# Tree-Based Solution Methods for Multiagent POMDPs with Delayed Communication

**Frans A. Oliehoek**
MIT CSAIL / Maastricht University
P.O. Box 616
6200 MD Maastricht, The Netherlands

**Matthijs T.J. Spaan**
Delft University of Technology
Mekelweg 4
2628 CD Delft, The Netherlands

### Abstract

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. In this setting, 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 overcome this problem by demonstrating that computation of the MPOMDP-DC backup can be structured as a tree and by 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 (MASs) that operate in stochastic environments and that share their individual observations with a one-step delay. This problem has been extensively studied in the decentralized control literature and dynamic programming algorithms date back to the seventies (Varaiya and Walrand 1978; Grizzle, Marcus, and Hsu 1981; Hsu and Marcus 1982). However, 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.

The task faced by a team of agents is complicated by partial or uncertain information about the world, as well as by stochastic actions and noisy sensors. Especially settings in which agents have to act based solely on their local information have received a large amount of attention in the last decade (Pynadath and Tambe 2002; Seuken and Zilberstein 2008; Oliehoek, Spaan, and Vlassis 2008; Amato, Dibangoye, and Zilberstein 2009; Spaan, Oliehoek, and Amato 2011). The Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework can be used to formulate such problems (Bernstein, Zilberstein, and Immerman 2000), but decentralization comes at a high computational cost: solving a Dec-POMDP is NEXP-complete.

Communication can be used to mitigate the problem of decentralized information: by sharing their local observations, agents can reason about the optimal joint action given the global information state, called *joint belief*. That is, when instantaneous communication is available, it allows to reduce the problem to a special type of POMDP (Pynadath and Tambe 2002), 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 (Oliehoek, Spaan, and Vlassis 2008).

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 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 (Bander and White 1999). However, such formulations are unsuitable for tasks that require a high responsiveness to certain local observations.

A prime example is decentralized protection control in electricity distribution networks by so-called Intelligent Electronic Devices (IED). As power grids move towards integrating more distributed generation capability (e.g., provided by solar panels or fuel cells), more intricate protection schemes have to be developed as power flow is no longer unidirectional (Hadjsaid, Canard, and Dumas 1999). In response, modern IEDs not only decide based on locally available sensor readings, but can receive information from other IEDs through a communication network with deterministic delays (Xyngi and Popov 2010). 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 for one-step delayed communication, in which the agents base their action on their private observation, in addition to the global information of the last stage.[1] This results in the *MPOMDP with delayed communication (MPOMDP-DC)*, a strict generalization of the delayed observation MPOMDP. The MPOMDP-DC solution also plays an important role when considering stochastic delays or delays of more time steps (Spaan, Oliehoek, and Vlassis 2008).

For an MPOMDP-DC one can compute a value function that is piecewise-linear and convex (PWLC) (Hsu and Marcus 1982). This means that it can be represented by a set of vectors, and therefore many POMDP solution methods can be extended to the MPOMDP-DC setting (Oliehoek, Spaan, and Vlassis 2007). However, *incremental pruning (IP)* (Cassandra, Littman, and Zhang 1997), that performs a key operation more efficiently in the case of regular (M)POMDPs, is not directly able to achieve the same improvements for MPOMDP-DCs; a naive application of this technique (NAIVE IP) needs to loop over a number of decentralized control laws that is exponential both in the number of agents and in the number of observations.

In this paper, we target this additional complexity by proposing two novel methods that operate over a tree structure. These methods prune exactly the same vectors as NAIVE IP, but they iterate over the set of candidate vectors in a different way: NAIVE IP loops over all decentralized control laws $\beta$, while TBP methods exploit the similar parts in different $\beta$. 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 expense of memory. The second algorithm, branch and bound (TBP-BB), tries to avoid *unnecessary* computation by making use of upper and lower bounds to prune parts of the tree, providing a different space/time tradeoff.

The empirical evaluation of the proposed methods on a number of test problems shows a clear improvement over NAIVE IP. 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

In this section we formally introduce the MPOMDP model.

**Definition 1.** A *multiagent partially observable Markov decision process* $\mathcal{M} = \langle n,\mathcal{S},\mathcal{A},P,R,\mathcal{O},O,h,b^0 \rangle$ consists of the following: a finite set of $n$ agents; $\mathcal{S}$, a finite set of states; $\mathcal{A} = \times_i \mathcal{A}_i$, the set $\{a^1, \ldots ,a^J\}$ of $J$ joint actions ($\mathcal{A}_i$ is the set of actions available to agent $i$); $P$, the transition function specifies, $P^a(s'|s)$ the probability of transferring from $s$ to $s'$ given $a$; $R$, the reward function, specifies $R^a(s)$ the reward accumulated when taking $a$ from $s$; $\mathcal{O} = \times_i \mathcal{O}_i$, the set $\{o^1 \ldots o^K\}$ of $K$ joint observations; $O$, the observation function, specifies $O^a(o|s)$ the probability of $o$ after taking $a$ and ending up in $s$; $h$, the planning horizon; $b^0$, the initial state distribution.

Every time step one $a = \langle a_1,\ldots,a_n \rangle$ is taken and an $o = \langle o_1,\ldots,o_n \rangle$ is observed. The special case with 1 agent is called a (regular) POMDP. 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 $BU$ implements Bayes rule (Kaelbling, Littman, and Cassandra 1998). We denote the set of all joint beliefs by $\mathcal{B}$, and 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 $\pi^*$ that maximizes the total expected reward. In this paper we will consider planning over a finite horizon $h$.

The $h - t$ steps-to-go *action-value* of $b$ is

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

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 $t$, $\delta^{t*}$, selects the maximizing $a$ for each $b$ and thereby defines the *value function*: $V^t(b) = \max_a Q^t(b,a) = Q^t(b,\delta^{t*}(b))$.

**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 $a_i^t$, and 6) executes $a_i^t$.

It is clear that there are quite a few differences with an MPOMDP. Most notably, 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 $\delta_i^t : \mathcal{B}^{t-1} \times \mathcal{A} \times \mathcal{O}_i \rightarrow \mathcal{A}_i$ 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 beliefs:

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

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 $\delta^t$ into a collection of $\beta$, one for each $\langle b,a \rangle$-pair. The maximization that (2) performs for each such $\langle b,a \rangle$-pair corresponds

[1] The information available to the agents in this case is also referred to as a *one-step delayed sharing pattern*.

to solving a collaborative Bayesian game (Oliehoek et al. 2010) or, equivalently, a Team Decision Problem and is NP-complete (Tsitsiklis and Athans 1985). We also note that the set of possible $\beta$ is a strict super set of the set of joint actions. That is, 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 (Bander and White 1999).

For both settings, the optimal joint policy can be derived from the value functions defined above. The difficulty, however, is that these value functions are defined over the continuous space of beliefs. When no initial $b^0$ is known ahead of time, most methods for solving a (multiagent) POMDP use the fact that (1) 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:

$$Q^t(b,a) = \max_{v_a \in \mathcal{V}^t_a} b \cdot v_a = \max_{v_a \in \mathcal{V}^t_a} \sum_s b(s) v_a(s). \tag{3}$$

We will also write $\mathcal{V}^t = \bigcup_{a \in \mathcal{A}} \mathcal{V}^t_a$ for the complete set of vectors. 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: $v^t_a = R^a + \sum_o g^i_{ao}$, with $g^i_{ao}$ the back-projection of $v^i$, the $i$-th vector in $\mathcal{V}^{t+1}$:

$$g^i_{ao} = \sum_{s'} O^a(o|s') P^a(s'|s) v^i(s'). \tag{4}$$

The set of such *gamma vectors* is denoted by $\mathcal{G}_{ao}$. The algorithm by Monahan (1982) simply generates all possible vectors by, for each joint action $a$, for each possible observation selecting each possible next-stage vector:

$$\mathcal{V}^t_a \quad = \quad \{R^a\} \oplus \mathcal{G}_{ao^1} \oplus \cdots \oplus \mathcal{G}_{ao^K}, \tag{5}$$

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. The operation `Prune` removes all dominated vectors by solving a set of linear programs (Cassandra, Littman, and Zhang 1997; Feng and Zilberstein 2004). That way, the *parsimonious* representation of $\mathcal{V}^t$ can be computed via pruning:

$$\mathcal{V}^t_a = \texttt{Prune}(\{R^a\} \oplus \mathcal{G}_{ao^1} \oplus \cdots \oplus \mathcal{G}_{ao^K}). \tag{6}$$

The technique called *incremental pruning* (IP) (Cassandra, Littman, and Zhang 1997) speeds up the computation of the value function tremendously by realizing that (6) can be re-written to interleave pruning and cross-sums in the following way:

$$\texttt{Prune}((\ldots \texttt{Prune}(\mathcal{G}_{ao^1} \oplus \mathcal{G}_{ao^2}) \ldots) \oplus \mathcal{G}_{ao^K}).$$

## 3 Computing DC Value Functions

As for an MPOMDP, we can represent the value function under delayed communication using vectors (Hsu and Marcus 1982; Oliehoek, Spaan, and Vlassis 2007). 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 $\beta$. That is, we define vectors $g_{aoa'} \in \mathcal{G}_{aoa'}$ analogously to (4), but now we restrict $v^i$ to be chosen from $\mathcal{V}^{t+1}_{a'}$. From these we construct

$$\mathcal{V}^t_a = \{R^a\} \oplus \mathcal{G}_{ao^1\beta(o^1)} \oplus \cdots \oplus \mathcal{G}_{ao^K\beta(o^K)}. \tag{7}$$

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 $\beta$ as a vector of joint actions $\langle a_{(1)} \ldots a_{(K)} \rangle$, 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: $\forall a \;\; \mathcal{V}^t_{a,DC} \subset \mathcal{V}^t_{a,P}$.*

*Proof.* To see this, realize that $\mathcal{G}_{ao} = \bigcup_{a'} \mathcal{G}_{aoa'}$ and that therefore (5) can be rewritten as

$$\begin{aligned} \mathcal{V}^t_{a,P} &= \cup_a (R^a \oplus [\cup_{a'} \mathcal{G}_{ao^1a'}] \oplus \cdots \oplus [\cup_{a'} \mathcal{G}_{ao^Ka'}]) \\ &= \cup_a \cup_{\langle a_{(1)}, \ldots a_{(K)} \rangle \in \mathcal{A}^K} \\ &\qquad\qquad (R^a \oplus \mathcal{G}_{ao^1a_{(1)}} \oplus \cdots \oplus \mathcal{G}_{ao^ka_{(k)}}). \end{aligned}$$

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

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.

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

$$\begin{aligned} \mathcal{V}^t_a &= \texttt{Prune}(\bigcup_{\beta \in B} \mathcal{V}^t_{a\beta}), && (8) \\ \mathcal{V}^t_{a\beta} &= \texttt{Prune}\big(\{R^a\} \oplus \mathcal{G}^t_{ao^1\beta(o^1)} \oplus \cdots \oplus \mathcal{G}^t_{ao^K\beta(o^K)}\big) && (9) \\ \mathcal{G}^t_{aoa'} &= \texttt{Prune}(\mathcal{G}_{aoa'}), && (10) \end{aligned}$$

where (9) uses incremental pruning.

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 $\beta$ 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. In practice, this way of performing the backup requires nearly a factor $|B|$ more time than a POMDP backup. Second, it performs a lot of duplicate work. E.g., there are many $\beta$ 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.

# 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

$$\mathcal{V}^t_a = \texttt{Prune}\Big(\bigcup_{\beta\in B} \big(\{R^a\}\oplus \mathcal{G}_{ao^1\beta(o^1)} \oplus \cdots \oplus \mathcal{G}_{ao^K\beta(o^K)}\big)\Big). \qquad (11)$$

As mentioned, we can interpret $\beta$ as a vector of joint actions. This allows us to decompose the union over $\beta$ into dependent unions over joint actions, as follows:

$$\begin{aligned}\mathcal{V}^t_a = & \bigcup_{\langle a_{(1)}\dots a_{(k)}\rangle\in B} \big(\{R^a\} \oplus \mathcal{G}_{ao^1a_{(1)}} \oplus \cdots \oplus \mathcal{G}_{ao^ka_{(k)}}\big)\\ =&\{R^a\} \oplus \bigcup_{a_{(1)}\in A} \bigcup_{\langle a_{(2)}\dots a_{(k)}\rangle\in B|_{a_{(1)}}} \big(\mathcal{G}_{ao^1a_{(1)}} \oplus \cdots \oplus \mathcal{G}_{ao^ka_{(k)}}\big)\\ =&\{R^a\} \oplus \bigcup_{a_{(1)}\in A} \Big[\mathcal{G}_{ao^1a_{(1)}} \oplus \bigcup_{a_{(2)}\in A|_{a_{(1)}}} \bigcup_{\langle a_{(3)}\dots a_{(k)}\rangle\in B|_{a_{(1)}a_{(2)}}}\\ & \big(\mathcal{G}_{ao^2a_{(2)}} \oplus \mathcal{G}_{ao^3a_{(3)}} \oplus \cdots \oplus \mathcal{G}_{ao^ka_{(k)}}\big)\Big]\\ =&\{\dots\text{etc..}\dots\} \end{aligned} \qquad (12)$$

Here $B|_{a_{(1)}a_{(2)}}$ denotes the set of $\beta$ consistent with $a_{(1)}, a_{(2)}$, and $A|_{a_{(1)}\dots a_{(k-1)}}$ denotes the set of joint actions (for the $k$-th joint observation) that result in a valid $\beta$. The result of completing equation (12) defines a computation tree illustrated in Fig. 1 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 of vectors 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 down the tree have only 2 or even just 1.

Now, to compute (11) 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 shows another important issue: there are identical subtrees 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, we can characterize the nodes inside the ovals as $\langle -, -, \langle *,x\rangle, \langle *,y\rangle\rangle$, 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.

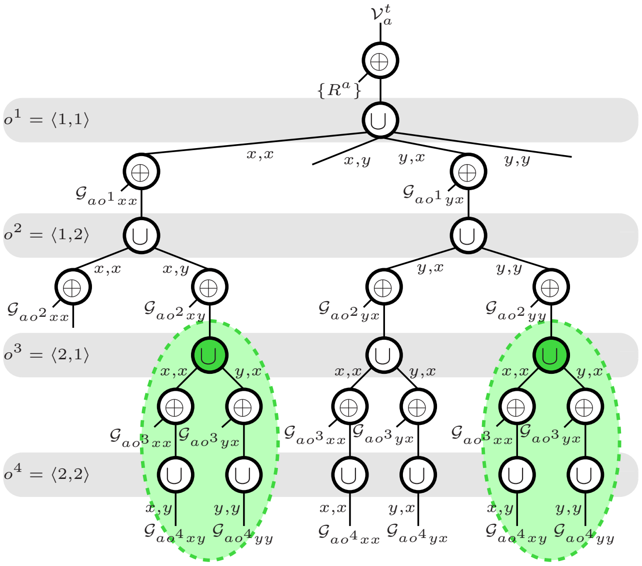

Figure 1: The computation tree of $\mathcal{V}^t_a$.

# 5 Branch & Bound

Tree-based pruning using branch and bound (TBP-BB) is a second method to overcome the problems of naive incremental pruning. Branch and bound (BB) is a well-known method to prune parts of *search trees*. The idea is that while TBP-M avoids duplicate work, TBP-BB may be able to avoid doing unnecessary work. Also, TBP-M needs to cache results which may lead to memory problems. TBP-BB does not perform caching, creating an attractive alternative to trade off space and time complexity.

In particular, standard BB 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 an admissible heuristic estimate. Since our computation tree has vector-valued leaves, BB 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 $v^t_a$.

This idea is illustrated in Fig. 2. While the computation tree works bottom-up, we can also interpret it top-down: by associating the null-vector $\vec{0}$ with the root node, we can now pass 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

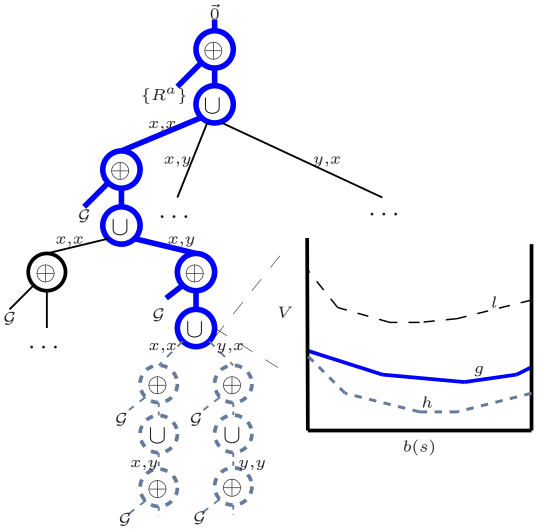

Figure 2: A part of the search tree from Fig. 1 illustrating the heuristics for a search node.

to that node. 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:

$$H_j = \texttt{Prune}(\mathcal{G}_{ao^{j+1}} \oplus \cdots \oplus \mathcal{G}_{ao^K}). \qquad (13)$$

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 an optimisic estimate of the actual complete (alpha) vectors produced by this node. That is if

$$\forall_b \quad \exists_{v \in F_n} \forall_{v' \in (G_n \oplus \mathcal{V}_n)} \qquad b \cdot v \geq b \cdot v'. \qquad (14)$$

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 $\forall_b \quad f_n(b) \geq g_n(b) + h^\star_n(b)$.

**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* (13)*, 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 (13), while the latter is induced by cross-sums of DC back projections ( $\cup_{\beta \in B'} (\mathcal{G}_{ao^{j+1}\beta(o^{j+1})} \oplus \cdots \oplus \mathcal{G}_{ao^K\beta(o^K)})$ for a subset $B'$ of $\beta$) and the former are supersets of the latter ($\mathcal{G}_{ao^i} \supset \mathcal{G}_{ao^i a'}, \forall a'$) as indicated in the proof of Prop. 1. □

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 $\exists b \; v \cdot b > w \cdot b, \forall w \in L$. 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 (Cassandra, Littman, and Zhang 1997; Feng and Zilberstein 2004). At the bottom of the tree, we add any non-dominated vectors to $L$.

| 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 |

Table 1: Overview of several characteristics of the problem domains. All problems have 2 agents.

