DECENTRALIZED COMMUNICATION STRATEGIES FOR COORDINATED MULTI-AGENT POLICIES

Maayan Roth, Reid Simmons, Manuela Veloso
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
mroth@andrew.cmu.edu, reids@cs.cmu.edu, veloso@cs.cmu.edu

Abstract Although the presence of free communication reduces the complexity of multi-agent POMDPs to that of single-agent POMDPs, in practice, communication is not free and reducing the amount of communication is often desirable. We present a novel approach for using centralized "single-agent" policies in decentralized multi-agent systems by maintaining and reasoning over the possible *joint beliefs* of the team. We describe how communication is used to integrate local observations into the team belief as needed to improve performance. We show both experimentally and through a detailed example how our approach reduces communication while improving the performance of distributed execution.

Keywords: Communication, distributed execution, decentralized POMDP

1. Introduction

Multi-agent systems and multi-robot teams can be used to perform tasks that could not be accomplished by, or would be very difficult with, single agents. Such teams provide additional functionality and robustness over single-agent systems, but also create additional challenges. In any physical system, robots must reason over, and act under, uncertainty about the state of the environment. However, in many multi-agent systems there is additional uncertainty about the collective state of the team. If the agents can maintain sufficient collective belief about the state of the world, they can coordinate their joint actions to achieve high reward. Conversely, uncoordinated actions may be costly.

Just as Partially Observable Markov Decision Problems (POMDPs) are used to reason about uncertainty in single-agent systems, there has been recent interest in using multi-agent POMDPs for coordination of teams of agents [15]. Unfortunately, multi-agent POMDPs are known to be highly intractable [2].

Communicating (at zero cost) at every time step reduces the computational complexity of a multi-agent POMDP to that of a single agent [13]. Although single-agent POMDPs are also computationally challenging, a significant body of research exists that addresses the problem of efficiently finding near-optimal POMDP policies [6]. However, communication is generally not free, and forcing agents to communicate at every time step wastes a limited resource.

In this paper, we introduce an approach that exploits the computational complexity benefits of free communication at policy-generation time, while at run-time maintains agent coordination and chooses to communicate only when there is a perceived benefit to team performance. Section 2 of this paper gives an overview of the multi-agent POMDP framework and discusses related work. Sections 3 and 5 introduce our algorithm for reducing the use of communication resources while maintaining team coordination. Section 4 illustrates this algorithm in detail with an example and Section 6 presents experimental results that demonstrate the effectiveness of our approach at acting in coordination while reducing communication.

2. Background and related work

There are several equivalent multi-agent POMDP formulations (i.e. DEC-POMDP [2], MTDP [13], POIPSG [11]). In general, a multi-agent POMDP is an extension of a single-agent POMDP where α agents take individual actions and receive local observations, but accumulate a joint team reward. The multi-agent POMDP model consists of the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, O, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of n world states, $\{s^1 \dots s^n\}$ and $\mathcal{A}$ is the set of m joint actions available to the team, where each joint action, a^i, is comprised of α individual actions $\langle a^i_1 \dots a^i_\alpha \rangle$. Agents are assumed to take actions simultaneously in each time step. The transition function, $\mathcal{T}$, depends on joint actions and gives the probability associated with starting in a particular state s^i and ending in a state s^j after the team has executed the joint action a^k. Although the agents cannot directly observe their current state, s^t, they receive information about the state of the world through Ω, a set of possible joint observations. Each joint observation ω^i is comprised of α individual observations, $\langle \omega^i_1 \dots \omega^i_\alpha \rangle$. The observation function, O, gives the probability of observing a joint observation ω^i after taking action a^k and ending in state s^j. The reward function $\mathcal{R}$ maps a start state and a joint action to a reward. This reward is obtained jointly by all of the agents on the team, and is discounted by the discount factor γ.

