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 frequency 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 collection of possible *joint beliefs* of the team. We describe how communication can be used to integrate local observations into the team belief as needed to improve performance, and show both experimentally and through a detailed example how our approach minimizes communication while improving the performance of distributed execution.

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 prove to be costly.

Just as Partially Observable Markov Decision Problems (POMDPs) have been used to reason about uncertainty in single-agent systems, there has been recent interest in using multi-agent POMDPs to address the issues of acting in coordination with a team in an uncertain environment [1] [2]. Unfortunately, multi-agent POMDPs are known to be highly intractable. The presence of free communication reduces the computational complexity of a multi-agent POMDP to that of a single agent. 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 [3]. However, communication is generally not free, and forcing agents to communicate at every time step is wasteful of a potentially 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. Section 3 introduces our algorithm for minimizing the use of communication resources, while maintaining team coordination. Section 4 illustrates this algorithm in detail with an example in the tiger domain, and Section 5 presents experimental results.

2 Background and related work

There are several equivalent multi-agent POMDP formulations (i.e. DEC-POMDP [1], MTDP [4], POIPSG [5]). 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 $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, \mathcal{O}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of *n* world states, $\{\mathcal{S}_1 \ldots \mathcal{S}_n\}$. $\mathcal{A}$ is the set of *m* joint actions available to the team, where each joint action, a^i, is comprised of α individual actions taken by each teammate. 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 $\mathcal{S}_i$ and ending in a state $\mathcal{S}_j$ after the team has executed the joint action a_k. Although the agents cannot directly observe their current state, $\mathcal{S}^t$, 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, $\langle \omega_1^i \ldots \omega_\alpha^i \rangle$. The observation function, $\mathcal{O}$, gives the probability of observing a joint observation ω^i after taking action $\mathcal{A}_k$ and ending in state $\mathcal{S}_j$. 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.

In the absence of communication, solving for the optimal policy of a multi-agent POMDP is known to be NEXP-complete, making these problems fundamentally harder than single-agent POMDPs, which are known to be PSPACE-complete [1] [6]. Some recent work has been done that focuses on finding heuristic solutions that may speed up the computation of locally optimal multi-agent POMDP policies (i.e. [7] [5]), but these algorithms either place limitations on the types of policies that can be discovered (e.g. limited-memory finite state controllers) or may, in the worst case, still have the same complexity as an exhaustive search for the optimal policy.

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 computational complexity of optimal policy generation for a multi-agent POMDP [4]. 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 [8].

In the following section, 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 instances of communication only when they are 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, at the potential cost of a small amount of team performance.

3 Dec-Comm algorithm

Policies for multi-agent POMDPs are computationally difficult to generate because agents, observing only their own local observations, must still reason about the possible observations and actions of their teammates. 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. Some work has been done exploring transition-independent systems, that is multi-agent systems in which the local state transitions and observations of each agent are independent of the actions of the other agents on the team, and the team is linked only through a joint reward function, but this independence does not hold in general [9].

As discussed in the previous section, a multi-agent POMDP can be transformed into a single-agent POMDP by communicating at every time step. A standard POMDP solver can 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. Assuming the agents start at a known *synchronized belief*, meaning that the agents have the same probability distribution over the possible states of the world, each agent can construct the joint observation of the team by listening to the individual observations communicated to it, and can therefore calculate both the joint action that is indicated by the policy and the resulting belief of the team. This belief, 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. Because we wish to minimize the use of communication resources, we introduce the Dec-Comm algorithm that, in a decentralized fashion, selects actions based on the possible joint beliefs of the team and chooses to communicate when an agent's local observations indicate that sharing information would lead to an increase in expected reward.

3.1 Reasoning over possible joint beliefs

The Q-MDP method was introduced by Littman *et al.* as an approach for finding approximate solutions to large POMDPs 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 systems' underlying MDPs [10]. In Q-MDP, the best action for a particular belief is chosen according to Q-MDP$(b) = \arg\max_a \sum_{s \in \mathcal{S}} b(s) \times \mathcal{V}_a(s)$, 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.

Likewise, 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 members. We define $\mathcal{L}^t$ to be the set of possible joint beliefs of the team at time *t*. Each $\mathcal{L}^t_i$ 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 $\mathcal{L}^t_i$, b^t is the joint belief at that observation history, and p^t is the probability of the team observing that history.

The algorithm for expanding a single leaf in a tree of possible joint beliefs can be found in Table 1. 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. 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. This enables the agents to compute identical trees in a decentralized fashion. The Q-POMDP heuris-

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

```
GROWTREE($\mathcal{L}_i^t$, $a$)
    $b^t \leftarrow b(\mathcal{L}_i^t)$
    $\mathcal{L}^{t+1} \leftarrow \emptyset$

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

    return $\mathcal{L}^{t+1}$
```

tic, Q-POMDP($\mathcal{L}^t$) $= \arg\max_a \sum_{\mathcal{L}_i \in \mathcal{L}^t} p(\mathcal{L}_i) \times \mathcal{Q}(b(\mathcal{L}_i), a)$, 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 $\mathcal{Q}$ function, $\mathcal{Q}(b^t, a) = \sum_{s \in \mathcal{S}} \mathcal{R}(s, a) b^t(s) + \gamma \sum_{\omega \in \Omega} Pr(\omega|a, b^t) \mathcal{V}^{\pi}(b^{t+1})$, 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 $\mathcal{Q}$ 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. b^{t+1} and $Pr(\omega|a, b^t)$ 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 execute the same joint action at each time step. However, this clearly leads to very conservative action choice, as agents are forced to select an action that takes 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 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 $\omega_j^{t+1}$
            $\vec{\omega}_i^{t+1} \leftarrow \vec{\omega}_j^t \circ \langle \omega_j^{t+1} \rangle$
            $\mathcal{L}^{t+1} \leftarrow \emptyset$
            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 $[\mathcal{L}^{t+1}, \vec{\omega}_j^{t+1}]$
```

3.3 Synchronization

In addition to the communication discussed above, which allows agents to choose to communicate independently of the communication decisions of the other agents, there may also arise a need for a joint communication action that we call synchronization. In a particular domain, if communication is not frequently necessary, the tree of joint beliefs may grow intractably large. Periodic synchronization, a communication action in which all of the agents transmit their observation histories to each other simultaneously, reduces the tree of possible joint beliefs to a single belief. We distinguish synchronization from communication because it is performed solely for the purpose of reducing the size of the tree of possible joint beliefs, independent of whether or not communication at that point in time would be beneficial for the performance of the team.

4 Example

To illustrate the details of our algorithm, we present an example in the two-agent tiger domain introduced by Nair *et al.* [7]. We use the tiger domain because it is small and clear enough to be 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 (P(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 ($\mathcal{T}(s_i, \langle \text{LISTEN}, \text{LISTEN} \rangle, s_i) = 1.0\, \forall s_i$); if either agent opens a door, the world is randomly and uniformly reset to a new state (i.e. $\mathcal{T}(s_i, \langle \text{OPENR}, * \rangle, s_j) = 0.5\, \forall s_i, s_j$). The agents receive two observations, HL and HR, corresponding to hearing the tiger behind the left of 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 still uniformly chosen (i.e. $\mathcal{O}(s_i, \langle \text{OPENR}, * \rangle, *) = 0.5\, \forall s_i$); the probability of an individual agent hearing the correct observation if both agents LISTEN is 0.7 (i.e. $\mathcal{O}(\text{SR}, \langle \text{LISTEN}, \text{LISTEN} \rangle, \text{HR}) = 0.7$). This change makes it such that the optimal policy is to hear two consistent observations (i.e. HR, HR) before opening a door. Table 3 gives the joint observation function for the case in which the agents perform ⟨LISTEN, LISTEN⟩.

Table 3: Joint observation function for the action ⟨LISTEN, LISTEN⟩

State/Observation	HL,HL	HL, HR	HR,HL	HR,HR
