$$

$$B(\mathbf{S^{i+1}}) = B(\mathbf{S^i}) \sum_{o^{i-1}} P(o^{i-1}|h^{i-1}, \mathbf{S^i}) \cdot O_1'(S_i, (a_1^i, a_2(o^{i-1})), \omega_1^i) \cdot P(S_i, (a_1^i, a_2(o^{i-1})), S_{i+1}) / \sum_{\mathbf{S^i}} B(\mathbf{S^i}) \sum_{o^{i-1}} P(o^{i-1}|h^{i-1}, \mathbf{S^i}) \cdot O_1'(S_i, (a_1^i, a_2(o^{i-1})), \omega_1^i)$$

  where a_1^i, ω_1^i is i-th action and observation in h^{t-1}.

- Assume we are designing a k step policy.
- We define a value function V_j, which is a function of a history h^{k-j}. $V_j(h^{k-j})$ represents the expected reward for taking an optimal policy for remaining j steps after history h^{k-j}.
- V_j is derived from $V_j^{a_1}$, i.e., $V_j(h^{k-j}) = \max_{a_1 \in A_1} V_j^{a_1}(h^{k-j})$.
- $V_1^{a_1}$ is defined as follows.

$$V_1^{a_1}(h^{k-1}) = ER(a_1|h^{k-1})$$

- $V_j^{a_1}$ is recursively defined as follows.

$$V_j^{a_1}(h^{k-j}) = ER(a_1|h^{k-j}) + \sum_{\omega_1 \in \Omega_1} P(\omega_1|h^{k-j}, a_1) \cdot V_{j-1}(< h^{k-j}, (a_1, \omega_1) >)$$

$$ER(a_1|h^{t-1}) = \sum_{\mathbf{S^t}} \sum_{o^{t-1}} B(\mathbf{S^t}) \cdot P(o^{t-1}|h^{t-1}, \mathbf{S^t}) \cdot R(S_t, (a_1, a_2(o^{t-1})))$$

$$P(\omega_1|h^{k-j}, a_1) = \sum_{\mathbf{S^{k-j+1}}} \sum_{o^{t-1}} B(\mathbf{S^{k-j+1}}) \cdot P(o^{k-j}|h^{k-j}, \mathbf{S^{k-j+1}}) \cdot O_1'(S_{k-j+1}, (a_1, a_2(o^{k-j})), \omega_1)$$

Experimental Results

In this section, we perform an empirical comparison of the algorithms described in sections 3 and 4.1 in terms of time and performance. We are currently working on an implementation of the dynamic programming approach. The following subsections describes the experimental setup and the results.

Experimental Description

We consider a multiagent version of the classic tiger problem that has been used in order to explain approaches to solving single agent POMDPs(Cassandra *et al* 1994). Our modified version of the problem is as follows:

Two agents are in a corridor facing two doors:"left" and "right". Behind one door lies a hungry tiger and behind the other lies untold riches but the agents don't know the position of either. The agents can jointly or individually open either door. If either of them opens the door behind which the tiger is present they are attacked by the tiger, however the injury sustained if the opened the door to the tiger is less severe if they open that door jointly. Similarly, they receive wealth when they open the door to the riches in proportion to the number of agents that opened that door. The agents can independently listen for the presence of the tiger at a small cost. If the tiger is behind the left door and the agent has listened, the agent received a . The agents cannot communicate with each other at all. Further we assume that every time either agent opens either one of the doors, the riches and the tiger are randomly repositioned and the agents have to solve the problem again. This problem can be generalized to n agents but we will only consider the 2 agent problem here.

Clearly, acting jointly is beneficial because the agents receive more riches and sustain less damage if they acted together. However, because the agents receive independent observations and cannot communicate, they need to consider the observation histories of the other agent and what action they are likely to perform. We consider two cases of the reward function, where we vary the penalty for jointly opening the door to the tiger. The exact problem specification is as described in Fig. 1. Fig. 1(a) describes the transition probabilities for each state transition given start state and joint action, Fig. 1(b) describes the observation probabilities for each joint observation given state and joint action last performed and Figs. 1(c) and 1(d) describe the two reward functions A and B that we use to evaluate the algorithms in section 3 and 4.1.

Evaluation Results

We ran the exhaustive globally optimal algorithm(sect 3) and the Exhaustive JESP algorithm(section 4.1) for the multiagent tiger problem using the specifications provided in Fig. 1. Finding the globally optimal policy is extremely slow for this problem and is exponential in the finite horizon, T. Hence, we evaluate the algorithms only for finite horizons of 1 and 2. Fig 2 shows the results of this evaluation. We ran the JESP algorithm for 5 different randomly selected initial policy settings. We compare the performance of the algorithms in terms of the number of policy evaluations that were necessary and in terms of the expected value of the policy.