Without communication, solving for the optimal policy of a multi-agent POMDP is known to be NEXP-complete [2], making these problems fundamentally harder than single-agent POMDPs, which are known to be PSPACE-complete [10]. A recent approach presents a dynamic programming algorithm for finding optimal policies for these problems [5]. In some domains, dynamic

programming may provide a substantial speed-up over brute-force searches, but in general, this method remains computationally intractable. Recent work focuses on finding heuristic solutions that may speed up the computation of locally optimal multi-agent POMDP policies, but these algorithms either place limitations on the types of policies that can be discovered (e.g. limited-memory finite state controllers [11]), or make strong limiting assumptions about the types of domains that can be solved (e.g. transition-independent systems [1]), or may, in the worst case, still have the same complexity as an exhaustive search for the optimal policy [8]. Another method addresses the problem by approximating the system (in this case, represented as a POIPSG) with a series of smaller Bayesian games [4]. This approximation is able to find locally optimal solutions to larger problems than can be solved using exhaustive methods, but is unable to address situations in which a guarantee of strict agent coordination is needed. Additionally, none of these approaches address the issue of communication as a means for improving joint team reward.

Although free communication transforms a multi-agent POMDP into a large single agent POMDP, in the general case where communication is not free, adding communication does not reduce the overall complexity of optimal policy generation for a multi-agent POMDP [13]. Unfortunately, for most systems, communication is not free, and communicating at every time step may be unnecessary and costly. However, it has been shown empirically that adding communication to a multi-agent POMDP may not only improve team performance, but may also shorten the time needed to generate policies [9].

In this paper, we introduce an algorithm that takes as input a single-agent POMDP policy, computed as if for a team with free communication, and at run-time, maintains team coordination and chooses to communicate only when it is necessary for improving team performance. This algorithm makes two trade-offs. First, it trades off the need to perform computations at run-time in order to enable the generation of an infinite-horizon policy for the team that would otherwise be highly intractable to compute. Secondly, it conserves communication resources, with the potential trade-off of some amount of reward.

3. Dec-Comm algorithm

Single-agent POMDP policies are mappings from beliefs to actions ($\pi : \mathcal{B} \rightarrow \mathcal{A}$), where a belief, $b \in \mathcal{B}$, is a probability distribution over world states. An individual agent in a multi-agent system cannot calculate this belief because it sees only its own local observations. Even if an agent wished to calculate a belief based only on its own observations, it could not, because the transition and observation functions depend on knowing the joint action of the team.

A multi-agent POMDP can be transformed into a single-agent POMDP by communicating at every time step. A standard POMDP solver can then be used

to generate a policy that operates over joint observations and returns joint actions, ignoring the fact that these joint observations and actions are comprised of individual observations and actions. The belief over which this policy operates, which is calculated identically by each member of the team, is henceforth referred to as the *joint belief*.

Creating and executing a policy over joint beliefs is equivalent to creating a centralized controller for the team and requires agents to communicate their observations at each time step. We wish to reduce the use of communication resources. Therefore, we introduce the DEC-COMM algorithm that:

- in a decentralized fashion, selects actions based on the possible joint beliefs of the team
- chooses to communicate when an agent's local observations indicate that sharing information would lead to an increase in expected reward

3.1 Q-POMDP: Reasoning over possible joint beliefs

The Q-MDP method is an approach for finding an approximate solution to a large single-agent POMDP by using the value functions ($\mathcal{V}_a(s)$ is the value of taking action a in state s and henceforth acting optimally) that are easily obtainable for the system's underlying MDP [7]. In Q-MDP, the best action for a particular belief, b, is chosen according to $\text{Q-MDP}(b) = \arg\max_a \sum_{s \in S} b(s) \times \mathcal{V}_a(s)$, which averages the values of taking each action in every state, weighted by the likelihood of being in that state as estimated by the belief.

Analogously, we introduce the Q-POMDP method for approximating the best joint action for a multi-agent POMDP by reasoning over the values of the possible joint beliefs in the underlying centralized POMDP. In our approach, a joint policy is created for the system, as described above. During execution, each agent calculates a tree of possible joint beliefs of the team. These joint beliefs represent all of the possible observation histories that could have been observed by the team. We define $\mathcal{L}^t$, the set of leaves of the tree at depth t, to be the set of possible joint beliefs of the team at time t. Each $\mathcal{L}_i^t$ is a tuple consisting of $\langle b^t, p^t, \vec{\omega}^t \rangle$, where $\vec{\omega}^t$ is the joint observation history that would lead to $\mathcal{L}_i^t$, b^t is the joint belief at that observation history, and p^t is the probability of the team observing that history.