## 6 Experiments

In order to assess the efficacy of the proposed methods, we performed an empirical evaluation. This evaluation is not geared at examining the assumptions of the MPOMDP-DC model; as mentioned it is a strict generalization of the centralized model assuming one-step delayed joint observations and an evaluation of incorrectly assuming instantaneous communication is presented by Spaan, Oliehoek, and Vlassis (2008). We tested our methods on a set of six problems: Dec-Tiger, OneDoor, GridSmall, Cooperative Box Pushing, Dec-Tiger with Creaks[2] (Gmytrasiewicz and Doshi 2005), and MeetingGrid2x2[3]. The main characteristics of these problems can be found in Table 1.[4] Of particular interest is the right-most column showing the number of $\beta$ (denoted by $|B|$) 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. However, as problem domains grow larger in terms of $|B|$, TBP-M is able to

[2] To turn the problem into a Dec-POMDP, we sum both agents' reward functions.

[3] Courtesy of Jilles Dibangoye.

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

| 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.

cache larger subtrees, leading to larger computational savings. Indeed, when comparing TBP-M vs. TBP-noM in Table 2, we can see that the gap between them grows as $|B|$ increases (c.f. Table 1). Regarding the performance of TBP-BB, in domains in which TBP-BB can hardly prune any branches such as 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 (13) using incremental pruning just corresponds to computing (2) 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.

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. In many situations synchronization is expected to take multiple stages. However, even for such cases our techniques are useful: the one-step delayed solution can be used as a part of a solution with longer delays (Spaan, Oliehoek, and Vlassis 2008).

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 will directly lead to further improvements to the exact backup, point-based methods have led to state-of-the-art results for approximate POMDP methods (Pineau, Gordon, and Thrun 2003; Spaan and Vlassis 2005; Kurniawati, Hsu, and Lee 2008). While approximate point-based value iteration for MPOMDP-DC has been proposed (Oliehoek, Spaan, and Vlassis 2007), 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 (Oliehoek et al. 2010). We expect that it will be possible to draw on work performed on point-based backups for the Dec-POMDP (Amato, Dibangoye, and Zilberstein 2009; Dibangoye, Mouaddib, and Chai-draa 2009; Kumar and Zilberstein 2010).

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. The optimal solution of the MPOMDP-DC is useful to upper bound the expected value of MASs without communication (Grizzle, Marcus, and Hsu 1981; Ooi and Wornell 1996) and thus as a heuristic in solving Dec-POMDPs (Oliehoek, Spaan, and Vlassis 2008; Spaan, Oliehoek, and Amato 2011). 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 (Seuken and Zilberstein 2007), samples 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 (Hansen, Bernstein, and Zilberstein 2004).

Furthermore, our methods can also be used to find the Pareto-optimal set of decentralized control laws in multi-objective team decision problems. This is an extension of the team decision problem (Tsitsiklis and Athans 1985) or identical payoff Bayesian game (Oliehoek et al. 2010), where the payoff function generates a vector of payoffs $u(o,a) \in \mathbb{R}^k$. That means that each $\beta$ defines a value vector $v_\beta \in \mathbb{R}^k$. Given a weight $w$ of the objectives, the final value is given by the inner product $v_\beta \cdot w$. When solving the multiobjective problem, we are interested in the set $L \subseteq B$ of $\beta$ that are maximizing at some point in the space of parameters $w$. That is, we want to compute $L = \texttt{Prune}(\{v_\beta | \beta \in B\})$. This directly corresponds to the problem we solved in this paper. Here we we would have sets $\mathcal{G}_{o^l} = \left\{u(o^l,a^1),\ldots,u(o^l,a^J)\right\}$ for all joint types. Since a (non-identical payoff) Bayesian game is exactly the same from mathematical point of view (only now there is one objective associated with each agent), our methods can be used to find Pareto-optimal joint strategies for them.

## 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 $\beta$), 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 we proposed 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 by 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 out 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. Finally, we discussed how our results are applicable to Dec-POMDPs and multi-objective team decision 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).

## References

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

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

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

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

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

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

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

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

Hadjsaid, N.; Canard, J.-F.; and Dumas, F. 1999. Dispersed generation impact on distribution networks. *IEEE Computer Applications in Power* 12(2):22–28.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Spaan, M. T. J., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *JAIR* 24:195–220.

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

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

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

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

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