Hybrid Value Iteration for POMDPs

Diego Maniloff *
SENSEable City Laboratory
Massachusetts Institute of Technology

Piotr Gmytrasiewicz
Artificial Intelligence Laboratory
University of Illinois at Chicago

Abstract

The Partially Observable Markov Decision Process (POMDP) provides a rich mathematical model for designing agents that have to formulate plans under uncertainty. The curses of dimensionality and history associated with solving POMDPs have lead to numerous refinements of the value iteration algorithm. Several exact methods with different pruning strategies have been devised, yet, limited scalability has lead research to focus on ways to approximate the optimal value function. One set of approximations relies on point-based value iteration, which maintains a fixed-size value function, and is typically executed offline. Another set of approximations relies on tree search, which explores the implicit tree defined by the value iteration equation, and is typically executed online. In this paper we present a hybrid value iteration algorithm that combines the benefits of point-based value iteration and tree search. Using our approach, a hybrid agent executes tree search online, and occasionally updates its offline-computed lower bound on the optimal value function, resulting in improved lookahead and higher obtained reward, while meeting real-time constraints. Thus, unlike other hybrid algorithms that use an invariant value function computed offline, our proposed scheme uses information from the real-time tree search process to reason whether to perform a point-based backup online. Keeping track of partial results obtained during online planning makes the computation of point-based backups less prohibitive. We report preliminary results that support our approach.

1 Introduction

The Partially Observable Markov Decision Process (POMDP) is a powerful mathematical model for designing agents that have to plan under uncertainty. POMDPs are inherently hard to solve using value iteration given the curse of dimensionality (Kaelbling, Littman, and Cassandra 1998), whereby planners must handle large state spaces, and the curse of history (Pineau, Gordon, and Thrun 2003), whereby the number of possible histories to be considered grows exponentially with the planning horizon. For these reasons, researchers have explored different ways of refining the value iteration algorithm. Numerous authors have proposed exact methods with various pruning strategies (Cassandra et al. 1997; Kaelbling, Littman, and Cassandra 1998). However, given the limited scalability of these algorithms, the focus has shifted toward approximate solutions. One set of approximations relies on point-based value iteration to build a fixed-size value function (Poon 2001; Pineau, Gordon, and Thrun 2003; Spaan and Vlassis 2005), and is typically executed offline. Another set of approximations relies on tree search to construct a local policy for a single belief state (Paquet, Tobin, and Chaib-draa 2005; Ross and Chaib-draa 2007; Ross et al. 2008), and is typically executed online.

Offline algorithms take a POMDP problem as input and compute a value function from which an associated policy is extracted *before* the agent is deployed in the environment. These algorithms have to face the difficulty of planning for all possible situations a priori, only knowing the initial belief state at which the agent will be deployed. On the other hand, online algorithms take a POMDP specification and the agent's current belief state as input and compute a single action *while* the agent is in the environment. The advantage here is that the agent knows what belief state it is currently in and can therefore plan for immediate contingencies. However, online planning is generally required to meet real-time constraints and this can be difficult for large POMDPs.

Recent research has shown a creative way to combine offline and online computation that can outperform either technique alone (Paquet, Chaib-draa, and Ross 2006; Ross et al. 2008). A hybrid design makes use of an offline-computed value function as a partial result that will be improved upon by an online search process. Typically, a hybrid algorithm invests offline computation to calculate upper and lower bounds on the optimal value function and uses these bounds, which do not change during online computation, at the leaves of a search tree. Once online, the tree search process is guided by these bounds to prune suboptimal actions and identify the belief states that contribute most to the approximation error at the current state. The search seeks to minimize the interval defined by the bounds. Therefore, we can improve the performance of a hybrid algorithm by supplying tighter bounds.