Fig. 2(a), shows the results of running the globally optimal algorithm and the Exhaustive JESP algorithm using reward function A for finite horizon=1. As can be seen from this figure, the JESP algorithm requires much fewer evaluations to arrive at an equilibrium. Also the policy at equilibrium is the same as the globally optimal policy for this particular setting. The difference in the run times of the globally optimal algorithm and the JESP algorithm is more apparent when the finite horizon=2 (See Fig. 2(b)). Here the globally optimal algorithm performed 531441 policy evaluations while the JESP algorithm did fewer than 3000 evaluations. In this case, too, JESP succeeded in finding the globally optimal policy. However, this is not always the case. Fig. 2(c), shows the results of running the globally optimal algorithm and the Exhaustive JESP algorithm using reward function B for finite horizon=1. Here, the JESP algorithm often settles on a locally optimal policy that is different from the globally optimal policy.

Based on Fig 2, we can conclude that the exhaustive JESP algorithm performs better than an exhaustive search for the globally optimal policy but can some times settle on a policy that is only locally optimal. This could be sufficient for problems where the difference between the locally optimal policy's value and the globally optimal policy's value is small and it is imperative that a policy be found quickly. The JESP algorithm can be altered so that it doesn't get stuck in a local optimum using techniques like *simulated annealing*.

Summary and Conclusion

Decentralized POMDPs can be used as a decision theoretic model for many multiagent problems highlighting the importance of finding algorithms for solving decentralized POMDPs. Most of the work in the literature

Action/Transition	Tiger Left ← Tiger Left	Tiger Left ← Tiger Right	Tiger Right ← Tiger Right	Tiger Right ← Tiger Left
<Open Right,Open Right>	0.5	0.5	0.5	0.5
<Open Left,Open Left>	0.5	0.5	0.5	0.5
<Open Right,Open Left>	0.5	0.5	0.5	0.5
<Open Left,Open Right>	0.5	0.5	0.5	0.5
<Listen,Listen>	1.0	0.0	1.0	0.0
<Listen,Open Right>	0.5	0.5	0.5	0.5
<Open Right,Listen>	0.5	0.5	0.5	0.5
<Listen,Open Left>	0.5	0.5	0.5	0.5
<Open Left,Listen>	0.5	0.5	0.5	0.5

(a)

Action	**State**	<Tiger Left,Tiger Left>	<Tiger Left,Tiger Right>	<Tiger Right,Tiger Right>	<Tiger Right,Tiger Left>
<Listen,Listen>	Tiger Left	0.7225	0.1275	0.0225	0.1275
<Listen,Listen>	Tiger Right	0.0225	0.1275	0.7225	0.1275
<Open Right,Open Right>	*	0.25	0.25	0.25	0.25
<Open Left,Open Left>	*	0.25	0.25	0.25	0.25
<Open Right,Open Left>	*	0.25	0.25	0.25	0.25
<Open Left,Open Right>	*	0.25	0.25	0.25	0.25
<Listen,Open Right>	*	0.25	0.25	0.25	0.25
<Open Right,Listen>	*	0.25	0.25	0.25	0.25
<Listen,Open Left>	*	0.25	0.25	0.25	0.25
<Open Left,Listen>	*	0.25	0.25	0.25	0.25

(b)

Action/State	Tiger Left	Tiger Right
<Open Right,Open Right>	+20	-50
<Open Left,Open Left>	-50	+20
<Open Right,Open Left>	-100	-100
<Open Left,Open Right>	-100	-100
<Listen,Listen>	-2	-2
<Listen,Open Right>	+9	-101
<Open Right,Listen>	+9	-101
<Listen,Open Left>	-101	+9
<Open Left,Listen>	-101	+9

(c)

Action/State	Tiger Left	Tiger Right
<Open Right,Open Right>	+20	0
<Open Left,Open Left>	-50	+20
<Open Right,Open Left>	-100	-100
<Open Left,Open Right>	-100	-100
<Listen,Listen>	-2	-2
<Listen,Open Right>	+9	-101
<Open Right,Listen>	+9	-101
<Listen,Open Left>	-101	+9
<Open Left,Listen>	-101	+9

(d)

Figure 1: (a) Transition function; (b) Observation function; (c) Reward function A;(d)Reward function B