SL	0.49	0.21	0.21	0.09
SR	0.09	0.21	0.21	0.49

The reward function for this problem is structured to create an explicit coordination problem between the agents. The highest reward is achieved when both agents open the same door, and that door does not contain the tiger (i.e $\mathcal{R}(\text{SR}, \langle \text{OPENL}, \text{OPENL} \rangle) = +20$). A slightly lower reward is received when both agents open the incorrect door (i.e $\mathcal{R}(\text{SL}, \langle \text{OPENL}, \text{OPENL} \rangle) = -50$). The worst case is when the agents open opposite doors, or when one agent opens the incorrect door while the other agent listens (i.e $\mathcal{R}(\text{SL}, \langle \text{OPENL}, \text{OPENR} \rangle) = -100$). The cost of ⟨LISTEN, LISTEN⟩ is -2. We generated a joint policy for this problem with Cassandra's POMDP solver, using a discount factor of γ = 0.9 [11]. Note that although there are eight 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.

4.1 Time step 0

In this example, the agents start out with a synchronized joint belief of P(SR) = 0.5. According to the policy, the optimal joint action at this belief is ⟨LISTEN, LISTEN⟩, with an expected reward of 18.199738. Because their observation histories are empty, there is no need for the agents to communicate.

4.2 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. Table 4 shows the expected value of taking each possible joint action at the possible beliefs. This table shows that, using the Q-POMDP heuristic, the best possible joint action is ⟨LISTEN, LISTEN⟩.

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 and the bold-ed entries in Table 4 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 decide to perform ⟨OPENR, OPENR⟩. This is an instance in which our algorithm, because it does not yet have sufficient

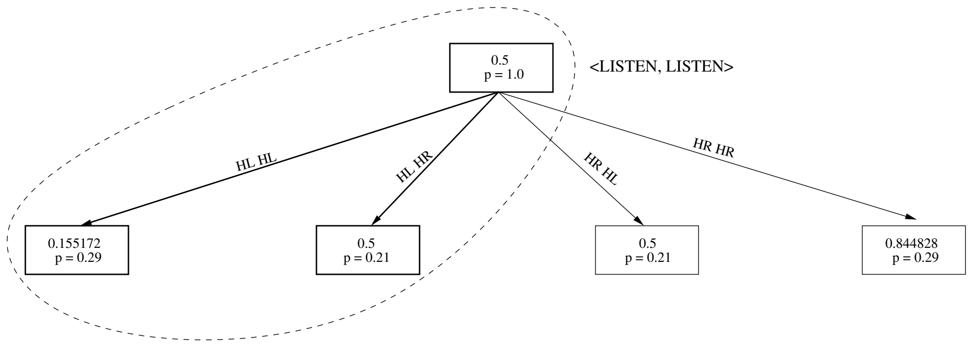

Figure 1: Joint beliefs after a single action

Table 4: Values of the joint beliefs in $\mathcal{L}^1$

	b($\mathcal{L}_i^1$)	p($\mathcal{L}_i^1$)	<OpenL,OpenL>	<OpenR,OpenR>	<Listen,Listen>
1	**0.155**	**0.29**	**-22.758196**	**25.517725**	**21.532808**
2	**0.5**	**0.21**	**1.379765**	**1.379765**	**18.199738**
3	0.5	0.21	1.379765	1.379765	18.199738
4	0.845	0.29	25.517725	-22.758196	21.532808

reason to believe that there will be a gain in expected reward through communication, performs worse than a centralized controller.

4.3 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. Due to space constraints, we do not enumerate all of the resulting joint beliefs. The Q-POMDP heuristic again indicates that the best joint action is ⟨LISTEN, LISTEN⟩.

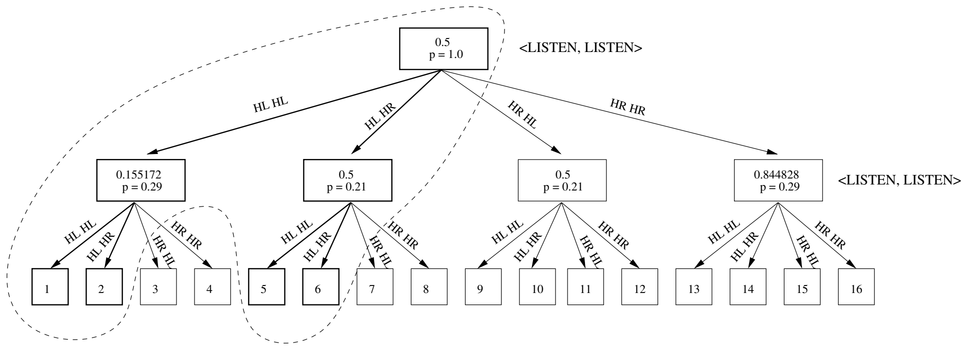

Figure 2: Joint beliefs after the second action

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. Table 5 shows the beliefs and probabilities for the remaining nodes in agent 1's tree of possible joint beliefs, as well as the values of the joint actions in each belief. For the pruned tree, Q-POMDP indicates that the best action is $\langle$OPENR, OPENR$\rangle$. 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.

Table 5: Values of the pruned joint beliefs in $\mathcal{L}^{2'}$

	b($\mathcal{L}_i^{2'}$)	p($\mathcal{L}_i^{2'}$)	<OpenL,OpenL>	<OpenR,OpenR>	<Listen,Listen>
1	0.032635	0.428	-31.335785	34.095314	23.311438
2	0.155172	0.21	-22.758196	25.517725	21.532808
5	0.155172	0.21	-22.758196	25.517725	21.532808
6	0.5	0.152	1.379765	1.379765	18.199738

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 pruning their trees of possible joint beliefs, there is only a single belief that is consistent with the observation histories of both agents, P(SR) = 0.032635. The optimal action for this belief is $\langle$OPENR, OPENR$\rangle$, which is now performed by the agents.

This 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 nor, or examples in which the 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 also clear that the tree of possible joint beliefs grows rapidly when communication is not chosen. In cases where the agents do not communicate for a long period of time, it may be necessary to introduce a synchronization to re-shrink the tree to a tractable size.

5 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 the previous section. In each experiment, the world state was initialized randomly, and the agents were allowed to act for 8 time steps. We ran 30000 trials of this experiment. Table 6 summarizes the results of these trials.

Table 6: Experimental results

	μ_{Reward}	σ_{Reward}	μ_{Comm}	σ_{Comm}
Full Comm.	17.0	37.9	30	0
DEC-COMM	8.9	28.9	2.9	1.1

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 third as much communication as the fully communicating system.

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 [8]. However, this comparison is difficult for several reasons. First of all, the COMMUNICATIVE JESP approach treats communication as an action that can replace other domain-level actions in the policy. Thus, in their model, if an agent chooses to communicate in a particular time step, it cannot take an action. Secondly, their approach can only plan for finite-horizon problems, whereas our system uses an infinite-horizon policy as the solution for the underlying POMDP. Most significantly, however, 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.

6 Conclusion

We present in this paper an approach for trading off the benefits of assuming free communication at policy-generation time, allowing us to use single-agent POMDP solvers to generate infinite-horizon policies for multi-agent POMDPs, with the cost of maintaining agent synchronization and reasoning about necessary instances of communication at execution-time. We introduce the novel approach of maintaining a tree of possible joint beliefs of the team, and describe a heuristic, Q-POMDP, that allows agents to, in a decentralized fashion, select the best action over the possible beliefs. We show through a detailed example and experimentally that our DEC-COMM algorithm makes communication decisions that improve team performance while mimimizing the instances of communication.

In this work, we explicitly maintain the entire tree of possible joint beliefs until it is pruned by communication. Although we describe how synchronization actions can be used to reduce the size of the tree, we believe that there may be other methods for improving the tractability, as well as the performance, of our algorithm. One idea for more efficiently storing the possible joint beliefs is to encode them in a directed acyclic graph instead of a tree, allowing for the compression of identical (or possibly similar) beliefs into a single node. We are also interested in looking at factored representations that may reveal structural relationships between state variables, as opposed to single states. 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] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov Decision Processes. In *Uncertainty in Artificial Intelligence*, 2000.

[2] P. Xuan and V. Lesser. Multi-agent policies: From centralized ones to decentralized ones. In *International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2002.

[3] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable domains. *Artificial Intelligence*, 1998.

[4] D. V. Pynadath and M. Tambe. The communicative Multiagent Team Decision Problem: Analyzing teamwork theories and models. *Journal of AI Research*, 2002.

[5] L. Peshkin, K-E Kim, N Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *Uncertainty in Artificial Intelligence*, 2000.

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

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

[8] R. Nair, M. Roth, M. Yokoo, and M. Tambe. Communication for improving policy computation in distributed POMDPs. In *International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2004.

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

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

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