# Reasoning About Joint Beliefs for Execution-Time Communication Decisions

Maayan Roth
Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
mroth@andrew.cmu.edu

Reid Simmons
Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
reids@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
veloso@cs.cmu.edu

## ABSTRACT

Just as POMDPs have been used to reason explicitly about uncertainty in single-agent systems, there has been recent interest in using multi-agent POMDPs to coordinate teams of agents in the presence of uncertainty. Although multi-agent POMDPs are known to be highly intractable, communication at every time step transforms a multi-agent POMDP into a more tractable single-agent POMDP. In this paper, we present an approach that generates "centralized" policies for multi-agent POMDPs at plan-time by assuming the presence of free communication, and at run-time, handles the problem of limited communication resources by reasoning about the use of communication as needed for effective execution. This approach trades off the need to do some computation at execution-time for the ability to generate policies more tractably at plan-time. In our algorithm, each agent, at run-time, models the distribution of possible joint beliefs. Joint actions are selected over this distribution, ensuring that agents remain synchronized. Communication is used to integrate local observations into the team belief only when those observations would improve team performance. We show, both through a detailed example and with experimental results, that our approach allows for effective decentralized execution while avoiding unnecessary instances of communication.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

## General Terms

Algorithms, Design, Performance


*AAMAS'05,* July 25-29, 2005, Utrecht, Netherlands.


## Keywords

Communication, Distributed Execution, POMDP, Robot Teams

## 1. INTRODUCTION

A fully cooperative team of agents is one in which the members work to achieve a shared reward, and no agent has individual preferences that conflict with the team goals. In addition to domains such as robotic soccer where a cooperative team is inherent in the problem description [17], multi-agent teams are useful for performing tasks that would be difficult or expensive for a single agent. For those tasks, which are present in domains such as planetary exploration or urban search and rescue [5, 12], multi-agent teams can provide additional robustness to failure and may allow for the completion of tasks that would be impossible for a single agent. However, multi-agent teams also present additional challenges. Because there will be instances in the problem space of a multi-agent team where the actions of one agent affect the optimal action choice for other team members, in addition to reasoning about uncertainty in the state of the world, agents must also reason about uncertainty in the internal state of their teammates. The difficulty of reasoning over uncertainty in this collective state of the team is the source of the increased intractability of policy-generation for multi-agent teams.

Partially Observable Markov Decision Problems (POMDPs) have been used extensively to plan over uncertainty in single-agent systems (e.g. [8, 18, 9]). Recently, a multi-agent extension to POMDPs has been proposed as a mechanism for coordinating teams of agents. Unfortunately, the problem of generating optimal policies for these multi-agent POMDPs is known to be NEXP-complete [2], making these problems highly intractable. Communication provides a valuable tool both for improving the performance of a multi-agent team and for improving the tractability of team coordination. Free and instantaneous communication allows for perfect inter-agent coordination by transforming a multi-agent POMDP into a large single-agent POMDP [16] which has lower computational complexity. However, in most domains, communication is a limited resource and therefore cannot be treated as having zero cost. When communication has a cost, it can still be used to improve team performance by allowing agents to share information with their teammates, but reasoning about communication as part of the policy-

generation process does not improve planning tractability.

In this paper, we introduce an approach that exploits the computational complexity benefits of free communication at policy-generation time by generating a centralized plan as if the agents were going to communicate at every time step. Then, at run-time, each agent maintains a distribution of *possible joint beliefs* of the team and chooses to communicate only when it perceives that communication will benefit 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 [16], POIPSG [14]), all of which model cooperative multi-agent teams under *collective partial observability*, a class of observability in which the union of the observations of all of the teammate agents may be insufficient to uniquely identify the current world state. In all of these models, the agents take individual actions and receive local observations, but accumulate a joint team reward. The multi-agent POMDP model consists of the tuple $\langle \alpha, \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, \mathcal{O}, \mathcal{R}, \gamma \rangle$:

- $\alpha$ - the number of agents in the team
- $\mathcal{S}$ - the set of $n$ world states, $\{s^1 \dots s^n\}$
- $\mathcal{A}$ - the set of $m$ possible joint actions of the team, where each joint action, $a^i$, is composed of $\alpha$ individual actions $\langle a^i_1 \dots a^i_\alpha \rangle$
- $\mathcal{T}$ - the transition function ($\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re$), which 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 $a^k$
- $\Omega$ - the set of possible joint observations, where each joint observation, $\omega^i$, is composed of $\alpha$ individual observations, $\langle \omega^i_1 \dots \omega^i_\alpha \rangle$
- $\mathcal{O}$ - the observation function ($\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow \Re$), which gives the probability of observing the joint observation $\omega^i$ after taking action $a^k$ and ending in state $s^j$
- $\mathcal{R}$ - the reward function ($\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$), which indicates the reward that is received when the team starts in a state $s^i$ and takes the joint action $a^k$

While the problem of solving for the optimal policy of a single-agent POMDP is known to be PSPACE-complete [13], the problem of finding the optimal policy, either with or without communication, for a multi-agent POMDP is known to be NEXP-complete [2, 16], putting multi-agent POMDPs in a fundamentally harder complexity class than single-agent POMDPs. For this reason, prior approaches to working with multi-agent POMDP models have primarily focused on finding heuristic approaches to generate locally optimal policies or on methods that may be used to speed up the computation of optimal policies. Hansen *et al.* recently presented a dynamic programming algorithm for finding optimal policies for POIPSGs [7]. Experimental results indicate that in some domains, dynamic programming may provide a substantial speed-up over brute-force searches, but in general, using this method to generate optimal policies remains computationally intractable. Therefore, the majority of work in the area of multi-agent POMDPs has focused on finding heuristic algorithms that may allow for the faster computation of locally optimal multi-agent POMDP policies.

One set of approaches involves placing limiting assumptions on the types of domains that can be solved. Goldman *et al.* identify a range of decentralized control problem classes and demonstrate that by varying the limiting assumptions placed on the problem domains, the overall complexity of the problems varies from P to NEXP [6]. One example of a limiting assumption that affects solution complexity is identified by Becker *et al.*, who define a class of multi-agent POMDP problems with the property of "transition independence". This property holds in domains where the global state can be partitioned into local states of individual agents, and each agent's local actions affect only its own local state and not the state transitions of its teammates [1]. Agents must still coordinate in this framework because they share a team reward that is dependent on joint actions. For domains where this assumption holds, Becker *et al.* present an algorithm that allows for more tractable policy computation.

Other heuristic approaches include a policy search method that restricts the space of policies that can be discovered only to those that can be expressed as finite state controllers with limited memory [14], and an iterative method that repeatedly solves for the optimal policy for a single agent while holding the policies of its teammates fixed, terminating when the policies converge to a local optimum [10]. Emery-Montemerlo *et al.* approximate a POIPSG with a series of smaller Bayesian games and learn a policy over this approximate representation [4]. Their approach is able to solve larger problems than those that could be solved exactly, but is unable to guarantee strict coordination between agents when it is needed.

Recent work has also begun to address the problem of using communication effectively to improve performance of multi-agent teams within the multi-agent POMDP framework. The COM-MTDP framework developed by Pynadath *et al.* provides a theoretical framework for reasoning about communication at plan-time [16]. In practice, however, the number of different messages that agents may wish to communicate can grow as large as the set of every possible observation history observed by an agent, since every observation history may represent a different belief that an agent could wish to convey. Reasoning about communication at plan time would require the enumeration of all of these possible messages, as well as the intended effect of each message on the team belief. One approach, COMMUNICATIVE JESP, addresses this problem by placing a restriction on the frequency at which communication takes place, requiring agents to communicate at least every $K$ time steps [11]. Therefore, only messages that encode observation histories up to length $K$ are considered. However, this restriction creates situations in which agents communicate not because the information is expected to increase perfor-

mance but because the communication was needed to allow for tractable policy computation. Other heuristics strictly limit the allowed communications to a very small set of messages whose effect on belief can be easily encoded as part of the model [19, 20, 4]. Our approach differs from all of these previous approaches in that we reason about communication at execution-time. This allows us to consider as possible messages only those observation histories that have actually been observed, instead of all of the histories that could potentially be seen. Additionally, our approach is applicable to all multi-agent POMDP domains, and does not place any restrictions on the types of policies that can be discovered.

It is known that free communication transforms a multi-agent POMDP into a large single agent POMDP. This is done by making each agent broadcast its local observations to the entire team at each time step. When all of the local observations are known to every team member, they can be treated as a single joint observation, giving the system the same complexity as a single-agent POMDP, PSPACE-complete [16]. However, because communication is not generally free, it is necessary to reason about its effective use.

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 reason about communication decisions at execution-time in exchange for the ability to generate infinite-horizon policies for decentralized teams that would otherwise be highly intractable to compute. Secondly, it minimizes communication, with the potential sacrifice of some amount of reward.

# 3. DEC-COMM ALGORITHM

Policies for multi-agent POMDPs are computationally difficult to generate because agents must reason not only about their own local observations and actions but also 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.

A multi-agent POMDP can be transformed into a single-agent POMDP by having each agent communicate its local observation to all of its teammates at every time step. A standard POMDP solver can then be used to generate a policy that takes as input joint observations and outputs joint actions. The belief over which this policy operates is henceforth referred to as the *joint belief*. Creating policy over joint beliefs is equivalent to creating a centralized controller for the team, but executing this policy with a decentralized team would require agents to communicate their observations at each time step. We wish to reduce the amount of communication resources used by eliminating instances of communication that do not improve team performance. Therefore, in this paper, we introduce the DEC-COMM algorithm that, in a decentralized fashion, selects actions for each agent based on the *possible joint beliefs* of the team and inserts instances of communication when an agent's analysis of its local observations indicate that sharing this information would lead to an increase in expected team reward.

## 3.1 Q-POMDP: Reasoning over possible joint beliefs

The Q-MDP heuristic is an approach for finding approximate solutions to large single-agent POMDPs by first solving the underlying MDP and then, at execution-time, selecting the action that maximizes expected reward over the current belief [9]. The solution for an MDP provides a set of value functions, $\mathcal{V}$, where $\mathcal{V}_a(s)$ is the value of taking action $a$ in state $s$ and henceforth acting optimally. These value functions are used by the Q-MDP heuristic to choose the best action at a given belief, $b$, by examining the average value of a particular action in each state, weighted by the probability of being in that state at the current time:

$$\text{Q-MDP}(b) = \arg\max_a \sum_{s \in \mathcal{S}} b(s) \times \mathcal{V}_a(s) \quad (1)$$

Analogously, we introduce a heuristic, Q-POMDP, which approximates the best joint action for a multi-agent POMDP by reasoning over the values of actions in each possible joint belief, as provided by the solution to the underlying centralized POMDP. In our approach, a joint policy is generated for the system as described above. During execution, each agent calculates the distribution of possible joint beliefs of the team. This distribution can be represented by tree, with each path through the tree representing a possible joint observation history 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 growing a single leaf in a tree of possible joint beliefs given a joint action. Every possible joint observation resulting from the joint action taken at time $t$ generates a child leaf at time $t+1$. $Pr(\omega|a, b^t)$, the probability of receiving an observation $\omega$ while in belief state $b^t$ after having taken action $a$, is calculated for each $\omega \in \Omega$. The resulting belief, $b^{t+1}$, is calculated using a standard Bayesian update [8]. The child leaf associated with this joint observation is composed of the new belief, $b^{t+1}$, the probability of reaching that belief, which is equivalent to the probability of receiving the observation $\omega$ in $\mathcal{L}_i^t$ times the probability of reaching $\mathcal{L}_i^t$, and the observation history $\omega$. It is important to note that this algorithm does not take into account the actual observations seen by each agent, enabling the agents to independently compute identical trees.