Joint Policy	Number of evaluations	Value
Globally Optimal Policy	81	-2
JESP Policy (at start, Agent1←Policy1,Agent2←Policy7)	27 (3 iterations)	-2
JESP Policy (at start, Agent1←Policy4,Agent2←Policy8)	18 (2 iterations)	-2
JESP Policy (at start, Agent1←Policy4,Agent2←Policy7)	27 (3 iterations)	-2
JESP Policy (at start, Agent1←Policy6,Agent2←Policy8)	18 (2 iterations)	-2
JESP Policy (at start, Agent1←Policy8,Agent2←Policy6)	27 (3 iterations)	-2

(a)

Joint Policy	Number of evaluations	Value
Globally Optimal Policy	531441	-4
JESP Policy (at start, Agent1←Policy275,Agent2←Policy713)	1458 (2 iterations)	-4
JESP Policy (at start, Agent1←Policy140,Agent2←Policy522)	2187 (3 iterations)	-4
JESP Policy (at start, Agent1←Policy245,Agent2←Policy523)	2187 (3 iterations)	-4
JESP Policy (at start, Agent1←Policy678,Agent2←Policy310)	2187 (3 iterations)	-4
JESP Policy (at start, Agent1←Policy421,Agent2←Policy720)	1458 (2 iterations)	-4

(b)

Joint Policy	Number of evaluations	Value
Globally Optimal Policy	81	10
JESP Policy (at start, Agent1←Policy7,Agent2←Policy4)	18 (2 iterations)	10
JESP Policy (at start, Agent1←Policy0,Agent2←Policy3)	27 (3 iterations)	-2
JESP Policy (at start, Agent1←Policy1,Agent2←Policy5)	27 (3 iterations)	-2
JESP Policy (at start, Agent1←Policy6,Agent2←Policy8)	18 (2 iterations)	-2
JESP Policy (at start, Agent1←Policy5,Agent2←Policy6)	27 (3 iterations)	-2

(c)

Figure 2: Evaluation Results for (a) Reward A, Horizon = 1; (b) Reward A, Horizon = 2; (c) Reward B, Horizon = 1

is restricted to the single agent POMDP case and does not directly apply because the decentralized problem is much *harder*. Most algorithms for solving decentralized POMDPs make some simplifying assumptions that restrict their applicability and do not explicitly model communication amongst the agents. To overcome these shortcomings, we present a formal model, COM-MTDP (Pynadath & Tambe 2002), that can explicitly reason about communication (when and what to communicate).

We then describe a exhaustive search algorithm for a globally optimal solution and analyze the complexity of this algorithm, which we find to be doubly exponential in the number of agents and time, highlighting the importance of more feasible approximations. We then describe a class of algorithms which we refer to as "Joint Equilibrium-based Search for Policies" that will find a local optimum and describe an exhaustive algorithm and a dynamic programming algorithm for JESP. We empirically compared the exhaustive JESP algorithm with the globally optimal exhaustive algorithm for a 2 agent "tiger problem". For this problem it was found that the JESP algorithm was much faster than the global search. *Simulated Annealing* like techniques can be used to prevent the exhaustive JESP algorithm from getting stuck in a local optimum.

References

Bernstein, D. S.; Zilberstein, S.; and Immerman, N. 2000. The complexity of decentralized control of MDPs. In *UAI-00*.

Cassandra, A. R.; Kaelbling, L. P.; and Littman, M. L. 1994. Acting optimally in partially observable stochastic domains. In *AAAI-1994*.

Chadès, I.; Scherrer, B.; and Charpillet, F. 1994. A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem. In *SAC-2002*.

Kitano, H. *et al.* 1999. Robocup-rescue: Search and rescue for large scale disasters as a domain for multiagent research. In *IEEE Conference SMC-99*.

Nair, R.; Tambe, M.; and Marsella, S. 2002. Team formation for reformation in multiagent domains like RoboCupRescue. In *RoboCup Symposium*.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. Complexity of markov decision processes. *Mathematics of Operatios Research* 12(3):441–450.

Peshkin, L.; Meuleau, N.; Kim, K.-E.; and Kaelbling, L. 2000. Learning to cooperate via policy search. In *UAI-2000*.

Pynadath, D., and Tambe, M. 2002. Multiagent teamwork: Analyzing the optimality complexity of key theories and models. In *AAMAS-02*.

Xuan, P.; Lesser, V.; and Zilberstein, S. 2001. Communication decisions in multiagent cooperation. In *Agents-01*.