Table 1 presents the algorithm for expanding a single leaf in a tree of possible joint beliefs. Each leaf has a child leaf for every possible joint observation. For each observation, $Pr(\omega^i | a, b^t)$, the probability of receiving that observation while in belief state b^t and having taken action a, is calculated. The resulting belief, b^{t+1}, is calculated using a standard Bayesian update [6]. The child leaf is composed of this new belief, b^{t+1}, the probability of reaching that belief, which is equivalent to the probability of receiving this particular observation

in the parent leaf times the probability of reaching the parent leaf, and the corresponding observation history. Note that this algorithm entirely ignores the actual observations seen by each agent, enabling the agents to compute identical trees in a decentralized fashion.

Table 1. Algorithm to grow the children of one leaf in a tree of possible beliefs

GROWTREE($\mathcal{L}_i^t$, a)
  $\mathcal{L}^{t+1} \leftarrow \emptyset$
  for each $\omega^j \in \Omega$
    $b^{t+1} \leftarrow \emptyset$
    $Pr(\omega^j|a,b^t) \leftarrow \sum_{s' \in \mathcal{S}} O(s',a,\omega^j) \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') b^t(s)$
    for each s' $\in \mathcal{S}$
      $b^{t+1}(s') \leftarrow \frac{O(s',a,\omega^j \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') b^t(s)}{Pr(\omega^j|a,b^t)}$
    $p^{t+1} \leftarrow p(\mathcal{L}_i^t) \times Pr(\omega^j|a,b^t)$
    $\vec{\omega}^{t+1} \leftarrow \vec{\omega}(\mathcal{L}_i^t) \circ \langle \omega^j \rangle$
    $\mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup [b^{t+1}, p^{t+1}, \vec{\omega}^{t+1}]$
  return $\mathcal{L}^{t+1}$

The Q-POMDP heuristic, Q-POMDP$(\mathcal{L}^t) = \arg\max_a \sum_{\mathcal{L}_i^t \in \mathcal{L}^t} p(\mathcal{L}_i^t) \times Q(b(\mathcal{L}_i), a)$, selects a single action that maximizes expected reward over all of the possible joint beliefs. Because this reward is a weighted average over several beliefs, there may exist domains for which an action that is strictly dominated in any single belief, and therefore does not appear in the policy, may be the optimal action when there is uncertainty about the belief. We define the Q function, $Q(b^t,a) = \sum_{s \in \mathcal{S}} \mathcal{R}(s,a) b^t(s) + \gamma \sum_{\omega \in \Omega} Pr(\omega|a,b^t) \mathcal{V}^{\pi}(b^{t+1})$, in order to take these actions into account. The value function, $\mathcal{V}^{\pi}(b)$, gives the maximum attainable value at the belief b, but is only defined over those actions which appear in the single-agent policy π. The Q function returns expected reward for any action and belief. b^{t+1} is the belief that results from taking action a in belief state b^t and receiving the joint observation ω. $Pr(\omega|a,b^t)$ and b^{t+1} are calculated as in Table 1.

Since all of the agents on a team generate identical trees of possible joint beliefs, and because Q-POMDP selects actions based only on this tree, ignoring the actual local observations of the agents, agents are guaranteed to select the same joint action at each time step. However, this joint action is clearly very conservative, as agents are forced to take into account all possible contingencies. The DEC-COMM algorithm utilizes communication to allow agents to integrate their actual observations into the possible joint beliefs, while still maintaining team synchronization.

3.2 Dec-Comm: Using communication to improve performance

An agent using the DEC-COMM algorithm chooses to communicate when it sees that integrating its own observation history into the joint belief would cause a change in the joint action that would be selected. To decide whether or not to communicate, the agent computes a_{NC}, the joint action selected by the Q-POMDP heuristic based on its current tree of possible joint beliefs. It then prunes the tree by removing all beliefs that are inconsistent with its own observation history and computes a_C, the action selected by Q-POMDP based on this pruned tree. If the actions are the same, the agent chooses not to communicate. If the actions are different, this indicates that there is a potential gain in expected reward through communication, and the agent broadcasts its observation history to its teammates. When an agent receives a communication from one of its teammates, it prunes its tree of joint beliefs to be consistent with the observations communicated to it, and recurses to see if this new information would lead it to choose to communicate. Because there may be multiple instances of communication in each time step, agents must wait a fixed period of time for the system to quiesce before acting. Table 2 provides the details of the DEC-COMM algorithm.