In this paper we present a hybrid value iteration algorithm that combines the benefits of point-based value iteration and tree search, following up on the above idea, also suggested

*Part of this work was done while the first author was at UIC.

by (Ross et al. 2008). Our algorithm takes as input a pair of bounds, initially computed offline, to perform heuristic tree search online, and occasionally utilizes a portion of the online planning time to update one of these bounds across the entire belief space. Two points are crucial. First, the agent can use the information present in its tree to estimate the benefit that such an update will provide for future online tree searches. Second, by allowing a point-based update to piggy-back on the online tree search we speed up the expensive alpha vector computation without violating real-time constrains.

2 Background

Formally, a POMDP problem instance is specified as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R \rangle$, where $\mathcal{S}$ is the set of world states, $\mathcal{A}$ is the set of actions, $\mathcal{O}$ is the set of observations, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is the stochastic transition function, $O : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow [0,1]$ is the stochastic observation function, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. In a POMDP, the agent's task is to maximize the expected future reward, usually discounted by a factor γ.

In the context of combining offline and online algorithms, the following parameters will be necessary:

Offline time t_{off} is defined as the time measured from the moment the agent is presented with the POMDP specification to the moment in which it is launched in the environment;

Online time t_{on} is defined as the time measured from the perception of an observation until the execution of an action, and determines the real-time constraints of the system.

2.1 Belief state computation

Given that in a POMDP model the world is partially observable, an agent has to resort to noisy observations to estimate the state it is in. It does so by maintaining an internal *belief state* b that summarizes its previous interaction with the environment. Based on this estimate, the agent can output the action that yields the highest expected reward. From this we can describe an agent in terms of a *state estimator* component and a *policy* component (Kaelbling, Littman, and Cassandra 1998). If we let s_t, a_t, and o_t denote the state, action, and observation at time t, we can define $b_0(s) = P(s_0 = s)$, $T(s,a,s') = P(s_{t+1} = s' | s_t = s, a_t = a)$, and $O(s',a,o) = P(o_{t+1} = o | s_{t+1} = s', a_t = a)$. From this we learn that the internal belief state is realized as a probability distribution over $\mathcal{S}$. This belief is updated with every observation as follows:

$$\begin{aligned} b_o^a(s') &= \frac{O(s',a,o) \sum_s T(s,a,s') b(s)}{P(o|b,a)} \\ &= \beta \, O(s',a,o) \sum_s T(s,a,s') b(s), \end{aligned} \tag{1}$$

where we treat $1/P(o|b,a)$ as a normalizing coefficient β.

2.2 Value Iteration

A POMDP is solved optimally for a particular time horizon by means of the value iteration algorithm (Sondik 1971). This dynamic programming approach is used to iteratively build a value function that expresses the expected reward for each belief state b with t steps to go. The value function for t steps to go is obtained from the one with $t-1$ steps to go using the Bellman Update:

$$V_t(b) = \max_a \left[R(b,a) + \gamma \sum_o P(o|b,a) V_{t-1}(b_o^a) \right]. \tag{2}$$

We now present three alternative ways of performing value iteration.

Exact value backups Given that the value function is piece-wise linear and convex in the belief (Smallwood and Sondik 1973), we can represent it as a set of α-vectors $V_t = \{\alpha_0, \alpha_1, ..., \alpha_{|V_t|}\}$, and the value of a belief state will be $V_t(b) = \max_{\{\alpha_i\}_i} [\alpha_i \cdot b]$. Thus, the Bellman Update amounts to transforming every α-vector in V_{t-1} to obtain a new set V_t. This operation, called an exact value backup, is implemented by generating the following sets (Monahan 1982):

$$\Gamma^{a,*} : \alpha^{a,*}(s) = R(s,a) \tag{3}$$

$$\Gamma^{a,o} : \alpha_i^{a,o}(s) = \gamma \sum_{s'} O(s',a,o) T(s,a,s') \alpha_i(s') \tag{4}$$

$$\Gamma^a = \Gamma^{a,*} \oplus \Gamma^{a,o_1} \oplus \cdots \oplus \Gamma^{a,o_{|\mathcal{O}|}}. \tag{5}$$

Finally, the new value function is the union $V_t = \bigcup_a \Gamma^a$.

An exact value backup amounts to considering all conditional plans that result from executing an action a at time t and receiving every possible observation. There are $|V_{t-1}|^{|\mathcal{O}|}$ ways of continuing execution for each action a, therefore each new value function V_t will have size $|V_t| = \mathrm{O}(|\mathcal{A}||V_{t-1}|^{|\mathcal{O}|})$.

Point-based value backups In order to avoid the exponential growth of the number of α-vectors that results from an exact value backup, an alternative is to approximate V_t by computing point-based backups for a finite set of beliefs $\mathcal{B}$ (Poon 2001; Pineau, Gordon, and Thrun 2003; Spaan and Vlassis 2005). The main difference with respect to the exact value backup is that the point-based backup operator yields a single α-vector per belief point. The sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$ remain the same as in the exact value backup; however, the set Γ^a now becomes dependent on b:

$$\Gamma_b^a = \Gamma^{a,*} + \sum_o \arg\max_{\{\alpha_i^{a,o}\}_i} (\alpha_i^{a,o} \cdot b), \alpha_i^{a,o} \in \Gamma^{a,o}, \tag{6}$$

from which we define the point-based backup operator

$$\text{backup}(b) = \arg\max_{\{\Gamma_b^a\}_a} (\Gamma_b^a \cdot b). \tag{7}$$

Finally, the new value function is approximated by the union $V_t = \bigcup_{b \in \mathcal{B}} \text{backup}(b)$.

A point-based backup of point b involves finding the α-vectors $\alpha_i \in V_{t-1}$ that once back-projected according to a fixed action and every possible observation will yield the highest dot product with b. These vectors are then summed with $\Gamma^{a,*}$ and the process is repeated for each action a, resulting in a time complexity of $\mathrm{O}(|\mathcal{S}|^2 |\mathcal{A}||\mathcal{O}||V_{t-1}|)$, for each point in $\mathcal{B}$.

Online search A third way of performing value iteration is by exploring the implicit tree that is generated by Eq. 2. Online search constructs a tree of beliefs, expands its nodes by computing reachable beliefs from the current one, and propagates value estimates up from the fringe nodes. Usually, upper and lower bound estimates are maintained per belief in order to better guide the expansion process via heuristics:

$$L_T(b) = \begin{cases} V^L(b) & b \in \mathcal{F}(T) \\ \max_a L_T(b,a) & \text{otherwise} \end{cases} \tag{8}$$

$$L_T(b,a) = R(b,a) + \gamma \sum_o P(o|b,a) L_T(b_o^a) \tag{9}$$

$$U_T(b) = \begin{cases} V^U(b) & b \in \mathcal{F}(T) \\ \max_a U_T(b,a) & \text{otherwise} \end{cases} \tag{10}$$

$$U_T(b,a) = R(b,a) + \gamma \sum_o P(o|b,a) U_T(b_o^a), \tag{11}$$

where $\mathcal{F}(T)$ indicates the fringe of the tree, and V^L, V^U are bounds typically computed offline, both of which remain constant during the online search process.

Online search operates with the values of single beliefs, rather than with entire α-vectors, thereby returning a local policy. Evaluating a tree of depth d_T has a cost of $O((|\mathcal{A}||\mathcal{O}|)^{d_T}|S|^2)$, which highlights the importance of using heuristics to selectively expand nodes. Online algorithms thus interleave planning and execution, can be especially useful in changing environments, and exhibit substantially shorter policy construction times (Ross et al. 2008).

3 Hybrid value iteration

The previous section presented three general alternatives for performing value iteration: one based on exact methods, and two based on approximations. This section presents Hybrid Value Iteration (HYVI), a hybrid algorithm that combines the latter two approximate methodologies, namely point-based value iteration and tree search. The main idea is based on the notion of effective lookahead and how to maximize it.

A typical hybrid algorithm goes through an offline phase and an online phase. During the offline phase, the algorithm spends t_{off} computing an upper bound V^U and a lower bound V^L on the optimal value function V^*. During the online phase, the algorithm interleaves instances of execution and instances of online planning. At each instance of online planning, a hybrid algorithm spends t_{on} operating on a point-by-point basis, trying to reduce, for as many belief points b as possible, the interval $V^U(b) - V^L(b)$ defined by the offline-computed bounds. The more belief points it reduces this interval for, the better the performance of the algorithm.

Definition 1. We define the *effective lookahead* of a hybrid algorithm as the number of belief nodes b for which the interval $V^U(b) - V^L(b)$ has been reduced after an instance of online planning.

In order to maximize its effective lookahead, a hybrid agent will attempt to perform the most exhaustive exploration of its tree, expanding as many belief nodes as allowed by the real-time constrains of the system. There is, however, an alternative to performing individual node expansions in the tree of beliefs, namely the possibility of performing online, an update of the offline-computed bounds via a point-based backup. This presents a hybrid POMDP agent with the following metareasoning question: should the effective lookahead be maximized by executing as many individual node expansions as possible during t_{on}, or does it make sense to sometimes allocate a portion of this time to update the offline-computed bounds? The reason why the second option posed by this question could make sense, is that an update of the bounds can have an impact across the *entire* belief space, and future explorations of the tree from subsequent beliefs can make use of such improved bounds, effectively achieving higher effective lookaheads.

Our proposed scheme takes as input a POMDP problem specification and spends t_{off} time to compute an upper bound V^U and a lower bound V^L on the optimal value function V^*. Next, once the agent is launched in the environment, it will partition its online planning time t_{on} into two slots: t_{exp} will be used for regular tree search exploration, and the remaining fraction t_{bak} is reserved for a potential point-based backup of a candidate node within the tree. Once t_{exp} has expired, the agent interrupts its online tree search process and invokes a heuristic to decide whether to invest the remaining t_{bak} on a point-based value backup of the lower [1] bound V^L, or simply continue with further node expansions in its tree. Our implementation is composed of the following elements:

- a pair of offline-computed value function bounds;
- an online tree search strategy;
- a heuristic method to keep track of the best candidate node to backup;
- a decision rule to determine whether to perform a point-based backup of V^L; and
- en efficient routine to compute a point-based backup.

3.1 Offline-computed value function bounds

In order to report results comparable with previous work, we have chosen the QMDP bound (Littman, Cassandra, and Kaelbling 1995) for V^U and the Blind policy bound (Hauskrecht 2000) for V^L. We defer to their authors for a detailed treatment, and provide a concise description below.

QMDP The QMDP upper bound results from swapping max and sum operators in the fully observable MDP approximation (Hauskrecht 2000). Given that QMDP is equivalent to performing a one-step lookahead on the single MDP α-vector, it is essentially assuming that any uncertainty regarding the state will disappear after taking one action.

Blind policy Just like any finite-state controller will have an associated a lower bound on the optimal value function, a blind policy is a particular controller that always selects the same action, regardless of the agent's present belief state. The resulting value function is composed of $|\mathcal{A}|$ vectors, each corresponding to one of the possible blind policies.

[1] In this work we focus on updating the lower bound V^L only, which is the value considered for decision making.

3.2 Online tree search strategy

For our tree search methodology, we have chosen the AEMS2 heuristic, previously used by (Hansen 1998) for policy search and by (Ross et al. 2008) for online POMDP planning. The basic idea of AEMS is to expand the tree such as to reduce the approximation error of the lower bound estimate at the root node. The choice of which node to expand is realized by choosing the belief point b in the fringe of the tree $\mathcal{F}(T)$ whose heuristic value is largest. The general equations to keep track of the best node to expand are as follows, where $H_T(b)$ is the basic heuristic value of fringe node b, and $H_T(b,a)$ and $H_T(b,a,o)$ are factors that weigh this value along the path from b to the root node:

$$\begin{aligned}
H^*_T(b) &= \begin{cases} H_T(b) & b \in \mathcal{F}(T) \\ \max_a H_T(b,a) H^*_T(b,a) & \text{otherwise} \end{cases} \\
H^*_T(b,a) &= \max_o H_T(b,a,o) H^*_T(b^a_o) \\
b^*_T(b) &= \begin{cases} b & b \in \mathcal{F}(T) \\ b^*_T(b, a^T_b) & \text{otherwise} \end{cases} \\
b^*_T(b,a) &= b^*_T(b^a_{o^T_{b,a}})
\end{aligned}$$

where $a^T_b = \arg\max_a H_T(b,a)H^*_T(b,a)$ and $o^T_{b,a} = \arg\max_o H_T(b,a,o)H^*_T(b^a_o)$. The AEMS2 procedure uses the following heuristic functions and factors: $H_T(b) = V^U(b) - V^L(b)$, $H_T(b,a) = 1$ if $a = \arg\max_{a'} U_T(b,a')$ and 0 otherwise, and $H_T(b,a,o) = \gamma P(o|b,a)$.

3.3 Keeping track of the best candidate to backup

The best candidate node to backup will be the one whose resulting α-vector improves V^L over the largest area of the belief space, and by the largest amount. It is difficult to precisely estimate these quantities a priori, as well as whether the new vector will even be used by the agent in its future interactions with the environment. We can, however, use the information present in the tree to identify the most promising belief point in terms of the potential area of impact and the amount of improvement.

We partition the nodes in the tree into fringe nodes and internal nodes, $T = \mathcal{F}(T) \cup \mathcal{I}(T)$. We construct a mapping [2] $f(b) : \mathcal{I}(T) \to \mathbb{R}$ that measures the potential benefit of backing up belief point b. Before presenting our proposed mapping $f(b)$, we introduce the following concepts: $I(b)$, $\hat{H}(b)$, $index(b)$, and $support(\alpha_i)$.

$\boldsymbol{I(b)}$ Is the amount by which belief point b has improved its lower bound value after a single expansion:

$$I(b) = L_T(b) - V^L(b),\ b \in \mathcal{I}(T), \tag{12}$$

where $L_T(b)$ is obtained after computing Eqs. 8 and 9 only once.

$\boldsymbol{\hat{H}(b)}$ Is the normalized entropy of belief point b:

$$\hat{H}(b) = \frac{-\sum_{s\in\mathcal{S}} b(s)log(b(s))}{log(|\mathcal{S}|)},\ b \in \mathcal{I}(T). \tag{13}$$

[2]The restriction of the domain of f to the internal nodes is justified in Section 3.5.

$\boldsymbol{index(b)}$ Is the α-vector in the current lower bound V^L that supports point b:

$$index(b) = \arg\max_{\{\alpha_i\}_i} [\alpha_i \cdot b],\ \alpha_i \in V^L,\ b \in T. \tag{14}$$

$\boldsymbol{support(\alpha_i)}$ Is the *support set* of α-vector α_i:

$$support(\alpha_i) = \{b : index(b) = \alpha_i\},\ \alpha_i \in V^L,\ b \in T. \tag{15}$$

$\boldsymbol{f(b)}$ After computing $I(b)$, $\hat{H}(b)$, $index(b)$, and $support(\alpha_i)$ during the online search procedure, we construct the following mapping, where d^b_T is the depth of belief node b:

$$f(b) = \gamma^{d^b_T}\ I(b)\ \hat{H}(b)\ |support(index(b))|,\ b \in \mathcal{I}(T). \tag{16}$$

The heuristic $f(b)$ captures how promising belief node b is for a point-based backup, both in terms of the amount of improvement and the area of the belief space that the new α-vector will improve on. The amount of improvement is estimated with $I(b)$, the height above b at which the vector backup(b) will pass. The area of improvement is estimated with $|support(index(b))|$, the size of the support set that b is in. Since each support set contains belief points that are geometrically proximal to each other in the belief space, whenever the most promising belief of a support set is chosen for backup it is likely that the new alpha vector resulting from such backup will improve the points in the set. Since it may be that not *all* nodes in the support set of b will improve, we affect this size by $\hat{H}(b)$. The idea here is that $\hat{H}(b)$ is a proxy for the gradient of the new α-vector, and higher gradients are likely to result in fewer belief nodes in the set being improved. Finally, $f(b)$ considers the discount factor γ to favor rewards that are obtainable in the near future over potential large impact beliefs that are only reachable after a large number of time-steps.

In order to efficiently identify the best candidate node to backup in the entire set $\mathcal{I}(T)$, each node $b \in \mathcal{I}(T)$ keeps track of the quantity $f^*(b) = \max_b f(b)$, where b ranges over all internal nodes in the subtree of b. Associated with this quantity is a reference b^*, that points to the corresponding node in the subtree of b that maximizes $f(b)$. This way, if b_c is the current root of the tree T, then b^*_c is the best candidate node to backup in the entire set $\mathcal{I}(T)$. Figure 1 shows a schematic view of all the metrics for a simple two-state problem.

3.4 Deciding whether to perform a backup

In order to decide whether to perform a point-based backup, a hybrid agent should compare the gain in effective lookahead of the node expansions that can be executed during t_{bak} against the gain in effective lookahead of adding a new α-vector to V^L that results from a point-based backup of b^*_c, the best candidate node in the entire set $\mathcal{I}(T)$. Let

r, be the rate of belief node expansions in nodes/sec of our implementation of online tree search;

$p = r\ t_{\text{on}}$, the number of belief nodes that can be expanded in t_{on} sec;

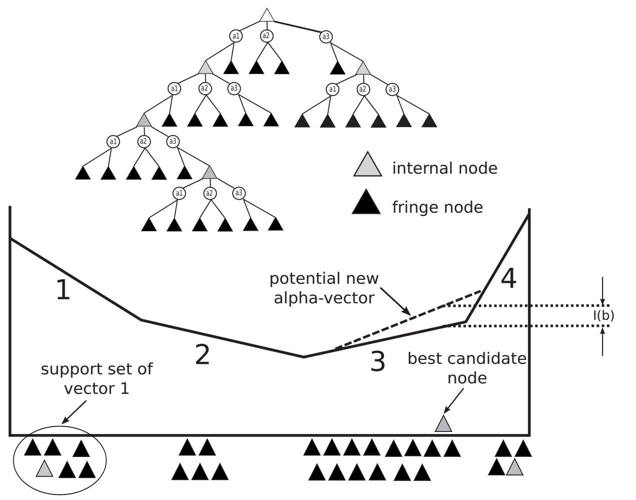

Figure 1: Sample two-state problem. Shown is a tree after an instance of online planning, together with the current lower bound V^L.

$k = r\ t_{\text{bak}}$, the number of belief nodes that can be expanded in t_{bak} sec;

$n_{b_c^*}$, the area of the belief space over which the vector backup(b_c^*) improves the lower bound V^L.

Our underlying assumption is that if the agent decides to update V^L, it will reduce its effective lookahead by k in the *present* time-step t_c, in order to increase its effective lookahead by $p\ n_{b_c^*}$ in *all future* time-steps $t > t_c$, given that the updated V^L continues to be reused. Further, we estimate $n_{b_c^*}$ as

$$n_{b_c^*} = \frac{\hat{H}(b)\ |support(index(b_c^*))|}{p-k}. \quad (17)$$

We later describe the caveats associated with this assumption and the weaknesses of this estimation.

Formally, for the backup to make sense, we must satisfy the following, where t_c is the current time-step in the lifetime of the agent and $\bar{H}$ is the average episode length, empirically measured for a given POMDP problem:

$$\begin{aligned}\gamma^{t_c}(p-k) + \sum_{t=t_c+1}^{\bar{H}} \gamma^t(p + p\ n_{b_c^*}) &> \sum_{t=t_c}^{\bar{H}} \gamma^t p \\ \sum_{t=t_c+1}^{\bar{H}} \gamma^t(p + p\ n_{b_c^*}) &> \sum_{t=t_c+1}^{\bar{H}} \gamma^t p + \gamma^{t_c} k \\ p\ n_{b_c^*} &> \frac{1-\gamma}{\gamma - \gamma^{\bar{H}-t_c+1}} k.\end{aligned} \quad (18)$$

This rule will execute a point-based backup of V^L whenever the effective lookahead that is lost from giving up k expansions in the present time-step is offset by the effective lookahead $p\ n_{b_c^*}$ that is gained in the future. The rule analyzes this tradeoff both in light of the discount factor γ, and the average number of time-steps left in the lifetime of the agent. Intuitively, deeper effective lookaheads in the present are preferred over deeper effective lookaheads in the future, and this preference is intensified as the end of the world nears.

3.5 Efficiently computing a point-based backup

There exists a close relationship between the computations that are part of online search and those that are part of point-based value backups, which makes these two approaches especially suitable for hybridization. In particular, consider a belief point $b \in \mathcal{I}(T)$ for which we have performed a single-step expansion as described in Eqs. 8 and 9. By storing partial results of this expansion, a potential backup(b) computation can be performed more efficiently by avoiding the $\arg\max_{\{\alpha_i^{a,o}\}_i}$ operation in Eq. 6, and the $\arg\max_{\{\Gamma_b^a\}_a}$ operation in Eq. 7. We first analyze the $\arg\max_{\{\alpha_i^{a,o}\}_i}$ operation in Eq. 6.

Proposition 1. *The index i of the back-projected α-vector, $\alpha_i^{a,o}$, that maximizes the dot product with belief b, is the same as the index of the unprojected α-vector, α_i, that maximizes the dot product with the unnormalized belief b_o^a.*

Proof.

$$\begin{aligned}\alpha_i^{a,o} \cdot b &= \sum_s b(s) \alpha_i^{a,o}(s) \\ &= \sum_s b(s) \sum_{s'} O(s',a,o) T(s,a,s') \alpha_i(s') \text{ (Eq. 4)} \\ &= \sum_{s'} \alpha_i(s') O(s',a,o) \sum_s T(s,a,s') b(s) \\ &= 1/\beta \sum_{s'} \alpha_i(s') b_o^a(s') \text{ (Eq. 1)} \\ &= 1/\beta\ \alpha_i \cdot b_o^a,\end{aligned}$$

where $\alpha_i \in V^L$. □

Since $\max_{\{\alpha_i\}_i} [\alpha_i \cdot b_o^a] = V^L(b_o^a)$, saving this index during the tree exploration in Eq. 8 avoids recomputing $\arg\max_{\{\alpha_i^{a,o}\}_i}$. With respect to the $\arg\max_{\{\Gamma_b^a\}_a}$ in Eq. 7, we simply keep track of the action a that yields the highest value of $L_T(b,a)$ in Eq. 9. This extra bookkeeping allows us to piggy-back a point-based backup of an internal belief node in the tree, now in O($|\mathcal{S}|^2|\mathcal{O}|$) time, on top of online tree search.

4 Evaluation

In order to evaluate our approach, we compared Hybrid Value Iteration to AEMS2, a the state-of-the-art online planning algorithm (Ross et al. 2008). We chose problems from the scalable POMDP literature that range from about 100 states to 100000 states: NetCycle[7] (Poupart and Boutilier 2004), RockSample[7,8], RockSample[7,10], and RockSample[10,10] (Smith 2007). All simulations were run on an Intel processor at 2.2 GHz.

Table 1 reports [3] averages over multiple runs for all possible initial belief states and includes the accumulated discounted reward, the number of nodes in the tree, point-based

[3]These results are for our implementation of AEMS2. As such they do not correspond precisely with the ones in (Ross et al. 2008).

backups, and ϵ-optimal actions found. We observe that by updating V^L, the values that are propagated back from the leaf nodes to the root are more precise than the ones of the Blind policy, essentially providing the HYVI agents with deeper effective lookaheads even with fewer belief nodes in the tree.

Algorithm	Return	Belief nodes	Reused nodes	Backups	# of ϵ-opt actions
NetCycle[7] (128s, 15a, 2o)					
AEMS2	54.9562	1439	269	-	0
HYVI	56.3696	794	0	47.9825	0
RockSample[7,8] (12544s, 13a, 2o)					
AEMS2	20.8999	2214	977	-	2.7797
HYVI	21.4284	1855	0	16.7414	3.8379
RockSample[7,10] (50176s, 13a, 2o)					
AEMS2	20.8980	1466	605	-	0.7342
HYVI	22.3899	936	0	17.5300	2.5088
RockSample[10,10] (102400s, 15a, 2o)					
AEMS2	18.3734	1010	366	-	0.1551
HYVI	19.9286	821	0	18.1629	2.0320

Table 1: Simulation results. In all cases, the online planning time t_{on} was limited to 1 second and partitioned into $t_{\text{exp}} = 0.9$s and $t_{\text{bak}} = 0.1$s. The latter slot is the max time it takes to compute a point-based backup as described in Section 3.5, and experimentally measured for our platform and problems. ϵ was set to 10^{-3}.

5 Discussion

We identify limitations and open questions that will give direction to our future research. The first limitation is that the assumption of Eq. 18 is an optimistic assessment. Even if Eq. 17 correctly estimates the area of improvement of the new α-vector, there is no guarantee that tree searches of *all* future interactions with the environment will reuse this vector in the same proportion. It is possible to introduce a parameter L in Eq. 18 as a "reuse lifetime" of the new α-vector. However, in this work we have tried to design a general approach, and avoided parameters that need empirical testing for different domains. The second limitation is the accuracy of Eq. 17, the estimate of the area of improvement of the new α-vector. A more precise estimate could be obtained by partially computing the backup operation. We could also substitute entropy with the curvature (Smith 2007) of V^L at the candidate point. These computations might be expensive to perform online, and we have yet to explore them.

An open question is whether it pays off to allocate online time to compute *several* point-based value backups rather than just one. It would also be interesting to explore online updates of both V^L and V^U.

6 Conclusion

Our approach is a first step towards designing hybrid POMDP agents that reason about the best way to combine online search and point-based updates for real-time planning. Preliminary empirical results indicate competitive performance with respect to traditional online search, while still meeting real-time constrains.

References

Cassandra, A.; Littman, M. L.; Zhang, N. L.; et al. 1997. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 54–61.

Hansen, E. A. 1998. Solving POMDPs by searching in policy space. *Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)* 211—219.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*.

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

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)*, 362–370.

Monahan, G. E. 1982. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science* 1–16.

Paquet, S.; Chaib-draa, B.; and Ross, S. 2006. Hybrid POMDP algorithms. *Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM-06)* 133–147.

Paquet, S.; Tobin, L.; and Chaib-draa, B. 2005. An online POMDP algorithm for complex multiagent environments. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 970–977.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, volume 18, 1025–1032.

Poon, K. M. 2001. A fast heuristic algorithm for decision-theoretic planning. Master's thesis, Hong-Kong University.

Poupart, P., and Boutilier, C. 2004. VDCBPI: an approximate scalable algorithm for large POMDPs. *Advances in Neural Information Processing Systems* 17:1081–1088.

Ross, S., and Chaib-draa, B. 2007. AEMS: an anytime online search algorithm for approximate policy refinement in large POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2592–2598.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21(5):1071–1088.

Smith, T. 2007. *Probabilistic planning for robotic exploration*. Ph.D. