Tree-based Solution Methods for Multiagent POMDPs with Delayed Communication

Frans A. Oliehoek
MIT CSAIL / Maastricht University
Maastricht, The Netherlands
frans.oliehoek@maastrichtuniversity.nl

Matthijs T.J. Spaan
Delft University of Technology
Delft, The Netherlands
m.t.j.spaan@tudelft.nl

ABSTRACT

Planning under uncertainty is an important and challenging problem in multiagent systems. Multiagent Partially Observable Markov Decision Processes (MPOMDPs) provide a powerful framework for optimal decision making under the assumption of instantaneous communication. We focus on a delayed communication setting (MPOMDP-DC), in which broadcasted information is delayed by at most one time step. This model allows agents to act on their most recent (private) observation. Such an assumption is a strict generalization over having agents wait until the global information is available and is more appropriate for applications in which response time is critical. From a technical point of view, MPOMDP-DCs are quite similar to MPOMDPs. However, value function backups are significantly more costly, and naive application of incremental pruning, the core of many state-of-the-art optimal POMDP techniques, is intractable. In this paper, we show how to overcome this problem by demonstrating that computation of the MPOMDP-DC backup can be structured as a tree and introducing two novel *tree-based pruning* techniques that exploit this structure in an effective way. We experimentally show that these methods have the potential to outperform naive incremental pruning by orders of magnitude, allowing for the solution of larger problems.

1. INTRODUCTION

This paper focuses on computing policies for multiagent systems (MAS) that operate in stochastic environments and that share their individual observations with a one step delay. While dynamic programming algorithms date back to the seventies [29, 3, 10, 12], computational difficulties have limited the model's applicability. In particular, the backup operator under delayed communication has an additional source of complexity when compared to settings with instantaneous communication. In this paper, we take an important step in overcoming these challenges by showing how this additional complexity can be mitigated effectively.

Our efforts are part of the greater agenda of multiagent planning under uncertainty. The task faced by a team of agents is complicated by partial or uncertain information about the world, as well as stochastic actions and noisy sensors. Especially settings in which agents have to act based on their local information only have received a large amount of attention in the last decade [22, 4, 25, 30]. The Dec-POMDP framework [5] can be used to formulate such problems, but decentralization comes at a high computational cost: optimally solving a Dec-POMDP is NEXP-complete.

Communication can be used to mitigate the problem of decentralized information: when agents can share their local observations, each agent can reason about the optimal joint action given this global information state, called *joint belief*, and perform its individual component of this joint action. That is, when instantaneous communication is available, it allows one to reduce the problem to a special type of POMDP [22], called a *Multiagent POMDP (MPOMDP)*, which has a lower computational complexity than a Dec-POMDP (PSPACE-complete) and will generally lead to a joint policy of a higher quality.

However, assuming instantaneous communication is often unrealistic. In many settings communication channels are noisy and synchronization of the information states takes considerable time. Since the agents cannot select their actions without the global information state (called 'joint belief'), this can cause unacceptable delays in action selection. One solution is to assume that synchronization will be completed within k time steps, selecting actions based on the last known joint belief. Effectively, this reduces the problem to a centralized POMDP with delayed observations [2]. However, such formulations are unsuitable for tasks that require a high responsiveness to certain local observations, such as applications for collission avoidance, ambient intelligence or other more general forms of human-computer interaction and distributed surveillance. Another prime example is decentralized protection control in electricity distribution networks by so-called Intelligent Electronic Devices (IED). Such IEDs not only decide based on locally available sensor readings, but can receive information from other IEDs through a communication network with deterministic delays [32]. When extreme faults such as circuit or cable failures occur, however, no time can be wasted waiting for information from other IEDs to arrive.

We therefore consider an alternative execution model that assumes that while communication is received with a delay of one stage, the agents can act based on their private observation as well as the global information of the last stage.[1] That is, all agents broadcast their individual observations, but in contrast to a delayed observation POMDP they do not wait with action selection until this communication phase has ended. Instead, they go ahead taking an action based

The Seventh Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (*MSDM-2012*), held in conjunction with *AAMAS*, June 2012, in Valencia, Spain.

[1]This is also referred to as a *one step delayed sharing pattern* in the decentralized control literature.

on the global information of the previous stage and the individual observation of the current stage. This results in a strict generalization of the delayed observation POMDP.

Thereby, this model uses communication to mitigate the computational complexity and achieve higher performance than Dec-POMDPs, while preventing delays in action selection. Moreover, the optimal solution of the MPOMDP-DC is useful in settings with longer communication delays [27], can be used as a heuristic in Dec-POMDPs [26], or to upper bound the expected value of MASs without communication (e.g., in the setting of communication channels) [10, 20].

From a technical perspective, the MPOMDP is equivalent to a POMDP with a centralized controller, which means that all POMDP solution methods apply. Many of these exploit the fact that the value function is piecewise-linear and convex (PWLC) over the joint belief space. Interestingly, the value function of an MPOMDP-DC exhibits the same property [12]. As such one would expect the same computational methods to be applicable. However, *incremental pruning (IP)* [6], that performs a key operation, the so-called *cross-sum*, more efficiently, is not directly able to achieve the same improvements for MPOMDP-DCs. A problem is the need to loop over a number of decision rules that is exponential both in the number of agents and in the number of observations. This means that the backup operator for MPOMDP-DC model is burdened with an additional source of complexity when compared to the regular POMDP backup.

In this paper, we target this additional complexity by proposing two novel methods that operate over a tree structure. The first method, called TBP-M for tree-based pruning with memoization, avoids *duplicate* work by caching the result of computations at internal nodes and thus accelerates computation at the cost of memory. The second, branch and bound (TBP-BB), is able to avoid *unnecessary* computation by making use of upper and lower bounds to prune parts of the tree. Therefore it provides a different space/time tradeoff. The empirical evaluation of our proposed methods on a number of test problems shows a clear improvement of a naive application of incremental pruning. TBP-M provides speedups of up to 3 orders of magnitude. TBP-BB does not consistently outperform the baseline, but is still able to provide large speedups on a number of test problems, while using little memory.

2. BACKGROUND

Here we present background on the relevant formal models and their solutions methods.

2.1 Models

In this section we formally introduce the multiagent POMDP (MPOMDP) model and describe the planning problem.

Definition 1. A *multiagent partially observable Markov decision process* $\mathcal{M} = \langle n,\mathcal{S},\mathcal{A},P,R,\mathcal{O},O,h,OC\rangle$ consists of

- a finite set of n agents;
- $\mathcal{S}$ is a finite set of states;
- $\mathcal{A} = \times_i \mathcal{A}_i$ is the set $\{a^1,\ldots,a^J\}$ of J joint actions. $\mathcal{A}_i$ is the set of actions available to agent i. Every time step one $a = \langle a_1,...,a_n\rangle$ is taken;
- P, the transition function. $P^a(s'|s)$ is the probability of transferring from s to s' under a;
- R is the reward function. We write $R^a(s)$ for the reward accumulated when taking a from s;
- $\mathcal{O} = \times_i \mathcal{O}_i$ is the set $\{o^1 \ldots o^K\}$ of K joint observations. Every stage an $o = \langle o_1,...,o_n\rangle$ is observed;
- O is the observation function. We write $O^a(o|s)$ for the probability of o after taking a and ending up in s;
- h is the horizon, the number of time steps or *stages* that are considered when planning.

The special case with 1 agent is called a (regular) POMDP. Execution in an MPOMDP is as follows. At every stage t, each agent i:

1. observes its individual o_i,
2. broadcasts its own observation o_i,
3. receives observations $o_{-i} = \langle o_1,\ldots,o_{i-1},o_{i+1},\ldots,o_n\rangle$ from the other agents,
4. uses the joint observation $o^t = \langle o_i,o_{-i}\rangle$ and previous joint action a^{t-1} to update the new joint belief $b^t = BU(b^{t-1},a^{t-1},o^t)$,
5. looks up the joint action for this stage in the joint policy $a^t \leftarrow \pi(b^t)$,
6. and executes its component a_i^t.

The belief update function BU updates the previous joint belief using Bayes rule:

$$b'(s') = BU(b,a,o) = \frac{1}{P^a(o|b)} O^a(o|s') \sum_s P^a(s'|s) b(s),$$

where $P^a(o|b)$ is a normalizing factor. We denote the set of all joint beliefs by $\mathcal{B}$. In the remainder of this paper, we will refer to a joint belief simply as 'belief'.

The joint policy $\pi = \left(\delta^0,\delta^1,\ldots,\delta^{h-1}\right)$ is a sequence of joint decision rules mapping beliefs to joint actions. The goal of the multiagent planning problem for an MPOMDP is to find an optimal joint policy π^* that maximizes the total expected sum of rewards, defined an arbitrary π as

$$V(\pi) = \mathbf{E}\Big[\sum_{t=0}^{h} R(s^t,a^t) \mid \pi,b^0\Big], \qquad (1)$$

where b^0 is the initial state distribution. In this paper we will consider planning over a finite horizon h. While (1) optimizes for a given b^0, we will be concerned with computation of all the optimal π for all possible b^0, which is beneficial for tasks in which b^0 is not known in advance.

The value can be expressed recursively as a function of the joint belief space. In particular, the $h-t$ steps-to-go *action-value* of b is

$$Q^t(b,a) = R^a_{\mathcal{B}}(b) + \sum_o P^a(o|b) \max_{a'} Q^{t+1}(b',a'), \qquad (2)$$

where $R^a_{\mathcal{B}}(b) = \sum_s R^a(s)b(s)$ and $b' = BU(b,a,o)$. An optimal joint decision rule for stage t, δ^{t*}, selects the maximizing a for each b and thereby defines the *value function*:

$$V^t(b) = \max_a Q^t(b,a) = Q^t(b,\delta^{t*}(b)).$$

When no communication is available, the multiagent planning problem can be formalized as a Dec-POMDP. This problem is significantly different as agents must base their decisions solely on local observations and have no means to compute the joint belief resulting in NEXP-complete complexity.

In the remainder of this paper we focus on settings with delayed communication.

Definition 2. An *MPOMDP with delayed communication (MPOMDP-DC)* is an MPOMDP where communication is received with a one-step delay.

Execution in an MPOMDP-DC is as follows. At decision point t, each agent i:

1. has received the *previous-stage* observations o_{-i}^{t-1} and actions a_{-i}^{t-1} of the other agents,
2. observes its individual o_i^t,
3. computes $b^{t-1} = BU(b^{t-2},a^{t-2},o^{t-1})$, using the *previous* joint observation $o^{t-1} = \langle o_i^{t-1},o_{-i}^{t-1}\rangle$ and joint action a^{t-2} (remembered from stage $t-2$ when it received a_{-i}^{t-2}),
4. looks up the individual action for this stage in the individual policy $a_i^t \leftarrow \pi_i(b^{t-1},a^{t-1},o_i^t)$,
5. broadcasts its own observations o_i^t and action a_i^t,[2]
6. and executes a_i^t.

From this it is clear that there are quite a few differences with the MPOMDP formulation. Most notably, the form of the joint policy is different. In an MPOMDP-DC a joint decision rule specifies an individual decision rule for each agent $\delta^t = \langle \delta_1^t,\ldots,\delta_n^t\rangle$, where each $\delta_i^t : \mathcal{B}^{t-1} \times \mathcal{A} \times \mathcal{O}_i \to \mathcal{A}_i$ is a mapping from a $\langle b^{t-1},a^{t-1},o_i^t\rangle$-tuple to an individual action a_i^t.

As such, it may come as a surprise that we can still define the optimal value of an MPOMDP-DC as a function of joint beliefs:

$$Q^t(b,a) = R_{\mathcal{B}}^a(b) + \max_{\beta\in B} \sum_o P^a(o|b) Q^{t+1}(b',\beta(o)), \qquad (3)$$

where B is the set of decentralized control laws $\beta = \langle \beta_1,\ldots,\beta_n\rangle$ which the agents use to map their individual observations to actions: $\beta(o) = \langle \beta_1(o_1),\ldots,\beta_n(o_n)\rangle$. Essentially we have decomposed a joint decision rule δ^t into a collection of β, one for each $\langle b,a\rangle$-pair. In fact, the maximization that (3) performs for each such $\langle b,a\rangle$-pair corresponds to solving a collaborative Bayesian game [17] or, equivalently, a Team Decision Problem and is NP-complete [28]. We also note that the set of possible β is a strict super set of the set of joint actions; control laws that ignore the private observation and just map from the previous joint belief (and joint action) to a joint action are included. As such, this approach gives a strict improvement over assuming delayed joint observations [2].

