Decentralized Planning under Uncertainty for Teams of Communicating Agents

Matthijs T.J. Spaan
University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
mtjspaan@science.uva.nl

Geoffrey J. Gordon
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

Nikos Vlassis
University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
vlassis@science.uva.nl

ABSTRACT

Decentralized partially observable Markov decision processes (DEC-POMDPs) form a general framework for planning for groups of cooperating agents that inhabit a stochastic and partially observable environment. Unfortunately, computing optimal plans in a DEC-POMDP has been shown to be intractable (NEXP-complete), and approximate algorithms for specific subclasses have been proposed. Many of these algorithms rely on an (approximate) solution of the centralized planning problem (i.e., treating the whole team as a single agent). We take a more decentralized approach, in which each agent only reasons over its own local state and some uncontrollable state features, which are shared by all team members. In contrast to other approaches, we model communication as an integral part of the agent's reasoning, in which the meaning of a message is directly encoded in the policy of the communicating agent. We explore iterative methods for approximately solving such models, and we conclude with some encouraging preliminary experimental results.

General Terms

Theory, Algorithms

Keywords

Artificial intelligence, planning under uncertainty, cooperative multiagent systems, decentralized POMDPs

1. INTRODUCTION

Planning refers to the process of computing a conditional sequence of actions that fulfill a given task as well as possible. It is a crucial part of any intelligent agent; human, robot or software agent alike. We are interested in planning under various forms of uncertainty for cooperative multiagent systems. First of all, the agent might be uncertain regarding the exact consequence of executing a particular action. Furthermore, the agent's sensors may be noisy or provide only a limited view of the environment. As the agent is part of a team, a third source of uncertainty are its teammates, as it should consider their actions as part of its own decision making. In this paper, we study scenarios in which agents have the ability to communicate, but bandwidth is limited and communication has a certain cost. The range of possible application domains is broad, including cooperative robotics [6], sensor networks [13], distributed space applications [4], communication networks [16] and supply chain management [7].

Single-agent planning under uncertainty has been formalized in the partially observable Markov decision process (POMDP) framework [12]. A POMDP defines stochastic models to capture the uncertain effects of actions and the fact that sensors are imperfect and have a limited scope. In particular, the transition model governs how the system's state is influenced by actions, and the observation model stochastically relates states to observations. Solving a POMDP provides the agent with an optimal plan for a given task, but it is intractable to compute exact solutions in general (as it is known to be PSPACE-complete [17]). However, recent years have seen much progress in the development of approximate methods [19, 20, 23, 24].

Planning for a team of agents proves to be significantly harder than planning for just a single agent. In order for the team to act optimally, each agent not only has to consider the consequences of its own action, but at the same time also the actions its teammates will execute. As each agent does not know what the other agents observe, it will be hard to predict their actions. Decentralized POMDPs (DEC-POMDPs) generalize the POMDP framework to cooperative multiagent settings, but solving finite horizon DEC-POMDPs has been proven to be NEXP-complete [2], a considerable jump in complexity from the single agent case. Besides brute-force enumeration and evaluation of all possible policy tuples, only one algorithm is known for computing optimal solutions [11], which in practice is limited to very small problems. As the general setting is very hard to solve optimally, research has focused on heuristic methods, which often also make additional assumptions on the planning domain [27, 8, 1, 10, 18].

Allowing agents to communicate with each other can increase team performance, as it mitigates two sources of uncertainty: a message from a teammate can either provide information about the state of the system, or about the teammate's intentions. In particular, if one assumes that the agents have unlimited free access to a noise-free communication channel, the multiagent planning problem reduces to a single-agent one: each agent shares its observations with all teammates, which allows each one of them to select the optimal joint action, and execute its part of the joint action vector. However, although optimal, such an approach is not desirable for several reasons. First of all, in reality a communication channel is likely to be bounded in bandwidth, and sending a message will have a particular cost. Second, the resulting single agent POMDP will still be hard to solve, as the size of the problem description grows exponentially in the number of agents. Therefore, we choose to tackle the problem in a decentralized manner, in which each agent considers only its local state plus some shared uncontrollable state features, instead of the cross-product of the local states of all agents and the uncontrollable state space. Furthermore, we consider communication as an integral part of the system, and model it directly in the DEC-POMDP, in contrast with other approaches in which communication decisions are governed by a separate algorithm [8, 5, 14, 22].

We propose a decentralized model for tackling cooperative multiagent planning problems, which incorporates a communication channel. In contrast to most work on distributed POMDPs, we do not define an explicit communication language, but instead treat the semantics of communication as part of the optimization problem. We will present an iterative method for computing a joint policy for the team in a decentralized fashion. Given a set of fixed policies for all agents but agent i, we convert the DEC-POMDP to a POMDP from i's perspective which incorporates the expected contribution to the joint team reward of the other agents' policies. We propose a heuristic method for computing a communication policy and we conclude with encouraging preliminary experimental results in two domains.

2. PLANNING FOR MULTIAGENT TEAMS

We will start by reviewing the cooperative multiagent planning problem from a decision-theoretic perspective. First we will briefly introduce partially observable Markov decision processes (POMDPs), a rich mathematical framework for single-agent planning under uncertainty. Next, we will look at decentralized POMDPs which constitute a multiagent extension of the POMDP model, and at existing exact and approximate methods for solving them.

2.1 Partially observable Markov decision processes

A partially observable Markov decision process (POMDP) models the repeated interaction of an agent with a stochastic environment, parts of which are hidden from the agent's view [12]. The agent's goal is to perform a task by choosing actions which fulfill the task best. At each time step the environment is in a state $s \in S$, and the agent executes an action $a \in A$, which will move the environment to a new state s' according to a stochastic transition model $p(s'|s,a)$. As a by-product the environment will generate a scalar reward signal $R(s,a)$, which defines the agent's task. Next the agent receives an observation $o \in O$ which is stochastically related to s' according to $p(o|s',a)$, but in general will not uniquely identify s'. As such, the agent is uncertain about the true state of the system, which is often modeled as a probability distribution over all states. The probability distribution, or belief $b(s)$ in POMDP terminology, summarizes all relevant history and is updated using Bayes' rule:

$$b^{ao}(s') = \frac{p(o|s',a)}{p(o|a,b)} \sum_{s \in S} p(s'|s,a)b(s), \qquad (1)$$

where $p(o|a,b) = \sum_{s' \in S} p(o|s',a) \sum_{s \in S} p(s'|s,a)b(s)$ is a normalizing constant.

The goal of the agent is to compute an optimal policy $\pi(b)$ that maps beliefs to actions. Optimal is defined as maximizing a certain performance measure, for instance the expected discounted sum of rewards $E\big[\sum_{t=0}^{h} \gamma^t R\big]$, where h is the planning horizon and γ is a discount rate, $0 < \gamma < 1$. Without going into details of exact POMDP solving, it suffices to say that computing an optimal policy is intractable in general [17]. In recent years a number of approximate algorithms including PBVI [19], BPI [20], HSVI [23], and PERSEUS [24] have been proposed, which allow for solving larger POMDP domains by, e.g., exploiting structure in the domain.

2.2 Decentralized POMDPs

A straightforward extension of the POMDP framework to cooperative multiagent settings is called the decentralized POMDP (DEC-POMDP) model [2]. Instead of a single agent we now consider a set of agents, which share the same reward function. If the agents have individual (and differing) reward functions, the model is known as a partially observable stochastic game (POSG) [11]. A DEC-POMDP is defined as follows:

- $I = \{1, \dots, n\}$ is a set of n agents.
- S is a finite set of states.
- A_i is a finite set of actions for agent i.
- O_i is a finite set of observations for agent i.
- $p(s'|s,a)$ is the joint transition model which defines the probability of jumping to state $s' \in S$ given that the team executed joint action $a \in A_1 \times \ldots \times A_n$ in $s \in S$.
- $p(o|s,a)$ is the joint observation model which defines the probability that after taking joint action a and ending up in state s results in joint observation $o \in O_1 \times \ldots \times O_n$.
- $R : S \times A_1 \times \ldots \times A_n \to \mathbb{R}$ is the reward function which gives a scalar reward signal to the team.
- $b_i^0 \in \Delta(S)$ is the initial state distribution.
- γ is a discount rate, $0 \leq \gamma < 1$.
- h is the planning horizon.

Contrary to the single agent case, in a DEC-POMDP it is not possible in general to compute a belief state as an agent only knows its local observation o_i but not the complete observation vector o, which is required for computing the belief update (1). As such, each agent chooses its action a_i at time t based on its policy $\pi_i : \times_{t-1}(A_i \times O_i) \to A_i$, i.e., it only considers its own history of actions taken and observations received until time t. The goal of the team of agents is to compute a joint policy $\pi = \{\pi_1, \dots, \pi_n\}$ which maximizes the future discounted reward $V^\pi = E_\pi\big[\sum_{t=0}^{h} \gamma^t R\big]$. Unfortunately, computing an optimal policy π^* is NEXP-complete [2], and there is only one known algorithm for computing an exact solution (apart from brute-force), which we will describe next [11].

2.3 Solving decentralized POMDPs

Hansen et al. [11] present a dynamic programming operator for finite horizon POSGs, which allows for incremental construction of policies: starting from a solution for horizon t it computes a solution for horizon $t+1$. When the POSG is a DEC-POMDP, i.e., the agents share the same reward function, the algorithm will result in a set of policies optimal for a certain planning horizon. A belief is maintained over the cross-product of the state of system and the future conditional plans, or policy trees, of all other agents. However, the set of possible policy trees grows doubly exponential in the planning horizon, which renders a brute-force approach infeasible in practice. To combat the growth, at every iteration (very weakly) dominated policy trees are pruned. A policy tree for agent i is dominated if its removal does not decrease the value of any belief for agent i. However, even with pruning at every iteration the algorithm can solve only very small problems before running out of memory.