Just as Q-MDP approximates a POMDP using the the policy for the underlying MDP, the Q-POMDP heuristic:

$$\text{Q-POMDP}(\mathcal{L}^t) = \arg\max_a \sum_{\mathcal{L}_i^t \in \mathcal{L}^t} p(\mathcal{L}_i^t) \times \mathcal{Q}(b(\mathcal{L}_i), a) \quad (2)$$

approximates a DEC-POMDP using the centralized policy generated for the underlying POMDP, selecting the joint action that maximizes expected reward over all of the possible joint beliefs in $\mathcal{L}^t$. Because this reward is a weighted aver-

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

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

age 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 use the one-step look-ahead $\mathcal{Q}$ function:

$$\mathcal{Q}(b^t, a) = \sum_{s \in \mathcal{S}} \mathcal{R}(s, a) b^t(s) + \gamma \sum_{\omega \in \Omega} Pr(\omega|a, b^t) \mathcal{V}^{\pi}(b^{t+1}) \quad (3)$$

in order to take these actions into account. Whereas the value function, $\mathcal{V}^{\pi}(b)$, is only defined over those actions which appear in the centralized policy $\pi$, the $\mathcal{Q}$ function returns expected reward for any action and belief. $b^{t+1}$ is the belief that results from receiving the joint observation $\omega$ after taking action $a$ in belief state $b^t$. $Pr(\omega|a, b^t)$ and $b^{t+1}$ are calculated as in Table 1.

By selecting an action that maximizes expected reward over $\mathcal{L}^t$, the distribution of possible joint beliefs that is maintained identically by all of the agents on the team, ignoring the local observations of the agents, the Q-POMDP heuristic guarantees that all of the agents will independently select the same joint action at every time step. It is clear, however, that this joint action selection will perform unnecessarily conservatively, choosing an action that takes into account all possible contingencies. In the next section, we introduce the DEC-COMM algorithm, which utilizes communication to allow agents to integrate their true observations into the possible joint beliefs of the team while still maintaining team coordination.

## 3.2 Dec-Comm: Using communication to improve performance

The DEC-COMM algorithm provides agents with a heuristic for determining when communication will be beneficial to team performance. The heuristic specifies that an agent should only communicate when it sees that integrating its own observation history into the joint belief of the team would cause a change in the joint action selected by Q-POMDP. The details of the DEC-COMM algorithm are provided in Table 2. When deciding whether or not to communicate, the agent computes $a_{NC}$, the joint action selected by Q-POMDP over $\mathcal{L}^t$, the current distribution of possible joint beliefs. It then prunes $\mathcal{L}^t$, removing all beliefs that are inconsistent with its own local observation history, to produce a new distribution, $\mathcal{L}'$. The action selected by Q-POMDP over $\mathcal{L}'$, $a_C$, is the action that would be selected by the team were the agent to communicate to its teammates. If the actions differ, this indicates that communication may cause an increase in expected team reward, and DEC-COMM instructs the agent to broadcast its observation history to all of its teammates. (Note that this algorithm can easily be extended to take into account communication cost by comparing the difference in expected reward achieved by actions $a_{NC}$ and $a_C$ over the possible beliefs in $\mathcal{L}'$ to the cost of communication.)

Agents that receive communication from their teammates prune their distributions of possible joint beliefs to take into account their teammate's observation history. Because receiving this new information may prompt an agent to decide to communicate its own observation history, there may be multiple instances of communication in each time step. Therefore, agents must wait a fixed period of time to allow the system to quiesce before acting.

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 $\omega_j^{t+1}$
      $\vec{\omega}_j^{t+1} \leftarrow \vec{\omega}_j^t \circ \langle \omega_j^{t+1} \rangle$
      $\mathcal{L}^{t+1} \leftarrow \emptyset$
      **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** $[\mathcal{L}^{t+1}, \vec{\omega}_j^{t+1}]$

**Table 2: One time step of the Dec-Comm algorithm for an agent $j$**

# 4. EXAMPLE

To illustrate the details of our algorithm, we present an example in the multi-agent tiger domain introduced by Nair *et al.* [10]. We use the tiger domain because while it is small and therefore easily understood, it is still a problem that requires coordinated behavior between the agents. The world in the tiger problem consists of two states, SL and SR. These states correspond to two doors, one on the left and one on the right. Behind one door is a tiger and behind the other is a treasure. (The state SL indicates that the tiger is behind the left door.) The agents start out with a uniform distribution over these two states ($b(\text{SR}) = 0.5$).

The goal in the tiger problem is to open the door with the treasure behind it. To this end, there are three actions that can be performed: OPENL, which opens the left

door, OPENR, which opens the right door, and LISTEN, an information-gathering action that does not change the state of the world but provides knowledge about the position of the tiger. In the multi-agent tiger problem, there are two agents acting the world, each of whom can independently perform any of the three actions.

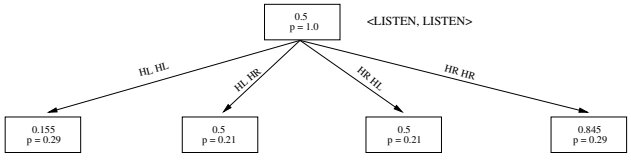
Rewards for the tiger problem are structured such that agents should avoid opening the door with the tiger. The maximum possible reward (+20) occurs when both agents open the door with the treasure. An explicit coordination problem is built into the domain by making it preferable for the agents to both open the wrong door together (reward = -50) rather than for each agent to open a different door (reward = -100). There is a small cost of -2 if the agents perform the joint action ⟨LISTEN, LISTEN⟩.

To increase their chances of opening the correct door, the agents must increase their certainty about the position of the tiger. At each time step, each agent receives an independent observation. The possible observations are HEARLEFT (or HL), indicating that the tiger is behind the left door, and HEARRIGHT (HR), indicating that the tiger is behind the right door. If either agent opens a door, the observations received in that time step have no informative value. If both agents perform LISTEN, the observations are independent (meaning agents may hear different observations in the same time step) and noisy, correctly identifying the position of the tiger with 0.7 probability. This particular observation model, a slight modification of the model presented by Nair *et al.*, makes it so that the optimal policy for a centralized team would be to hear two consistent observations (e.g. a joint observation of ⟨HR, HR⟩) before opening a door.

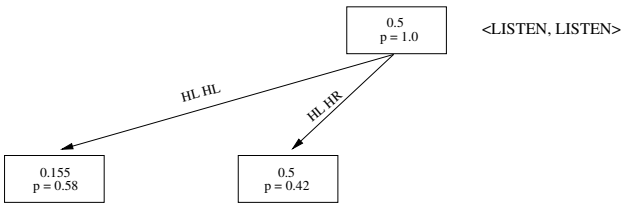
We generated a joint policy for this problem with Cassandra's POMDP solver [3], using a discount factor of $\gamma = 0.9$. Note that although there are nine possible joint actions, all actions other than ⟨OPENL, OPENL⟩, ⟨OPENR, OPENR⟩, and ⟨LISTEN, LISTEN⟩ are strictly dominated, and we do not need to consider them.

**Time Step 0:**
In this example, the agents start out with a synchronized joint belief of $b(\text{SR}) = 0.5$. The joint policy indicates that, for this belief, the best action is ⟨LISTEN, LISTEN⟩. The agents have no need to communicate at this point because their local observation histories are empty.

**Time Step 1:**
After executing the joint action ⟨LISTEN, LISTEN⟩, agent 1 observes HEARLEFT. (Although agent 1 does not know that this has happened, agent 2 has also observed HEARLEFT at this time step.) Agent 1 must now execute GROWTREE to determine the possible joint beliefs. There are four possible joint observations that could have been observed. Therefore, $\mathcal{L}^1$ contains four leaves:

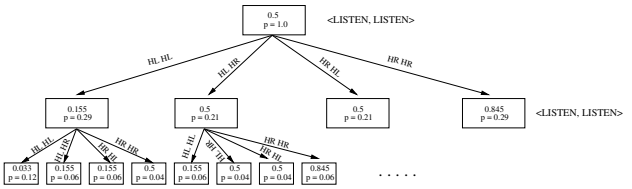



The Q-POMDP heuristic, executed over the leaves in $\mathcal{L}^1$, indicates that $a_{NC}$, the joint action that would be chosen without communication, is ⟨LISTEN, LISTEN⟩. To determine if communication is necessary, agent 1 now prunes $\mathcal{L}^1$ to contain only those leaves consistent with its own local observation of HL:

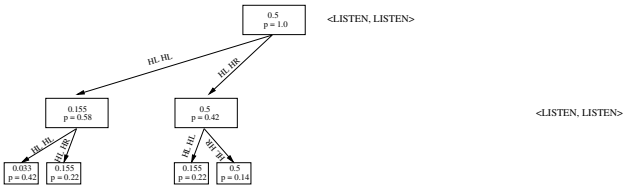



Running Q-POMDP on the pruned tree shows that the best post-communication action, $a_C$, would also be ⟨LISTEN, LISTEN⟩. Because agent 1 determines that communication would not change the joint action, it chooses not to communicate at this time step. It is important to note that at this point, if both agents could somehow be forced to communicate their local observations, only a single joint belief would remain, and the best action for that belief would be ⟨OPENR, OPENR⟩. This is an instance in which our algorithm, because each agent is reasoning over the information locally available to it and does not yet have sufficient reason to believe that communication will improve expected reward, under-performs a centralized controller or a system in which the agents communicate at every time step.

**Time Step 2:**
The agents execute ⟨LISTEN, LISTEN⟩, and again, agent 1 observes HEARLEFT. (Likewise, agent 2 also observes HEARLEFT.) Agent 1 performs GROWTREE on (the unpruned) $\mathcal{L}^1$ to get $\mathcal{L}^2$, which has 16 leaves:



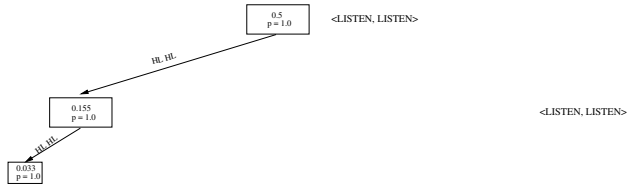

The Q-POMDP heuristic determines that the best action choice for $\mathcal{L}^2$ is, again, ⟨LISTEN, LISTEN⟩. Agent 1 reasons about communication by pruning all of the leaves that are not consistent with its entire observation history:




For the pruned tree, the best action selected by Q-POMDP is ⟨OPENR, OPENR⟩. Because it sees that the action selected before communication differs from the action that would be selected after communication, agent 1 chooses to communicate its observation history to agent 2. Once it has communicated, agent 1's observation history is considered part of the shared team information, and agent 1 is able to use the pruned tree as $\mathcal{L}^2$, the distribution of possible joint beliefs.

In the meantime, agent 2 has been performing identical computations and, independently of agent 1, also decides to communicate its observation history (which is two instances of HL). When agent 1 receives this communication from

agent 2, it prunes $\mathcal{L}^2$ to contain only those leaves that are also consistent with agent 2's observation history:




The set of possible joint beliefs for the team now consists of a single belief state, $b(\text{SR}) = 0.033$. For this belief, the best joint action is $\langle \text{OPENR}, \text{OPENR} \rangle$.

## 5. PARTICLE FILTER REPRESENTATION

The above example shows a situation in which both agents decide to communicate their observation histories after two time steps, reducing the size of $\mathcal{L}^2$ to a single leaf. It is easy to construct situations in which one agent would choose to communicate but the other agent would not, or examples in which neither agent would 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 unnecessary. To address cases where the agents do not communicate for a long period of time, we present a fixed-size method for modeling the distribution of possible joint beliefs by using a particle filter.

A particle filter is a sample-based representation used to model an arbitrary probability distribution with a fixed amount of memory. Particle filters have frequently been used with single-agent POMDPs (e.g. for state estimation during execution [15]). In applying a particle filter to the modeling of the team's distribution of possible joint beliefs, we draw our inspiration from an approach that finds a policy for a continuous state-space POMDP by maximizing over the distribution of possible belief states, which is represented by a particle filter [18].

In our particle filter representation, each particle $\mathcal{L}_i^t$ in the distribution $\mathcal{L}^t$ is a tuple of $\alpha$ observation histories, $\langle \vec{\omega}_a \ldots \vec{\omega}_\alpha \rangle$, corresponding to a possible observation history for each of the $\alpha$ agents. Together, these individual observation histories form a single possible joint observation history. Because the starting belief, $b^0$, and the history of joint actions taken by the team, $\vec{a}$, are known, a joint observation history uniquely identifies a possible joint belief. The likelihood of a particular belief is indicated by the frequency of occurrence of the particle representing that belief.

Every agent stores two particle filters, $\mathcal{L}_{joint}$, which represents the possible joint beliefs of the team, pruned only by communication, and $\mathcal{L}_{own}$, those beliefs that are consistent with the agent's own observation history. At each time step, the beliefs represented in the particle filters are propagated forward according to the algorithm presented in [18]. The possible next observations for $\mathcal{L}_{joint}$ are sampled from all possible joint observations, and the possible next observations for $\mathcal{L}_{own}$ are only those joint observations that are consistent with the agent's own local observation at that time step.

The DEC-COMM algorithm is executed as described in Section 3, with $\mathcal{L}_{joint}$ used to generate $a_{NC}$ and $\mathcal{L}_{own}$ used to generate $a_C$. However, a 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 can only approximate the distribution. Simply removing those particles not consistent with the communicated observation history (equivalent to the pruning done for the tree) 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 (e.g. if one agent observes $\langle \text{HL}, \text{HL} \rangle$, it is unlikely that the other agent will have observed $\langle \text{HR}, \text{HR} \rangle$). 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 corresponding to agent $i$ in each particle in $\mathcal{L}_{joint}$ is compared to $\vec{\omega}_i^t$. The comparison asks the question, "Suppose an agent in a team that started in belief $b^0$ and executed the joint action history $\vec{a}^t$ has observed the individual observation history $\vec{\omega}_i^t$. What is the likelihood that an identical agent could have observed an alternate observation history $\vec{\omega}_j^t$?" We call the value returned by this comparison the "similarity" of the two histories, and use it as a weight for the particle. The particles are then resampled according to the calculated weights, and the agent $i$ observation history for every particle in the new distribution is replaced with $\vec{\omega}_i^t$.

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 $s \in \mathcal{S}$
      $b(s) \leftarrow \mathcal{O}(s, a^{t'}, \omega_i^{t'}) b(s)$
    normalize $b$
    $sim \leftarrow sim \times \sum_{s \in \mathcal{S}} \mathcal{O}(s, a^{t'}, \omega_j^{t'}) b(s)$
    **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** $sim$

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

## 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. Because we wished to demonstrate our approach on a larger problem than the two-agent tiger domain described in Section 4, we ran our experiment on a planetary exploration domain with 32 states. In this domain, there are two rocks, each of which may be of either type A or type B, and two rovers that are attempting to gather scientific data about the rocks. One

rover is able to successfully characterize type-A rocks, and the other can characterize B rocks. At the start of the problem, the type of each rock is unknown. To create an explicit coordination problem between the agents, we specify that each rover may be at the location of either rock, but that they may not be at the same rock at the same time.

At every time step, each rover has the choice to attempt to characterize the rock at its current location (with a cost of -10), attempt to move to the other rock (with a cost of -3), or do nothing. Because the agents must be at different rocks at all times, MOVE only succeeds when both agents select that action simultaneously. CHARACTERIZE succeeds with 0.85 probability if the agent is attempting to characterize the correct type of rock. A successful characterization earns the team a reward of +30. At every time step, each agent receives a local observation of the type of the rock at its current location. The goal of the problem is to successfully characterize both rocks as quickly as possible.

We ran 500 trials of this experiment, each time initializing the problem to a random configuration of rock types and rover positions, and counted the average number of time steps until both rocks were successfully characterized. We compared the performance of a team with full communication to teams that employed the DEC-COMM algorithm, both with the tree and particle filter representations. The team using a particle representation used 50 samples to represent the possible beliefs. Table 4 summarizes the results of these trials.

| | $\mu$ Steps | $\sigma$ Steps | $\mu$ Comm | $\sigma$ Comm |
|---|---|---|---|---|
| Full Comm. | 3.27 | 1.01 | 6.54 | 1.01 |
| DEC-COMM (tree) | 3.40 | 1.00 | 2.34 | 0.63 |
| DEC-COMM (particles) | 3.38 | 0.99 | 2.82 | 0.75 |

**Table 4: Experimental results. $\mu_Steps$ is the mean number of time steps needed to characterize both rocks. $\mu_Comm$ is the mean number of communication instances per trial.**

From these results, it can be seen that, for this domain, there is a very small reduction in performance for teams using the DEC-COMM algorithm compared to a centralized team, but the reduction in communication usage is substantial. Note that in the centralized team, both agents communicate their observation in each time step, so the number of communication instances is exactly twice the number of time steps needed to complete the problem. The DEC-COMM teams demonstrate that much of this communication is extraneous, and therefore they choose to communicate significantly less. There is also no substantial difference in performance between a team using an exact tree representation of joint belief and a tree that approximates belief using a particle filter. However, for other domains, the number of particles needed to accurately model joint belief may be larger.

# 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 distribution of possible joint beliefs of the team, and describe a heuristic, Q-POMDP, that selects the best joint action over the possible joint beliefs in a decentralized fashion. We show both through a detailed example and experimentally that our DEC-COMM algorithm inserts communication actions as needed into the team execution only when it will improve team performance, thereby reducing the amount of communication needed for successful decentralized execution of a centralized policy. We also provide a fixed-size method for maintaining the distribution of possible joint team beliefs.

In the future, we intend to investigate 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 utility of *ask* communication, in which agents request information from their teammates, as opposed to the *tell* communication employed in this paper.

# 8. ACKNOWLEDGMENTS

This work has been supported by several grants, including NASA NCC2-1243, and by Rockwell Scientific Co., LLC under subcontract no. B4U528968 and prime contract no. W911W6-04-C-0058 with the US Army. This material was based upon work supported under a National Science Foundation Graduate Research Fellowship. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsoring institutions, the U.S. Government, or any other entity.

# 9. REFERENCES

[1] 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.

[2] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *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. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2004.

[5] D. Goldberg, V. Cicirello, M. B. Dias, R. Simmons, S. Smith, T. Smith, and A. Stentz. A distributed layered architecture for mobile robot coordination: Application to space exploration. In *International*

*NASA Workshop on Planning and Scheduling for Space*, 2002.

[6] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of AI Research*, 2004.

[7] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *National Conference on Artificial Intelligence*, 2004.

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

[9] 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.

[10] 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.

[11] 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.

[12] R. Nair, M. Tambe, and S. Marsella. Team formation for reformation in multiagent domains like RoboCupRescue. In *RoboCup-2002 International Symposium*, 2003.

[13] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 1987.

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

[15] P. Poupart, L. E. Ortiz, and C. Boutilier. Value-directed sampling methods for monitoring POMDPs. In *Uncertainty in Artificial Intelligence*, 2001.

[16] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI Research*, 2002.

[17] M. Roth, D. Vail, and M. Veloso. A real-time world model for multi-robot teams with high-latency communication. In *International Conference on Intelligent Robots and Systems*, 2003.

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

[19] 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.

[20] P. Xuan, V. Lesser, and S. Zilberstein. Formal modeling of communication decisions in cooperative multiagent systems. In *Workshop on Game-Theoretic and Decision-Theoretic Agents*, 2000.