Table 2. One time step of the DEC-COMM algorithm for an agent j

DEC-COMM($\mathcal{L}^t, \vec{\omega}_j^t$)
- $a_{NC} \leftarrow$ Q-POMDP($\mathcal{L}^t$)
- $\mathcal{L}' \leftarrow$ prune leafs inconsistent with $\vec{\omega}_j^t$ from $\mathcal{L}^t$
- $a_C \leftarrow$ Q-POMDP($\mathcal{L}'$)
- **if** $a_{NC} \neq a_C$
  - communicate $\vec{\omega}_j^t$ to the other agents
  - **return** DEC-COMM($\mathcal{L}', \emptyset$)
- **else**
  - **if** communication $\vec{\omega}_k^t$ was received from another agent k
    - $\mathcal{L}^t \leftarrow$ prune leafs inconsistent with $\vec{\omega}_k^t$ from $\mathcal{L}^t$
    - **return** DEC-COMM($\mathcal{L}^t, \vec{\omega}_j^t$)
  - **else**
    - take action a_{NC}
    - receive observation ω_j^{t+1}
    - $\vec{\omega}_j^{t+1} \leftarrow \vec{\omega}_j^t \circ \langle \omega_j^{t+1} \rangle$
    - $\mathcal{L}^{t+1} \leftarrow \emptyset$
    - **for** each $\mathcal{L}_i^t \in \mathcal{L}^t$
      - $\mathcal{L}^{t+1} \leftarrow \mathcal{L}^{t+1} \cup$ GROWTREE($\mathcal{L}_i^t, a_{NC}$)
    - **return** $[\mathcal{L}^{t+1}, \vec{\omega}_j^{t+1}]$

4. Example

To illustrate the details of our algorithm, we present an example in the two-agent tiger domain introduced by Nair *et al.* [8]. We use the tiger domain because it is easily understood, and also because it is a problem that requires coordinated behavior between the agents. The tiger problem consists of two doors, LEFT and RIGHT. Behind one door is a tiger, and behind the other is a treasure. $\mathcal{S}$ consists of two states, SL and SR, indicating respectively that the tiger is behind the left door or the right door. The agents start out with a uniform distribution over these states (b(SR) = 0.5).

Each agent has three individual actions available to it: OPENL, which opens the left door, OPENR, which opens the right door, and LISTEN, an information-gathering action that provides an observation about the location of the tiger. Together, the team may perform any combination of these individual actions. A joint action of ⟨LISTEN, LISTEN⟩ keeps the world in its current state. In order to make this an infinite-horizon problem, if either agent opens a door, the world is randomly and uniformly reset to a new state. The agents receive two observations, HL and HR, corresponding to hearing the tiger behind the left or right door. For the purposes of our example, we modify the observation function from the one given in Nair *et al.* If a door is opened, the observation is uniformly chosen and provides no information; the probability of an individual agent hearing the correct observation if both agents LISTEN is 0.7. (Observations are independent, so the joint observation function can be computed as the cross-product of the individual observation functions.) This change makes it such that the optimal policy is to hear two consistent observations (e.g. HR, HR) before opening a door.

The reward function for this problem is structured to create an explicit coordination problem between the agents. The highest reward (+20) is achieved when both agents open the same door, and that door does not contain the tiger. A lower reward (-50) is received when both agents open the incorrect door. The worst case is when the agents open opposite doors (-100), or when one agent opens the incorrect door while the other agent listens (-101). The cost of ⟨LISTEN, LISTEN⟩ is -2. We generated a joint policy for this problem with Cassandra's POMDP solver [3], using a discount factor of $\gamma = 0.9$. Note that although there are nine possible joint actions, all actions other than ⟨OPENL, OPENL⟩, ⟨OPENR, OPENR⟩, and ⟨LISTEN, LISTEN⟩ are strictly dominated, and we do not need to consider them.