2.2 Solution Methods

For both instantaneous and delayed communication, we can define a value function over joint beliefs. The problem with computing them, however, is that these spaces are continuous. Here we discuss methods that overcome this problem. We focus on the case of MPOMDPs and defer the necessary extensions for MPOMDP-DC to the next section.

Many methods for solving a (multiagent) POMDP make use of the fact that (2) is piecewise-linear and convex (PWLC) over the belief space. That is, the value at stage t can be expressed as a maximum inner product with a set of vectors:

$$Q^t(b,a) = \max_{v_a\in\mathcal{V}_a^t} b\cdot v_a = \max_{v_a\in\mathcal{V}_a^t} \sum_s b(s) v_a(s). \qquad (4)$$

[2]Strictly speaking each agent can compute a_{-i} given the common information, and broadcasting a_i is not essential.

We will also write $\mathcal{V}^t = \bigcup_{a\in\mathcal{A}} \mathcal{V}_a^t$ for the complete set of vectors that represent the value function. Each of these vectors v_a represents a conditional plan (i.e., policy) starting at stage t with joint action a and has the following form:

$$v_a^t = R^a + \sum_o g_{ao}^i, \qquad (5)$$

with g_{ao}^i the back-projection of v^i, the i-th vector in $\mathcal{V}^{t+1}$:

$$g_{ao}^i = \sum_{s'} O^a(o|s') P^a(s'|s) v^i(s'). \qquad (6)$$

The set of such *gamma vectors* is denoted $\mathcal{G}_{ao}$.

Monahan's algorithm [16] simply generates all possible vectors by, for each joint action a, for each possible observation selecting each possible next-stage vector:

$$\mathcal{V}_a^t = \{R^a\} \oplus \mathcal{G}_{ao^1} \oplus \cdots \oplus \mathcal{G}_{ao^K}, \qquad (7)$$

where the cross-sum $A \oplus B = \{a+b \mid a\in A, b\in B\}$.

A problem in this approach is that the number of vectors generated by it grows exponentially; at every backup, the algorithm generates $\left|\mathcal{V}^{t+1}\right| = J\left|\mathcal{V}^t\right|^K$ vectors. However, many of these vectors are *dominated,* which mean that they do not maximize any point in the belief space. Formally, a vector $v\in\mathcal{V}^t$ is dominated if

$$\nexists b \text{ s.t. } \quad b\cdot v > b\cdot v', \quad \forall v'\in\mathcal{V}^t.$$

The operation `Prune` removes all dominated vectors by solving a set of linear programs [6, 8]. That way, the *parsimonious* representation of $\mathcal{V}^t$ can be computed via

$$\mathcal{V}^t = \texttt{Prune}(\mathcal{V}_{a^1}^t \cup \cdots \cup \mathcal{V}_{a^J}^t), \qquad (8)$$

$$\mathcal{V}_a^t = \texttt{Prune}\left(\{R^a\} \oplus \mathcal{V}_{ao^1}^t \oplus \cdots \oplus \mathcal{V}_{ao^K}^t\right) \qquad (9)$$

$$\mathcal{V}_{ao}^t = \texttt{Prune}(\mathcal{G}_{ao}). \qquad (10)$$

The technique called *incremental pruning* (IP) [6] speeds up the computation of the value function tremendously by realizing that (9), the bottleneck in this computation, can be re-written to interleave pruning and cross-sums:

$$\texttt{Prune}(\mathcal{V}_{ao^1}^t \oplus \cdots \oplus \mathcal{V}_{ao^K}^t) = \texttt{Prune}((\ldots.\texttt{Prune}(\mathcal{V}_{ao^1}^t \oplus \mathcal{V}_{ao^2}^t)\ldots) \oplus \mathcal{V}_{ao^K}^t). \qquad (11)$$

That is, we can first prune parts of the larger cross sum to yield the same result.

3. COMPUTING DC VALUE FUNCTIONS

In this section, we show how the methods mentioned in the previous section can be extended to the MPOMDP-DC.

3.1 Vector Representations

As for an MPOMDP, we can represent the value function under delayed communication using vectors [12]. However, in the MPOMDP-DC case, not all combinations of next-stage vectors are possible; the actions they specify should be consistent with an admissible decentralized control law β. That is, we define vectors $g_{aoa'} \in \mathcal{G}_{aoa'}$ analogously to (6), but now restricting v^i to be chosen from $\mathcal{V}_{a'}^{t+1}$. From these we construct

$$\begin{aligned}\mathcal{V}_a^t &= \{R^a + g_{ao^1\beta(o^1)} + g_{ao^2\beta(o^2)} + \cdots + g_{ao^k\beta(o^k)} \\ &\qquad \mid \forall\beta, \forall g_{ao\beta(o)} \in \mathcal{G}_{ao\beta(o)}\} \\ &= \{R^a\} \oplus \mathcal{G}_{ao^1\beta(o^1)} \oplus \cdots \oplus \mathcal{G}_{ao^K\beta(o^K)} \qquad (12)\end{aligned}$$

Note that it is no longer possible to collect all the vectors in one set $\mathcal{V}^t$, since we will always need to discriminate which joint action a vector specifies.

In the following, we will also represent a β as a vector of joint actions $\langle a_{(1)} \dots a_{(K)} \rangle$, where $a_{(k)}$ denotes the joint action selected for the k-th joint observation.

PROPOSITION 1. *The (not pruned) set $\mathcal{V}^t_{a,DC}$ of vectors under delayed communication is a strict subset of the set $\mathcal{V}^t_{a,P}$ of MPOMDP vectors: $\forall a \quad \mathcal{V}^t_{a,DC} \subset \mathcal{V}^t_{a,P}$.*

PROOF. To see this, realize that

$$\mathcal{G}_{ao} = \bigcup_{a'} \mathcal{G}_{aoa'} \tag{13}$$

and that therefore (7) can be rewritten as

$$\begin{aligned} \mathcal{V}^t_{a,P} &= \bigcup_a (R^a \oplus \Big[\bigcup_{a'} \mathcal{G}_{ao^1 a'}\Big] \oplus \cdots \oplus \Big[\bigcup_{a'} \mathcal{G}_{ao^K a'}\Big]) \\ &= \bigcup_a \bigcup_{\langle a_{(1)}, \dots a_{(K)} \rangle \in \mathcal{A}^K} (R^a \oplus \mathcal{G}_{ao^1 a_{(1)}} \oplus \cdots \oplus \mathcal{G}_{ao^k a_{(k)}}). \end{aligned}$$

The observation follows from the fact that the set of admissible $\beta \in B$ is a subset of $\mathcal{A}^K$: each β can be represented as a vector of joint actions $\langle a_{(1)} \dots a_{(K)} \rangle$, but not every such vector is a valid β. □

This means that the number of vectors grows less fast when performing exhaustive generation. However, an effective method for doing the backup, such as incremental pruning for POMDPs, has not been developed.

3.2 Naive Incremental Pruning

An obvious approach to incremental pruning in MPOMDP-DCs is given by the following equations, which we will refer to as NAIVE IP:

$$\mathcal{V}^t_a = \texttt{Prune}(\bigcup_{\beta \in B} \mathcal{V}^t_{a\beta}), \tag{14}$$

$$\mathcal{V}^t_{a\beta} = \texttt{Prune}\big(\{R^a\} \oplus \mathcal{G}^t_{ao^1\beta(o^1)} \oplus \cdots \oplus \mathcal{G}^t_{ao^K\beta(o^K)}\big), \tag{15}$$

$$\mathcal{G}^t_{aoa'} = \texttt{Prune}(\mathcal{G}_{aoa'}), \tag{16}$$

where (15) uses incremental pruning.

Note that it is not possible to prune the union over joint actions as in (8), because this would correspond to a maximization over joint actions in (3). This is not possible, because under delayed communication, the joint action is that joint action that was taken in the previous time step. In particular, the sets of vectors $\mathcal{V}^t_a$ are used as follows. At a stage t, an agent knows b^{t-1} and a^{t-1}. It uses this information to determine the vector $v \in \mathcal{V}^{t-1}_{a^{t-1}}$ that maximizes $v \cdot b^{t-1}$. It retrieves the maximizing β for v, and executes $\beta_i(o^t_i)$.

There are two problems with the computation outlined above. First, it iterates over all possible $\beta \in B$, which is exponential both in the number of agents and in the number of observations. Performing the iteration in this way in practice this means that performing one backup takes nearly a factor $|B|$ as much as a POMDP backup. Second, it performs a lot of duplicate work. E.g., there are many β that specify $\beta(o^1) = a^k$,$\beta(o^2) = a^l$, but for each of them $\texttt{Prune}(\mathcal{G}_{ao^1a^k} \oplus \mathcal{G}_{ao^2a^l})$ is recomputed.

It is tempting to side step the problem by generating the POMDP value function using incremental pruning and then throw away the vectors that do not correspond to a valid β, i.e., throw away the vectors that are not *admissible*. This, however, is not correct: many vectors corresponding to valid β may have been pruned away because they were dominated by (a combination of) other vectors that are not admissible, which clearly is not desirable.

4. TREE-BASED PRUNING

In order to overcome the drawbacks of the naive approach outline above, we propose a different approach. Rather than creating sets $\mathcal{V}^t_{a\beta}$ for each $\beta \in B$, we directly construct

$$\mathcal{V}^t_a = \texttt{Prune}(\bigcup_{\beta \in B} \big(\{R^a\} \oplus \mathcal{G}^t_{ao^1\beta(o^1)} \oplus \cdots \oplus \mathcal{G}^t_{ao^K\beta(o^K)}\big)). \tag{18}$$

As mentioned, we can interpret β as a vector of joint actions. This allows us to decompose the union over β into dependent unions over joint actions, as illustrated in Fig. 1(a).

The resulting equation (17) defines a computation tree, as illustrated in Fig. 1(b) in the context of a fictitious 2-action (x and y) 2-observation (1 and 2) MPOMDP-DC. The root of the tree, $\mathcal{V}^t_a$, is the result of the computation. There are two types of internal, or operator, nodes: cross-sum and union. All the leaf nodes are sets of vectors. An operator node n takes as input the sets from its children, computes $\mathcal{V}_n$, the set resulting from application of its operator, and propagates this result up to its parent. When a union node is the j-th union node on a path from root to leaf, we say it has depth j. A depth-j union node performs the union over $a_{(j)}$ and thus has children corresponding to different assignments of a joint action to o^j (indicated by the gray bands). It is important to realize that the options available for $a_{(j)}$ depend on the action choices $(a_{(1)}, \dots, a_{(j-1)})$ made higher up in the tree; given those earlier choices, some $a_{(j)}$ may lead to conflicting individual actions for the same individual observation. Therefore, while there are 4 children for $\cup_{a_{(1)}}$, union nodes deeper in the tree have only 2 or even just 1.

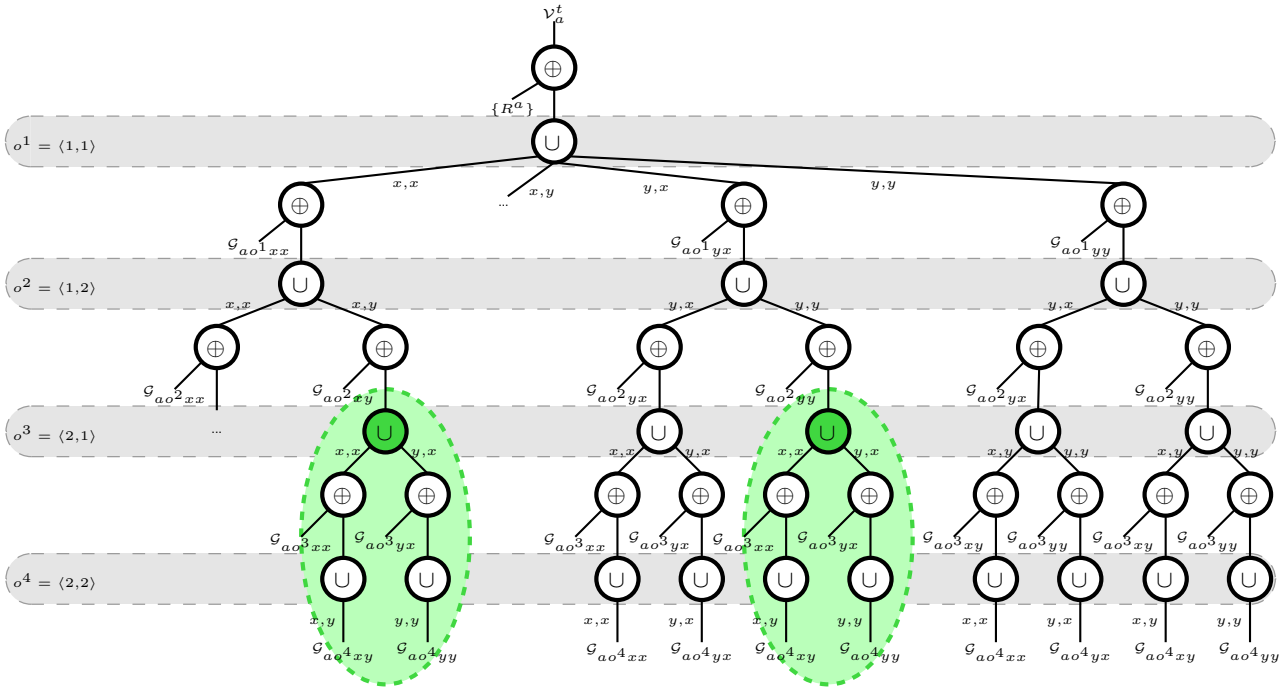
Now, to compute (18) we propose *tree-based (incremental) pruning (TBP)*: it expands the computation tree and, when the results are being propagated to the top of the tree, prunes dominated vectors at each internal node. However, Fig. 1(b) shows another important issue: there are identical sub-trees in this computation tree, as indicated by the dashed green ovals, which means that we would be doing unnecessary work. We address this problem by memoization, i.e., caching of intermediate results, and refer to the resulting method as TBP-M. Note that the sub-tree under a node is completely characterized by a specification of which joint action assignments are still possible for the unspecified joint observations. For instance, the nodes inside the ovals we can characterize as $\langle -, - \rangle, \langle *,x \rangle, \langle *,y \rangle\rangle$, where '$-$' means that the joint action for that joint observation is specified, $\langle *,x \rangle$ denotes the set $\{\langle x,x \rangle, \langle y,x \rangle\}$ ('*' acts as a wildcard) and similar for $\langle *,y \rangle$. We call such a characterization the ID string and it can be used as the key into a lookup table. This way we only have to perform the computation just once for each ID string.

5. BRANCH & BOUND

Here we introduce tree-based pruning using branch and bound (TBP-BB), which is a different method to overcome the problems of naive incremental pruning as described in Section 3.2. The motivation of introducing another method is twofold. First, while TBP-M effectively addresses the issue of performing duplicate work (i.e., it addresses the second problem in Section 3.2), it does not fully address the

$$
\begin{aligned}
\mathcal{V}_a^t &= \bigcup_{\langle a_{(1)}\cdots a_{(k)}\rangle \in B} \left(\{R^a\} \oplus \mathcal{G}_{ao^1 a_{(1)}} \oplus \cdots \oplus \mathcal{G}_{ao^k a_{(k)}} \right) \\
&= \{R^a\} \oplus \bigcup_{a_{(1)} \in A} \bigcup_{\langle a_{(2)}\cdots a_{(k)}\rangle \in B|_{a_{(1)}}} \left(\mathcal{G}_{ao^1 a_{(1)}} \oplus \cdots \oplus \mathcal{G}_{ao^k a_{(k)}} \right) \\
&= \{R^a\} \oplus \bigcup_{a_{(1)} \in A} \left[\mathcal{G}_{ao^1 a_{(1)}} \oplus \bigcup_{\langle a_{(2)}\cdots a_{(k)}\rangle \in B|_{a_{(1)}}} \left(\mathcal{G}_{ao^2 a_{(2)}} \oplus \cdots \oplus \mathcal{G}_{ao^k a_{(k)}} \right) \right] \\
&= \{R^a\} \oplus \bigcup_{a_{(1)} \in A} \left[\mathcal{G}_{ao^1 a_{(1)}} \oplus \bigcup_{a_{(2)} \in A|_{a_{(1)}}} \bigcup_{\langle a_{(3)}\cdots a_{(k)}\rangle \in B|_{a_{(1)} a_{(2)}}} \left(\mathcal{G}_{ao^2 a_{(2)}} \oplus \mathcal{G}_{ao^3 a_{(3)}} \oplus \cdots \oplus \mathcal{G}_{ao^k a_{(k)}} \right) \right] \\
&= \{R^a\} \oplus \bigcup_{a_{(1)} \in A} \left[\mathcal{G}_{ao^1 a_{(1)}} \oplus \bigcup_{a_{(2)} \in A|_{a_{(1)}}} \left[\mathcal{G}_{ao^2 a_{(2)}} \oplus \bigcup_{a_{(3)} \in A|_{a_{(1)} a_{(2)}}} \cdots \oplus \left[\bigcup_{a_{(k)} \in A|_{a_{(1)}\cdots a_{(k-1)}}} \mathcal{G}_{ao^k a_{(k)}} \right] \right] \right] \qquad (17)
\end{aligned}
$$

(a) Rewriting (18). $B|_{a_{(1)}a_{(2)}}$ denotes the set of β consistent with $a_{(1)}$,$a_{(2)}$, $A|_{a_{(1)}\cdots a_{(k-1)}}$ denotes the set of joint actions (for the k-th joint observation) that result in a valid β.



(b)

Figure 1: (a) By decomposing the joint union and moving it inwards, we create a tree-formed computation structure. (b) The computation tree of $\mathcal{V}_a^t$. See text for description.

other problem. Every leaf of a depth-K union node corresponds to exactly one β. Even though we hope to avoid visiting many leafs due to memoization, the number of β and thus number of sub-trees that will need to be computed at least once may still be prohibitive. TBP-BB can overcome this problem by pruning complete parts of the search tree, even if they have not been computed before. Second, as mentioned above, TBP-M needs to cache results in order to avoid duplicate work which may lead to memory problems. TBP-BB does not perform caching, but still can avoid expanding the entire tree, creating an attractive alternative to trade off space and time complexity.

Branch and bound (BB) is a well-known method to prune parts of *search trees*. It computes an f-value, for each visited node n in the tree via $f(n) = g(n) + h(n)$, where $g(n)$ is the actual reward achieved up to the node, and $h(n)$ is a heuristic estimate of the reward to be found in the remainder of the tree. When h is *admissible* (a guaranteed overestimation), so is f. This means that if the f-value of a node n is less than the value of the maximum lower bound l (i.e., the best full solution found so far), we can prune the sub-tree under n. Since we are not concerned with a search-tree, but rather a *computation tree*, this technique can not be applied directly. Still, we can generalize the idea of BB to be applicable to our setting, which requires specifying f, g and h as *PWLC functions* and comparing them to l, the *lower bound function*: the PWLC function over belief space induced by the set L of already found non-dominated vectors v_a^t.

This idea is illustrated in Fig. 2. The first insight that this figure illustrates is that, while the computation tree works bottom-up, we can also interpret it as top-down: by associating the null-vector $\vec{0}$ with the root node, we can now pass

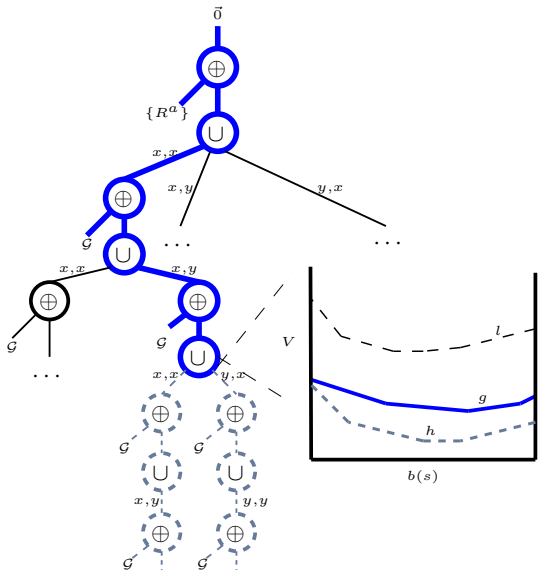


Figure 2: A part of the search tree from Fig. 1(b) illustrating the heuristics for a search node. Gamma sets are abbreviated by simply $\mathcal{G}$. If $f = g + h$ does not exceed the lower bound l (induced by already found full vectors) at any point in the belief space, the node does not have to be expanded further.

the result of $\vec{0} \oplus \{R^a\} = \{R^a\}$ down the tree. The union node now acts as a simple 'splitter' duplicating its input set down to all children. This way we can define with each node the cross-sum of sets encountered from the root to that node (indicated in thick blue lines in the figure). This cross-sum itself is a set G and thus defines a PWLC function over the belief space, namely g.

In a similar way, the PWLC heuristic function h is represented by a set of vectors H. The function should give a guaranteed overestimation of $\mathcal{V}_n$ (as formalized below) and we propose to use the POMDP vectors. That is, for a node at depth j (for which the first j joint observations' gamma vector sets are used in the cross-sum to compute G), we specify:

$$H_j = \texttt{Prune}(\mathcal{V}^t_{ao^{j+1}} \oplus \cdots \oplus \mathcal{V}^t_{ao^K}). \tag{19}$$

This can be pre-computed using incremental pruning.

With a slight abuse of notation, we will write f_n and F_n for the f-function at node n and the set of vectors representing it. We will say that f_n and F_n are *admissible* if they are a guaranteed over-estimation of the actual complete (alpha) vectors produced by this node. That is if

$$\forall_b \quad \exists_{v \in F_n} \forall_{v' \in (G_n \oplus \mathcal{V}_n)} \qquad b \cdot v \geq b \cdot v'. \tag{20}$$

Or, if we use $h^\star_n$ to denote the function induced by $\mathcal{V}_n$, the actual set of vectors computed by n, we can more simply state this requirement as

$$\forall_b \quad f_n(b) \geq g_n(b) + h^\star_n(b). \tag{21}$$

THEOREM 1. *Let n be a search node at depth j, then $F_n = G_n \oplus H_j$, where H_j is defined as the POMDP heuristic* (19)*, is admissible.*

PROOF. By $F_n = G_n \oplus H_j$ we have that the induced function $f_n(b) = g_n(b) + h_j(b)$. Therefore we only need to show that $\forall_b\, h_j(b) \geq h^\star_n(b)$. This clearly is the case because h_j is induced by a cross-sum of POMDP back projections (19), while the latter is induced by cross-sums of DC back projections ($\cup_{\beta \in B'}(\mathcal{V}^t_{ao^{j+1},\beta(o^{j+1})} \oplus \cdots \oplus \mathcal{V}^t_{ao^K,\beta(o^K)})$, for a subset B' of β) and the former are supersets of the latter ($\mathcal{V}^t_{ao^i} \supset \mathcal{V}^t_{ao^i,a'}, \forall a'$) as indicated by (13). □

Given g, h we want to see if we need to further expand the current node. Clearly, $f = g + h$ is represented by the upper surface implied by the set $F = \texttt{Prune}(G \oplus H)$. Therefore, to see if the current node could generate one or more vectors that are not dominated by the set L of full vectors found so far, we need to check if there is a $v \in F$, such that $\exists b$ such that $v \cdot b > w \cdot b, \quad \forall w \in L$. That is, we simply need to check for each $v \in F$ if it is dominated by the set of vectors L representing l. This can be done using the standard LP for checking for dominance [6, 8]. At the bottom of the tree, we add any non-dominated vectors to L.

A final note on the TBP-BB method here is that, while it may be able to avoid computation of complete branches when the heuristics are tight enough, it does not perform memoizationand therefore may need to perform some duplicate work in different parts of the tree. However, this does give this method the advantage of limited memory requirements.

6. EXPERIMENTS

We tested our methods on a set of six problems: Dec-Tiger, OneDoor, GridSmall, Cooperative Box Pushing, Dec-Tiger with Creaks [9], and MeetingGrid2x2[3]. The main characteristics of these problems can be found in Table 1(a).[4] Of particular interest is the right-most column showing the number of β (denoted $|B|$) for each problem, which is a key indicator of its complexity. As all methods compute optimal value functions, we only compare computation times.

Table 2 shows timing results for all six problems, for a set of planning horizons (depending on the problem). We can see that for all domains TBP-M outperforms NAIVE IP, often by an order of magnitude and up to 3 orders of magnitude. TBP-BB performs somewhat worse, but as noted before, requires much less memory.

We also compared against TBP-NOM: a strawman version of TBP-M that does not perform any memoization and re-computes duplicate parts of the tree. It allows us to see the effect of tree-based pruning, without the extra speedups provided by memoization: except for a very small problem (Dec-Tiger(5)), memoization significantly speeds up computations. The results also show that TBP-NOM still is faster than NAIVE IP on almost all problems.

Table 1(b) provides some statistics for TBP-M. The "Memoization" columns show how often the memoization procedure can retrieve a solution ("# hits"), both the absolute number as well as as a percentage of the total number of calls to the cache. The "Nodes" columns show how many nodes in the search tree are actually being visited ("# visited"), as well compared to the total number of nodes in the tree. We can see that as the problem domains grow larger in terms of $|B|$, the percentage of successful cache hits goes down somewhat. More importantly, however, the percentage of visited nodes decreases more rapidly, as larger subtrees have been cached, leading to larger computational savings. Indeed, when comparing TBP-M vs. TBP-NOM in Table 2,

[3]Courtesy of Jilles Dibangoye.

[4]Problems without citation are available from `http://www.isr.ist.utl.pt/~mtjspaan/decpomdp/`.

Table 1: (a) Overview of several characteristics of the problem domains. All problems have 2 agents. (b) Statistics for TBP-M (independent of h). (c) Statistics for TBP-BB (for a particular h).

(a) Problem domains.

Problem	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{O}\rvert$	$\lvert B\rvert$
Dec-Tiger	2	9	4	81
OneDoor	65	16	4	256
GridSmall	16	25	4	625
MG2x2	16	25	16	390625
D-T Creaks	2	9	36	531441
Box Push.	100	16	25	1048576

(b) TBP-M statistics.

	Memoization		Nodes	
Problem	# hits	% total	# visited	% total
Dec-Tiger	324	28.35	1143	54.04
OneDoor	1536	33.22	4624	42.94
GridSmall	5000	36.30	13775	35.53
MG2x2	265000	16.17	1638775	2.11
D-T Creaks	85212	8.02	1062567	1.39
Box Push.	261888	11.31	2314768	1.22

(c) TBP-BB statistics.

	Nodes	
Problem(h)	# visited	% total
Dec-Tiger(2)	2108	74.12
OneDoor(3)	14845	99.87
GridSmall(2)	38589	70.93
MG2x2(2)	26174987	29.98
D-T Creaks(2)	14074281	17.37
Box Push.(2)	206207504	99.99

Problem(h)	TBP-M	TBP-BB	NAIVE IP	TBP-NOM
Dec-Tiger(5)	0.13	0.09	0.23	0.09
Dec-Tiger(10)	0.31	0.43	0.73	0.33
Dec-Tiger(15)	0.98	1.44	2.54	1.19
OneDoor(3)	53.64	1546.73	304.72	56.53
GridSmall(2)	3.93	125.45	64.03	3.80
MG2x2(2)	171.07	2689.35	382093.00	516.03
MG2x2(3)	640.70	11370.40		1499.43
MG2x2(4)	1115.06	24125.30		2813.10
D-T Creaks(2)	63.14	93.16	109.27	121.99
D-T Creaks(3)	149.06	172.79	1595.17	471.57
D-T Creaks(4)	203.44	292.67	4030.47	1150.69
D-T Creaks(5)	286.53	619.25	8277.32	2046.73
Box Push.(2)	132.13	6663.04	1832.98	1961.38

Table 2: Timing results (in s), comparing TBP-M and TBP-BB to Naive IP and TBP-noM. The missing entries for Naive IP on MG2x2(3)/(4) are due to time limits.

we can see that the gap between them grows as $|B|$ increases (c.f. Table 1(a)).

Table 1(c) shows the amount of nodes visited by TBP-BB during the search for a particular horizon. There is a clear correlation with the TBP-BB in Table 2: domains in which TBP-BB can hardly prune any branches (OneDoor and Box Push.), its performance is much worse than TBP-NOM, due to the overhead of maintaining bounds. However, a tighter heuristic could change this picture dramatically. Additionally, computing the heuristic H_j is relatively costly in some domains: for GridSmall(2) it takes $83.95s$, and $1280.23s$ for OneDoor(3).

Finally, note that computing the heuristic (19) using incremental pruning just corresponds to computing (11) and therefore to doing a POMDP backup. However, we can see that TBP-M solves the mentioned problem instances in $3.93s$ resp. $53.64s$. That is, the DC backup is faster than the POMDP backup in these instances. While this is not a trend for all domains, this does suggest that the DC backup no longer inherently suffers from an additional complexity.

7. DISCUSSION & RELATED WORK

Here we discuss our approach, providing pointers to related work and possible directions of future work where possible.

Our experimental evaluation show very favorable results for TBP-M, but the results for TBP-BB are not that great in comparison. While the latter improves over NAIVE IP, it is only able to improve over TBP-NOM (i.e., simple execution of the entire computation tree) for one domain. A significant problem seems to be that the heuristic is not tight enough in many cases. In future research we plan to analyze what causes the big differences in effectiveness of TBP-BB between domains, which may also lead to new insights to improve the heuristics. Another interesting idea is to apply TBP-BB in approximate settings by using tighter (non-admissible) heuristics.

The suitability of the MPOMDP-DC model depends on the ratio of expected duration to synchronize the joint belief state and the duration of a joint action. It is certainly the case that there may be many situations where synchronization is expected to take multiple stages and in such settings our model will also lead to agents deferring their actions, leading to delays. However, even for such cases the techniques developed here may be useful: the one-step delayed can be used as a part of a solution with longer communication delays [27], or to provide upper bounds for such situations [19].

The MPOMDP-DC model is particularly useful for applications that require a low response time. In such applications it is also difficult to perform planning online [23], which motivates the need for offline planning as presented in this paper. We point out, however, that even in cases where online planning is feasible, having a value function to use as a heuristic for the leafs of the search tree is a valuable asset.

While IP entails a choice for the regular (vector-based) backup, an interesting other direction is the exploration of point-based backups. While we do not expect that this can directly lead to further improvements to the exact backup—IP is empirically faster than the (point-based) witness algorithm [13]—point based methods have led to state-of-the-art results for approximate POMDP methods [21, 15]. While a point-based value iteration for MPOMDP-DC has been proposed [18, 27], many questions remain open. For instance, in order to get any kind of quality guarantees for such methods, future research should investigate how to efficiently compute upper bounds for the DC setting. Moreover, it is still unclear how to efficiently perform the point-based backup itself, although there have been recent advances [17]. We expect that it will be possible to draw on work performed on point-based backups for the Dec-POMDP [1, 7, 14, 31].

In fact, this connection with Dec-POMDPs also works the other way around. An important direction of future work is to investigate whether it is possible to transfer TBP to Dec-POMDPs. As mentioned earlier, MPOMDP-DC value functions have also been used as upper bounds for the solution of Dec-POMDPs thereby significantly increasing the size of problems that can be addressed [19, 26]. However, there may be more direct ways in which our methods can advance Dec-POMDP algorithms. For instance, the most influential approximate solution method, MBDP [24], sam-

ples joint beliefs to admit point-based one-step backups. Our methods allow us to perform the one-step backup over the entire joint belief space and thus can find the complete set of useful sub-tree policies obviating the need to pre-specify a 'max-trees' parameter. We also plan to investigate whether our techniques may be useful for exact DP methods [11].

8. CONCLUSIONS

In this article we considered multiagent planning under uncertainty formalized as a multiagent POMDP with delayed communication (MPOMDP-DC). A key feature of this model is that it allows a fast response to certain local observations, relevant in time-critical applications such as intelligent power grid control. We showed that the set of legal vectors (corresponding to admissible joint policies) is a subset of the set of vectors for the MPOMDP. Still, because of the way this restriction is specified (as a union over decentralized control laws β), a naive application of incremental pruning (IP) suffers from a significant additional complexity when compared to the MPOMDP case.

In order to address this problem we presented an analysis that shows that the DC backup operator can be represented as a computation tree and presented two methods to exploit this tree structure. The first, TBP-M, is based on the original bottom-up semantics of the computation tree, and gains efficiency via memoization. The second, TBP-BB, broadens regular branch-and-bound methods by reinterpreting the computation tree in a top-down fashion and generalizing the concepts of f, g and h-values to PWLC functions.

We performed an empirical evaluation on a number of benchmark problems that indicates that TBP-M can realize speedups of 3 orders of magnitude over the Naive IP baseline. TBP-BB is not competitive with TBP-M on all but one domain (it can not prune enough nodes using its heuristic) but still shows the potential to significantly improve over Naive IP in three of six problems. These results show that we have successfully mitigated the additional complexity that the DC backup exhibits over the MPOMDP, allowing for the solution of larger problems.

Acknowledgments

Research supported by AFOSR MURI project #FA9550-09-1-0538 and NWO CATCH project #640.005.003. M.S. is funded by the FP7 Marie Curie Actions Individual Fellowship #275217 (FP7-PEOPLE-2010-IEF).

9. REFERENCES

[1] C. Amato, J. S. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *ICAPS*, pages 2–9, 2009.

[2] J. Bander and C. White, III. Markov decision processes with noise-corrupted and delayed state observations. *Journal of the Operational Research Society*, 50:660–668, 1999.

[3] T. Basar. Two-criteria LQG decision problems with one-step delay observation sharing pattern. *Information and Control*, 38(1):21–50, 1978.

[4] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *JAIR*, 22:423–455, December 2004.

[5] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *UAI*, pages 32–37, 2000.

[6] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *UAI*, pages 54–61. Morgan Kaufmann, 1997.

[7] J. S. Dibangoye, A.-I. Mouaddib, and B. Chai-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *AAMAS*, pages 569–576, 2009.

[8] Z. Feng and S. Zilberstein. Region-based incremental pruning for POMDPs. In *UAI*, pages 146–153, 2004.

[9] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *JAIR*, 24:49–79, 2005.

[10] J. W. Grizzle, S. I. Marcus, and K. Hsu. Decentralized control of a multiaccess broadcast network. In *IEEE Conference on Decision and Control*, pages 390–391, 1981.

[11] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715, 2004.

[12] K. Hsu and S. Marcus. Decentralized control of finite state Markov processes. *IEEE Transactions on Automatic Control*, 27(2):426–431, Apr. 1982.

[13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[14] A. Kumar and S. Zilberstein. Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *AAMAS*, pages 1315–1322, 2010.

[15] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science & Systems*, 2008.

[16] G. E. Monahan. A survey of partially observable Markov decision processes: theory, models and algorithms. *Management Science*, 28(1), Jan. 1982.

[17] F. A. Oliehoek, M. T. J. Spaan, J. Dibangoye, and C. Amato. Heuristic search for identical payoff Bayesian games. In *AAMAS*, pages 1115–1122, May 2010.

[18] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Dec-POMDPs with delayed communication. In *MSDM (AAMAS Workshop)*, May 2007.

[19] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *JAIR*, 32:289–353, 2008.

[20] J. M. Ooi and G. W. 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.

[21] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, pages 1025–1032, 2003.

[22] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16:389–423, 2002.

[23] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *JAIR*, 32:664–704, 2008.

[24] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *IJCAI*, pages 2009–2015, 2007.

[25] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[26] M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *IJCAI*, pages 2027–2032, 2011.

[27] M. T. J. Spaan, F. A. Oliehoek, and N. Vlassis. Multiagent planning under uncertainty with stochastic communication delays. In *ICAPS*, pages 338–345, 2008.

[28] J. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, 30(5):440–446, 1985.

[29] P. Varaiya and J. Walrand. On delayed sharing patterns. *IEEE Transactions on Automatic Control*, 23(3):443–445, June 1978.

[30] P. Varakantham, J. Kwak, M. E. Taylor, J. Marecki, P. Scerri, and M. Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *ICAPS*, 2009.

[31] F. Wu, S. Zilberstein, and X. Chen. Point-based policy generation for decentralized POMDPs. In *AAMAS*, pages 1307–1314, 2010.

[32] I. Xyngi and M. Popov. Smart protection in Dutch medium voltage distributed generation systems. In *Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES*, 2010.