As solving a DEC-POMDP is intractable in the general case, methods have been proposed for approximately solving more restricted DEC-POMDP variations. Simplified models that have been studied include extra assumptions regarding the observation model, in particular that an agent's observations are independent from those of its teammates [8] or that each agent can observe its own local state with full certainty, which reduces the problem to a decentralized MDP [27, 1, 10]. The latter assumption requires a factored state representation in which the state space is the cross-product of each agent's local state space and optionally a set of shared external state features which each agent can observe but not influence. Examples include time or any other external information relevant to the team's task. Given a factored state space one can further assume that the agents' transitions are independent [27, 1, 18], or impose a particular joint reward structure [1]. Apart from restricting the model one can also limit the expressive power of the solution method, for instance by only searching in the space of finite state controllers of a particular size [3, 25]. Another way of computing a locally optimal solution is to optimize one agent's policy while keeping the other policies fixed, and iterate over all agents until convergence [15, 5, 3].

A different dimension to tackle decentralized POMDP models is to assume the agents have a way to communicate with each other (which is not modeled in the DEC-POMDP). As we noted above, if a team of agents is allowed to communicate for free and without limitations, then each agent can broadcast at every time step its perceived observation to all of its teammates. As a result every agent knows the joint observation vector which it uses to update the joint belief of the team, based on which the next joint action is selected. As such, free communication reduces the distributed control problem to a centralized, single-agent one, which can be solved using standard POMDP solution techniques. However, in reality communication is often not free or unlimited, and modeling a team of agents in this way might not be desirable. Nevertheless, several methods treat the multiagent system as if communication were truly free, solve the large centralized POMDP, and execute the resulting joint policy in a distributed fashion while imposing communication constraints [5, 14, 22]. While executing the policy a limited form of communication is used to attempt to preserve coherent behavior of the team. Note that the communication decisions are not part of the centralized policy, but are made by a separate algorithm, which for instance monitors the uncertainty regarding the joint belief. We refer to [21, 9] for a detailed analysis of communication issues arising in DEC-POMDP settings.

In contrast, Xuan et al. [27] propose a model that incorporates the communication decision directly in the agent's policy. It adds a communication sub-stage to the decision process, in which an agent decides whether it will communicate with its teammates. In this way an agent can explicitly reason about communication, instead of relying on an independent instrument to handle communication issues. Extending this framework, Goldman and Zilberstein [8] present a formal model for decentralized control with communication decisions based on the DEC-POMDP model. In such decentralized models agents reason over their local state, instead of considering the state of all teammates. A decentralized model seems more appropriate for agents who cannot observe each other's state nor have free and unrestricted communication at their disposal. Decentralized models also do not suffer from the exponential growth in the size of the centralized state space when the number of agents increases. Limited communication abilities, however, can be exploited to improve the team's performance, by allowing agents to share information at a certain cost.

3. PROPOSED MODEL

In [8, 9] a framework for modeling cooperative multiagent systems as a decentralized POMDP with a global reward function is proposed. The agents reason explicitly over their communication decisions, but the observations of each agent are independent. We propose a different decentralized model which draws on [8, 9], but in which a communication channel is established by allowing agents to send messages as part of their action vector, and messages are received in the next time step as part of the recipients' observation vectors. In contrast to most work on distributed POMDPs, we do not manually define a communication language, but treat the semantics of communication as part of the optimization problem.

We refine the DEC-POMDP model as described in Sec. 2.2 as follows:

- The state space S factors into $n+1$ components: $S = S_0 \times S_1 \times \ldots \times S_n$, where S_i is the set of state features relevant to agent i and S_0 the set of external state features, which are shared by all agents but which cannot be controlled. A local state of an agent i is defined as $\bar{s}_i = (s_0, s_i)$, where $s_0 \in S_0, s_i \in S_i$.

- Σ_i is a finite set of possible messages, and $\sigma_i \in \Sigma_i$ denotes a message sent by agent i. Not sending any message is defined as sending the empty message $\sigma_i = \emptyset$.

- $A_i = A_i^d \times A_i^\sigma$ is the action set for agent i, where A_i^d is the set of domain level actions and A_i^σ the set of communication acts.

- $O_i = O_i^d \times O_i^\sigma$ is the observation set for agent i, where O_i^d is the set of domain level observations the agent receives through its sensors, and $O_i^\sigma = \Sigma_1 \times \ldots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \ldots \times \Sigma_n$ the set of possible messages the agent can receive from any of its teammates.

- The joint transition model $p(s'|s,a)$ depends only on the domain level component of each agent's action (sending messages does not influence the state of the environment). We assume that the agents' transitions are independent, allowing us to factor the joint transition model:

$$p(s'|s,a) = p(s'_0|s_0) \prod_{i=1\ldots n} p(s'_i|s_i, a^d_i), \qquad (2)$$

  where $s'_0, s_0 \in S_0$, $s'_i, s_i \in S_i$, and $a^d_i \in A^d_i$.

- The local observation model $p(o_i|\bar{s}_i, a_i)$ specifies the probability of receiving observation $o_i = (o^d_i, o^\sigma_i)$ after executing action a_i and ending up in $\bar{s}_i$.

- We define $p(o^\sigma_i|\bar{\sigma})$ as the probability that a message vector $\bar{\sigma} = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n)$ will be interpreted as a particular observation o^σ_i, which can be used to model noise in the communication channel.

- The reward function $R(s,a)$ gives the team of agents a scalar reward at each time step for taking joint action a in state s. It is defined as the sum of each agent's local reward function and the joint reward of the team:

$$R(s,a) = \sum_{i=1\ldots n} R_i(\bar{s}_i, a_i) + R(s_0, \ldots, s_n, a_1, \ldots, a_n). \qquad (3)$$

  The local reward $R_i(\bar{s}_i, a_i)$ is defined as the sum of the reward for the domain level action $R_i(\bar{s}_i, a^d_i)$ and the communication cost $R_i(\bar{s}_i, a^\sigma_i)$:

$$R_i(\bar{s}_i, a^\sigma_i) = \begin{cases} 0 & \text{if } \sigma_i = \emptyset \\ r_c < 0 & \text{otherwise} \end{cases} \qquad (4)$$

- Initial belief b^0_i, discount rate γ, and planning horizon h as in Sec. 2.2.

In the remainder of this paper we shall limit our discussion to two agents, but our methods and experiments generalize to more than two agents. Broadcast communication is used, and we will focus on noise-free communication, fixing $p(o^\sigma_i|\bar{\sigma})$ to be the identity matrix.

3.1 Example: Multiagent Heaven or Hell

To give some intuition on the range of problems our model captures we will now discuss a very simple example application. The "Multiagent Heaven or Hell" problem (inspired by [26]) is a scenario in which two agents have to meet in one of two locations, one called "heaven" and the other one "hell", see Fig. 1. At the start of each trial heaven is positioned either in the bottom-left state ($s_0 = \textit{left}$) or in the bottom-right state ($s_0 = \textit{right}$) with equal probability, and hell will be located in the other state. If the agents meet in heaven the team receives a reward of $r_h > 0$, but meeting in hell is penalized with a reward of $-r_h$. However, each agent does not know the location of heaven or hell until it visits the "priest" state, in which it receives an observation indicating whether heaven is left or right. Visiting the priest results in an individual negative reward of $r_p < 0$ for the particular agent, and we assume that the reward obtainable in heaven makes visiting the priest a viable option, i.e., $r_h > -nr_p$.

If no communication is possible, or the communication cost ($-r_c$) is prohibitively large, the optimal policy could be for each agent to visit the priest, find out where heaven

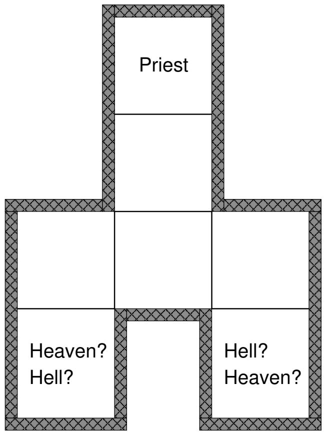


Figure 1: Multiagent Heaven or Hell environment. The priest (top state) can tell the agents where heaven is located (bottom-left or bottom-right state).

is located and go there. However, when communication is relatively cheap, i.e., $r_c > r_p$, more interesting scenarios become feasible. For instance, only agent i could ask the priest and inform agent j by sending it a message. This would save agent j the cost of visiting the priest and still be able to tell heaven from hell. However, instead of defining such semantics a priori, we would like the agents to learn when and how to communicate, and how to interpret incoming messages. In particular, the agents should optimize their use of communication with respect to the cost of sending a message r_c.

3.2 Belief tracking

In a general DEC-POMDP setting it is not possible for an agent to maintain an exact belief over the true state of the system, as it only receives its own part of the observation vector. As such, agent i has to reason about what observation agent j might have observed, but at the same time agent j is modeling agent i, which leads to the very high complexity class of solving DEC-POMDPs. As a result of the lack of independence between agents, only approximate beliefs can be computed. Alternatively, one can assume that agent i's observations are independent from those of agent j [15], which together with a common transition independence assumption precludes any use of direct communication in the DEC-POMDP model, as the agents cannot influence each other. In contrast, in our decentralized model the domain level component o^d_i of an observation $o_i = (o^d_i, o^\sigma_i)$ only depends on an agent's local state $\bar{s}_i$, but the communication component o^σ_i does depend on agent j, as follows

$$\begin{aligned} p(o_i|\bar{s}_i, a_i) &= p(o^d_i, o^\sigma_i|\bar{s}_i, a_i) && (5) \\ &= p(o^d_i|\bar{s}_i, a_i)p(o^\sigma_i), && (6) \end{aligned}$$

where $p(o^\sigma_i)$ is the probability that the other agents send agent i a particular message vector in the previous time step, which we will approximate based on their policies. Consequently, we can only perform an approximate belief update, but it allows us to model a built-in communication channel.

Analogous to (1), agent i's belief b_i is updated as follows when it takes action a_i and receives observation o_i:

$$b_i^{a_i o_i}(\bar{s}'_i) = \frac{p(o_i^d|\bar{s}'_i, a_i)p(o_i^\sigma)}{p(o_i^d, o_i^\sigma|a_i, b_i)} \sum_{\bar{s}_i \in (S_0 \times S_i)} p(\bar{s}'_i|\bar{s}_i, a_i)b_i(\bar{s}_i), \tag{7}$$

where $p(o_i^d, o_i^\sigma|a_i, b_i)$ is a normalizing constant. The probability that $b_i^{a_i o_i}$ will be the successor of b_i when agent i executes action a_i is defined as

$$p(b_i^{a_i o_i}|b_i, a_i) = p(o_i|b_i, a_i) \tag{8}$$
$$= \sum_{\bar{s}_i \in (S_0 \times S_i)} p(o_i|\bar{s}_i, a_i)b_i(\bar{s}_i). \tag{9}$$

Since individual beliefs can be defined through (7), local policies π_i can be defined as in a standard POMDP setting, i.e., as mappings $\pi_i : \Delta(S_0 \times S_i) \to A_i$ from local beliefs b_i to actions. The only term not yet defined in (7) is $p(o_i^\sigma)$: the probability that agent i receives a message σ_j and interprets it as observation o_i^σ. It is defined by two factors: the probability that agent i receives σ_j as o_i^σ, as defined by the channel noise model $p(o_i^\sigma|\sigma_j)$, times the probability that agent j actually sends σ_j. The latter probability is estimated by computing the set of possible beliefs $\{b_{j,t}^\sigma\}$ in which agent j sends σ_j according to π_j, and summing the probabilities $p(b_{j,t})$ that any of those beliefs in $b_{j,t} \in \{b_{j,t}^\sigma\}$ will occur:

$$p_{t+1}(o_i^\sigma|\pi_j) = p(o_i^\sigma|\sigma_j) \sum_{b_{j,t} \in \{b_{j,t}^\sigma\}} p(b_{j,t}), \text{ with} \tag{10}$$

$$\{b_{j,t}^\sigma\} = \{b_{j,t}|(a_j^d, a_j^\sigma) = \pi_j(b_{j,t}), a_j^\sigma = \sigma_j\}. \tag{11}$$

where $p(b_{j,t})$ is the probability that agent j is in belief $b_{j,t}$ at time t. The following equation computes $p(b_{j,t})$ as a recursion from time step 0, at which we know agent j is in its initial belief b_j^0 with probability 1:

$$p(b_{j,t}) = \begin{cases} 1 & \text{if } t = 0 \\ p(b_{j,t}|b_{j,t-1}, \pi_j(b_{j,t-1}))p(b_{j,t-1}) & \text{otherwise} \end{cases} \tag{12}$$

which uses (9). The set $\{b_{j,t}\}$ of all possible beliefs which agent j could believe to be true at a particular time step is finite for any given t, as we have assumed finite models.

3.3 Communication

The previous section defined how each agent updates its belief and how j's policy affects i's belief update, and here we provide the intuition how this implements a communication channel. For a particular incoming message σ_j agent i can use its knowledge of π_j and the set of beliefs $\{b_{j,t}\}$ for agent j to compute the set $\{b_{j,t}^\sigma\}$ (11) from which agent j would send the particular message σ_j. In this way, receiving message σ_j at time $t+1$ informs agent i about agent j's local state at time t:

$$p_{t+1}^{\sigma_j}(\bar{s}_j) = \sum_{b_{j,t} \in \{b_{j,t}^\sigma\}} b_{j,t}(\bar{s}_j)p(b_{j,t}), \tag{13}$$

combining (12) and (11). The crucial point is that, as agent j's local state $\bar{s}_j$ includes S_0, receiving message σ_j will provide agent i with information about s_0:

$$p_{t+1}^{\sigma_j}(s_0) = \sum_{s_j \in S_j} p_{t+1}^{\sigma_j}(\bar{s}_j) \text{ with } \bar{s}_j = (s_0, s_j). \tag{14}$$

In summary, the fact that s_0 is shared by all agents allows for communication-based learning, by enabling an agent j to modify the observation model of some other agent i, via the term $p(o_i^\sigma)$ in (7).

Depending on the quality of agent j's communication policy, communication can improve team performance. For instance, if agent j communicates uniformly at random, agent i will not benefit from communication as the information content of the message is zero. However, in the Multiagent Heaven or Hell example described above, if agent j communicates $\sigma_j = 1$ if it knows heaven is left, communicates $\sigma_j = 2$ if it knows heaven is right, and does not communicate otherwise, then agent i can use σ_j to update its belief about the location of heaven in a useful manner and proceed to heaven without visiting the priest.

3.4 Discussion

When comparing with other recent approaches to planning for cooperative multiagent systems in partially observable environments, there are two main advantages to the proposed model. First of all, we treat communication as an integral part of an agent's reasoning, not as an add-on. In particular, we do not define a priori semantics for a message, but treat it as part of the optimization problem. The meaning of a particular message σ_i is defined by the situations in which agent i will send σ_i. Of course, assuming a message set Σ_i with user-defined, high quality and fixed semantics simplifies the optimization problem, but we believe it is worthwhile to explore optimizing communication as part of (approximately) solving the DEC-POMDP.

The second feature of the proposed model is that it is decentralized, and does not require an (approximate) solution to the large single-agent POMDP resulting from truly free communication (as described in Sec. 2.3). The centralized state, action and observation spaces grow exponentially in the number of agents, rendering even approximate solutions infeasible. In our decentralized model, only the observation set for each agent grows with the number of agents, which can be limited by manipulating the size of possible messages Σ_i. One other possible solution to combat this exponential growth would be to assume that messages from different agents are conditionally independent given the state, which would allow us to factorize each agent's observation model. A factorized observation model induces a particular structure in an agent's POMDP (see next section), which can facilitate solving the POMDP.

We trade off exploiting an (approximately) optimal centralized solution for a potentially more scalable decentralized setting, in which each agent only considers it own local state (and some shared uncontrollable state features). Furthermore, a decentralized framework is more natural for modeling agents who cannot observe each other's state nor have truly free communication available.

4. SOLUTION METHOD

After proposing a decentralized model for planning for a team of communicating agents, we will now present an approach for computing a plan in our model. We propose to compute a joint policy for the team of agents in an iterative fashion [15, 5, 3]. We start with a (randomly selected) policy π_i for each agent i. We fix the policy for all agents except for one, our protagonist agent i. From i's perspective we can treat the other agents as part of the environment, and i can build a POMDP model of this environment, given that it knows the policies of the other agents. For the resulting POMDP model it computes a policy using a standard (approximate) POMDP solver which replaces its current policy π_i. We then move on to the next agent and repeat the procedure from its perspective. This process is iterated until some convergence criterion is met, at which time the joint policy will have reached a local maximum. Multiple random restarts are necessary in general to reach good performance. We will solve the POMDP models using PERSEUS [24]. It computes a policy in the form of a value function $V_i : \Delta(S_0 \times S_i) \rightarrow \mathbb{R}$, which for every belief estimates the amount of future discounted reward the agent can obtain.

4.1 Computing an agent's POMDP model

A crucial component is building the POMDP model from the known problem description and the policies of all teammates. In this POMDP model the reward function for agent i should not only consist of its local reward function R_i but also the expected contribution of the joint reward function R. For instance, in the Multiagent Heaven or Hell example described in Sec. 5.1, if agent i predicts that agent j's policy directs j to the heaven location, agent i's local reward function should report that if i goes to heaven too, it can receive the joint reward of r_h. We summarize the influence of the other agent by computing two statistics: $p(a_j|\bar{s}_j)$ is the probability agent j will execute a_j in $\bar{s}_j$, and $p(\bar{s}_j)$ is the probability that agent j will be in a particular state $\bar{s}_j$. We approximate these probabilities by simulating the belief tree for the team, given the pair of policies $\{\pi_i, \pi_j\}$. The root of the belief tree at $t = 0$ is the starting belief b_i^0 of each agent i. We compute the set of possible $t + 1$ beliefs for each agent by propagating each of its beliefs at time t:

$$\{b_{i,t+1}\} = \{b_{i,t}^{a_i o_i} | a_i = \pi_i(b_{i,t}), o_i \in O_i\}. \tag{15}$$

In order to keep the memory requirements in check in general pruning will be necessary. From the belief sets at time step $\{b_{i,t}\}$ and the probability that each of the beliefs will be encountered (according to (12)) we can approximate the required statistics $p(a_j|\bar{s}_j)$ and $p(\bar{s}_j)$.

We use these statistics to define the local reward function R_{i,π_j} for agent i, which takes into account the expected contribution from agent j's policy π_j:

$$R_{i,\pi_j}(\bar{s}_i, a_i) = R_i(\bar{s}_i, a_i) + \sum_{\bar{s}_j \in (S_0 \times S_j), a_j \in A_j} R(s_0, s_i, s_j, a_i, a_j) p(a_j|\bar{s}_j) p(\bar{s}_j), \tag{16}$$

with $(s_0, s_i) = \bar{s}_i$. From the same statistics the observation model for agent i can be computed, which will encode the information contained in incoming messages from agent j, as we described in Sec. 3.3. To complete the POMDP model, the transition model for agent i is given by

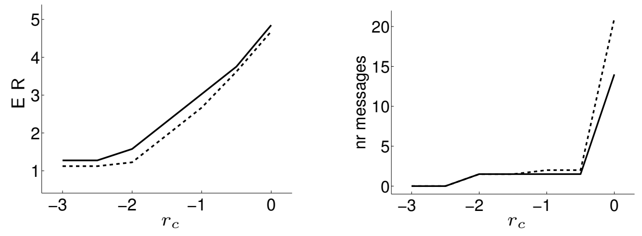
$$p(\bar{s}_i'|\bar{s}_i, a_i) = p(s_0', s_i'|s_0, s_i, a_i^d, a_i^\sigma) \tag{17}$$

$$= p(s_0'|s_0) p(s_i'|s_i, a_i^d). \tag{18}$$

4.2 Learning to communicate

The iterative scheme described above is able to react to any incoming communications, which are incorporated in the local reward model and exploited by the POMDP solver. However, it will not learn from scratch to send any messages, as the other agent's policy has not yet learned how it can benefit from incoming messages. Therefore we begin by equipping each agent with a fixed heuristic communication policy and iteratively optimize the agents' domain level component of their policies to exploit the incoming messages. Next we aim to optimize the communication component of each agent's policy with respect to the communication reward $r_c < 0$. In particular, initially all agents could at each time step send the last domain level observation they received, and we could improve team performance by only sending messages that are useful.

A message is useful when it conveys some information which the other agent can use to improve team performance. As the meaning of a message is defined over s_0, the entropy of $p^{\sigma_j}(s_0)$ (14) measures the information content of a particular message. However, if a message has low entropy does not necessarily mean it is useful to send. One proposal for testing its value is to modify each agent's local reward model (16) to prefer communicating low entropy messages to not sending them, but only at states in which it would send the messages under the initial heuristic communication policy. This ensures the other agent will know how to respond to the incoming message. Next we repeat the process of iterative optimization, allowing the agents to adjust to the new communication behavior.



(a) Reward. (b) Nr. of messages.

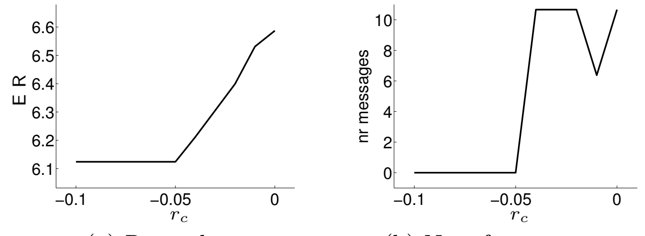
Figure 2: Multiagent Heaven or Hell results. Solid lines indicate results for 2 agents, and dashed lines show results for a team of 3 agents. Plot (a) shows the expected discounted reward (E R) obtained for different communication rewards (r_c), and figure (b) plots the average number of messages sent.

5. EXPERIMENTS

We will present experimental results in the Multiagent Heaven or Hell domain and in a game of tag, in which agents have to cooperate in order to tag an opponent. In these experiments we included the time step of the system in s_0, which allows for a single message to have a different meaning, depending on which time step it was sent. Adding time to s_0 requires no extra assumptions, but does allow the agents to more closely correlate their behavior. Providing correlation for agents in a DEC-POMDP has been shown to improve team performance [3].

5.1 Multiagent Heaven or Hell

First we will consider the Multiagent Heaven or Hell domain as introduced in Sec. 3.1. The environment contains seven grid cells ($|S_i| = 7$) and the shared state s_0 is the cross-product of the time index and {"heaven is in bottom-left grid cell","heaven is in bottom-right grid cell"}, where we set the horizon of the problem to 15. Each agent has a domain level action set $A_i^d = \{$*north, east, south, west,*



(a) Reward. (b) Nr. of messages.

Figure 3: Pack Tag results. Plot (a) shows the expected discounted reward (E R) obtained for different communication rewards (r_c), and figure (b) plots the average number of messages sent.

stay in place}, where the first four are movement actions that transport the agent one grid cell in the corresponding direction, and the last action is a no-op which has no effect on the agent's location. Each agent can observe whether there is a wall to the left or to the right of it, and in the priest state it can observe the location of heaven, resulting in six possible observations O_i^d ={*left, right, both, neither, heaven-left, heaven-right*}. Meeting in heaven by all agents is rewarded by $r_h = 10$ and meeting in hell by $-r_h$. Visiting the priest incurs a negative reward $r_p = -2$, every movement action a reward of -0.1, and the discount factor γ is set to 0.95. Both the transition and observation model are deterministic, and the agents always start in the center grid cell (the cell at the T-intersection).

Fig. 2 shows results obtained by using the local reward modification scheme described in Sec. 4.2, for teams of 2 and 3 agents. First, we see in Fig. 2(a) that the team performance increases when the penalty for communication decreases, i.e., r_c becomes less negative. In all cases one agent i will visit the priest, but it will only send a message indicating the location of heaven if the communication cost is low ($r_c \geq -2$), see Fig. 2(b). Otherwise the other agent(s) will go to one of the heaven/hell locations, and agent i will only join them if it is indeed the correct location. When communication is free, performance is best and a large number of messages are communicated, as one would expect. On the other extreme, when communication is expensive ($r_c = -3, -2.5$), the performance remains stable.