Time Step 0: In this example, the agents start out with a synchronized joint belief of b(SR) = 0.5. According to the policy, the optimal joint action at this belief is ⟨LISTEN, LISTEN⟩. Because their observation histories are empty, there is no need for the agents to communicate.

Time Step 1: The agents execute ⟨LISTEN, LISTEN⟩, and both agents observe HL. Each agent independently executes GROWTREE. Figure 1 shows the tree of possible joint beliefs calculated by each agent. The Q-POMDP heuristic, executed over this tree, determines that the best possible joint action is ⟨LISTEN, LISTEN⟩.

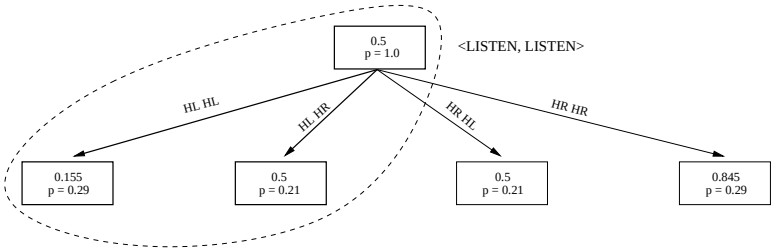

Figure 1. Joint beliefs after a single action

When deciding whether or not to communicate, agent 1 prunes all of the joint beliefs that are not consistent with its having heard HL. The circled nodes in Figure 1 indicate those nodes which are not pruned. Running Q-POMDP on the pruned tree shows that the best joint action is still ⟨LISTEN, LISTEN⟩, so agent 1 decides not to communicate. It is important to note that at this point, a centralized controller would have observed two consistent observations of HL and would perform ⟨OPENR, OPENR⟩. This is an instance in which our algorithm, because it does not yet have sufficient reason to believe that there will be a gain in reward through communication, performs worse than a centralized controller.

Time Step 2: After performing another ⟨LISTEN, LISTEN⟩ action, each agent again observes HL. Figure 2 shows the output of GROWTREE after the second action. The Q-POMDP heuristic again indicates that the best joint action is ⟨LISTEN, LISTEN⟩.

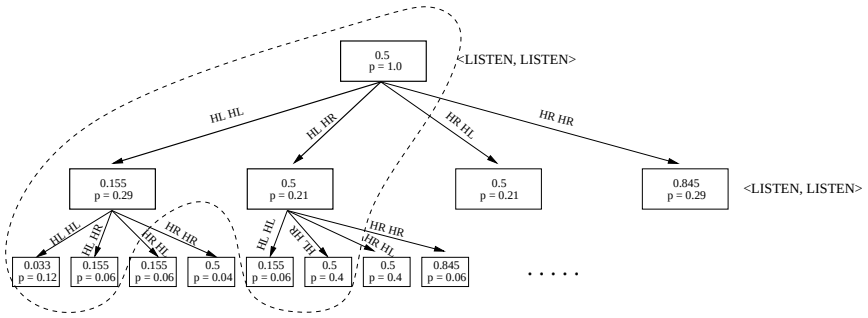

Figure 2. Joint beliefs after the second action

Agent 1 reasons about its communication decision by pruning all of the joint beliefs that are not consistent with its entire observation history (hearing HL twice). This leaves only the nodes that are circled in Figure 2. For the pruned tree, Q-POMDP indicates that the best action is ⟨OPENR, OPENR⟩. Because the pre-communication action, a_{NC}, differs from the action that would be chosen post-communication, a_C, agent 1 chooses to communicate its observation history to its teammate.

In the meantime, agent 2 has been performing an identical computation (since it too observed two instances of HL) and also decides to communicate. After both agents communicate, there is only a single possible belief remaining, b(SR) = 0.033. The optimal action for this belief is ⟨OPENR, OPENR⟩, which is now performed by the agents.

5. Particle filter representation

The above example shows a situation in which both agents decide to communicate their observation histories. It is easy to construct situations in which one agent would choose to communicate but the other agent would not, or examples in which both agents would decide not to communicate, possibly for many time steps (e.g. the agents observe alternating instances of HL and HR). From the figures, it is clear that the tree of possible joint beliefs grows rapidly when communication is not chosen. To address cases where the agents do not communicate for a long period of time, we present a method for modeling the distribution of possible joint beliefs using a particle filter.

A particle filter is a sample-based representation that can be used to encode an arbitrary probability distribution using a fixed amount of memory. In the past, particle filters have been used with single-agent POMDPs (i.e. for state estimation during execution [12]). We draw our inspiration from an approach that finds a policy for a continuous state-space POMDP by maximizing over a distribution of possible belief states, represented by a particle filter [14].

In our approach, each particle, $\mathcal{L}^i$ is a tuple of α observation histories, $\langle \vec{\omega}_a \ldots \vec{\omega}_\alpha \rangle$, corresponding to a possible observation history for each agent. Taken together, these form a possible joint observation history, and along with the system's starting belief state, b^0, and the history of joint actions taken by the team, $\vec{a}$, uniquely identify a possible joint belief. Every agent stores two particle filters, $\mathcal{L}_{joint}$, which represents the joint possible beliefs of the team, pruned only by communication, and $\mathcal{L}_{own}$, those beliefs that are consistent with the agent's own observation history. Belief propagation is performed for these filters as described in [14], with the possible next observations for $\mathcal{L}_{joint}$ taken from all possible joint observations, and the possible next observations for $\mathcal{L}_{own}$ taken only from those joint observations consistent with the agent's own local observation at that time step.

The DEC-COMM algorithm proceeds as described in Section 3, with $\mathcal{L}_{joint}$ used to generate a_{NC} and $\mathcal{L}_{own}$ used to generate a_c. The only complication arises when it comes time to prune the particle filters as a result of communication. Unlike the tree described earlier that represents the distribution of possible joint beliefs exactly, a particle filter only approximates the distribution. Simply removing those particles not consistent with the communicated observation history and resampling (to keep the total number of particles constant) may result in a significant loss of information about the possible observation histories of agents that have not yet communicated.

Looking at the example presented in Section 4, it is easy to see that there is a correlation between the observation histories of the different agents. (i.e. If one agent observes $\langle \mathrm{HL}, \mathrm{HL} \rangle$, it is unlikely that the other agent will have observed $\langle \mathrm{HR}, \mathrm{HR} \rangle$.) To capture this correlation when pruning, we define a similarity metric between two observation histories, Table 3. When an observation history $\vec{\omega}_i^t$ has been communicated by agent i, to resample the new $\mathcal{L}_{joint}$, the observation history in each particle corresponding to agent i is compared to $\vec{\omega}_i^t$. The comparison asks the question, "Suppose an agent has observed $\vec{\omega}_i^t$ after starting in belief b^0 and knowing that the team has taken the joint action history $\vec{a}^t$. What is the likelihood that an identical agent would have observed the observation history $\vec{\omega}_j^t$?" The value returned by this comparison is used as a weight for the particle. The particles are then resampled according to the calculated weights, and the agent i observation history for each particle is replaced with $\vec{\omega}_i^t$.

Table 3. The heuristic used to determine the similarity between two observation histories, where $\vec{\omega}_i^t$ is the true (observed) history

SIMILARITY$(\vec{\omega}_i^t, \vec{\omega}_j^t, \vec{a}^t)$
- $sim \leftarrow 1$
- $b \leftarrow b^0$
- **for** $t' = 1 \ldots t$
  - **for** each $s \in \mathcal{S}$
    - $b(s) \leftarrow O(s, a^{t'}, \omega_i^{t'}) b(s)$
  - normalize b
  - $sim \leftarrow sim \times \sum_{s \in \mathcal{S}} O(s, a^{t'}, \omega_j^{t'}) b(s)$
  - **for** each $s \in \mathcal{S}$
    - $b(s) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{T}(s', a^{t'}, s) b(s)$
  - normalize b
- **return** sim

6. Results and analysis

We demonstrate the performance of our approach experimentally by comparing the reward achieved by a team that communicates at every time step

(i.e. a centralized controller) to a team that uses the DEC-COMM algorithm to select actions and make communication decisions. We ran our experiment on the two-agent tiger domain as described in Section 4. In each experiment, the world state was initialized randomly, and the agents were allowed to act for 8 time steps. The team using a particle representation used 2000 samples to represent the possible beliefs. We ran 30000 trials of this experiment. Table 4 summarizes the results of these trials.

Table 4. Experimental results

	μ_{Reward}	σ_{Reward}	μ_{Comm}	σ_{Comm}
Full Comm.	17.0	37.9	16	0
DEC-COMM (tree)	8.9	28.9	2.9	1.1
DEC-COMM (particles)	9.4	30.3	2.6	1.0

It may appear at first glance as though the performance of the DEC-COMM algorithm is substantially worse than the centralized controller. However, as the high standard deviations indicate, the performance of even the centralized controller varies widely, and DEC-COMM under-performs the fully communicating system by far less than one standard deviation. Additionally, it achieves this performance by using less than a fifth as much communication as the fully communicating system. Note that the particle representation performs comparably to the tree representation (within the error margins), indicating that with a sufficient number of particles, there is no substantial loss of information.

We are currently working on comparing the performance of our approach to COMMUNICATIVE JESP, a recent approach that also uses communication to improve the computational tractability and performance of multi-agent POMDPs [9]. However, this comparison is difficult for several reasons. First of all, the COMMUNICATIVE JESP approach treats communication as domain-level action in the policy. Thus, if an agent chooses to communicate in a particular time step, it cannot take an action. More significantly, their approach deals only with synchronized communications, meaning that if one agent on a team chooses to communicate, it also forces all its other teammates to communicate at that time step.

7. Conclusion

We present in this paper an approach that enables the application of centralized POMDP policies to distributed multi-agent systems. We introduce the novel concept of maintaining a tree of possible joint beliefs of the team, and describe a heuristic, Q-POMDP, that allows agents to select the best action over the possible beliefs in a decentralized fashion. We show both through a detailed example and experimentally that our DEC-COMM algorithm makes communication decisions that improve team performance while reducing the

instances of communication. We also provide a fixed-size method for maintaining a distribution over possible joint team beliefs.

In the future, we are interested in looking at factored representations that may reveal structural relationships between state variables, allowing us to address the question of *what* to communicate, as well was *when* to communicate. Other areas for future work include reasoning about communicating only *part* of the observation history, and exploring the possibility of agents *asking* their teammates for information instead of only *telling* what they know.

References

[1] R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. Transition-Independent Decentralized Markov Decision Processes. *International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2003.

[2] D.S. Bernstein, S. Zilberstein, and N. Immerman. The Complexity of Decentralized Control of Markov Decision Processes. *Uncertainty in Artificial Intelligence*, 2000.

[3] A.R. Cassandra. POMDP solver software. http://www.cassandra.org/pomdp/code/index.shtml

[4] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate Solutions for Partially Observable Stochastic Games with Common Payoffs. *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2004.

[5] E.A. Hansen, D.S. Bernstein, and S. Zilberstein. Dynamic Programming for Partially Observable Stochastic Games. *National Conference on Artificial Intelligence*, 2004.

[6] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and Acting in Partially Observable Domains. *Artificial Intelligence*, 1998.

[7] M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Learning policies for partially observable environments: Scaling up. *International Conference on Machine Learning*, 1995.

[8] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings. *International Joint Conference on Artificial Intelligence*, 2003.

[9] R. Nair, M. Roth, M. Yokoo, and M. Tambe. Communication for Improving Policy Computation in Distributed POMDPs. *International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2004.

[10] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov Decision Processes. *Mathematics of Operations Research*, 1987

[11] L. Peshkin, K.-E. Kim, N. Meuleau, and L.P. Kaelbling. Learning to Cooperate via Policy Search. *Uncertainty in Artificial Intelligence*, 2000.

[12] P. Poupart, L.E. Ortiz, and C. Boutilier. Value-directed Sampling Methods for Monitoring POMDPs. *Uncertainty in Artificial Intelligence*, 2001.

[13] D.V. Pynadath and M. Tambe. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. *Journal of AI Research*, 2002.

[14] S. Thrun. Monte Carlo POMDPs. *Neural Information Processing Systems*, 2000.

[15] P. Xuan and V. Lesser. Multi-agent Policies: From Centralized Ones to Decentralized Ones. *International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2002.