5.2 Pack Tag

The "Pack Tag" problem models a game of tag, in which two robots have to tag an opponent robot [19, 5]. The game is played on a $k \times k$ grid, with one robot starting in the bottom-left corner and the other one in the top-right corner. At the beginning of each game the opponent is positioned uniformly at random in one of the grid cells and remains stationary. The two robots have to occupy the same grid cell as the opponent and execute the *tag* action simultaneously in order to tag the opponent. The robots can move in any of the four compass directions, and their actions are deterministic. The two robots have a sensor which detects the opponent with full certainty when the opponent is sharing the same grid cell. Each robot receives a null observation otherwise, but remains perfectly localized due to the known starting position and noise-free motion. Each action incurs a reward of -0.1 and successfully tagging the opponent results in a reward of 10. The agents have to tag the opponent in 10 time steps, and γ was set to 0.95.

As in Sec. 5.1 we varied the communication cost while employing the local reward modification method, and using a grid size $k = 3$. Fig. 3 shows that when the communication cost is high, $r_c \leq -0.05$, the agents converge to a non-communicating policy pair. They both search the grid, and when one of them sees the opponent it waits for the other to join it before jointly tagging it. However, when communication is cheaper the agents communicate when they observe the opponents, which indicates to the other agent the location of the opponent. Here the benefit of including time in s_0 is manifested, as receiving the same message at a different time step has a different meaning: agent i knows agent j's policy, and therefore knows j's search pattern, and in this way knows in what grid cell agent j observed the opponent when j send its message. After receiving an informative message an agent abandons its search pattern and moves directly to its teammate's position, where they jointly tag the opponent. The policy pair computed for $r_c = -0.01$ has converged to better local maximum, which uses a more efficient form of communication.

6. CONCLUSIONS

In this paper we studied the problem of computing a plan for a team of agents. We considered a general problem setting, in which agents inhabit a stochastic environment that is only partially observable to them. The agents have the capability to communicate, but sending messages has a certain cost and the available bandwidth is limited. Decentralized partially observable Markov decision processes (DEC-POMDPs) formalize such multiagent planning problems for groups of cooperating agents. We presented a decentralized model for tackling these kinds of planning problems, which incorporates a communication channel. Decentralized models potentially scale better and are a more natural paradigm for modeling agents that cannot observe each other's state. Contrary to other recent approaches, we treat communication as an integral part of the model, and model it directly in the DEC-POMDP. We proposed an approximate iterative method for computing policies in the proposed model, and obtained encouraging preliminary experimental results in two domains.

As future work we would like to examine possibilities of simultaneous learning of communication-derived POMDPs for two or more agents, which could potentially improve upon the alternating-maximization paradigm. Furthermore, we would like to experiment on problems with more agents, which are too large to be solved by centralized methods. For problems that are still solvable by centralized algorithms, we intend to compare the centralized solution to the decentralized one, to gain more insight in how optimality is traded off for more efficient computation.

Acknowledgments

M.T.J. Spaan and N. Vlassis are supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414. G.J. Gordon's work was supported in part by NSF grant EF-0331657.

7. REFERENCES

[1] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

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

[3] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.

[4] M. B. Dias, A. Stentz, and D. Goldberg. Market-based multirobot coordination for complex space applications. In *Proc. of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.

[5] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[6] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Game theoretic control for robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[7] M. S. Fox, M. Barbuceanu, and R. Teigen. Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems*, 12(2,3), 2000.

[8] C. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[9] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. In *Journal of Artificial Intelligence Research*, volume 22, pages 143–174, 2004.

[10] A. Guo and V. Lesser. Planning for weakly-coupled partially observable stochastic games. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.

[11] E. Hansen, D. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proc. of the National Conference on Artificial Intelligence*, San Jose, 2004.

[12] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[13] V. Lesser, C. L. Ortiz, and M. Tambe, editors. *Distributed sensor networks: A multiagent perspective*. Kluwer Academic Publishers, 2003.

[14] R. Nair, M. Tambe, M. Roth, and M. Yokoo. Communication for improving policy computation in distributed POMDPs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[15] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Acapulco, Mexico, Aug. 2003.

[16] J. Ooi and G. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proc. of the 35th Conference on Decision and Control*, pages 293–298, 1996.

[17] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[18] S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2005.

[19] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Acapulco, Mexico, Aug. 2003.

[20] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[21] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[22] M. Roth, R. Simmons, and M. Veloso. Decentralized communication strategies for coordinated multi-agent policies. In A. Schultz, L. Parker, and F. Schneider, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume IV. Kluwer Avademic Publishers, 2005.

[23] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, 2004.

[24] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[25] D. Szer and F. Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *European Conference on Machine Learning*, 2005.

[26] S. Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[27] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proc. of the Fifth Int. Conference on Autonomous Agents*, 2001.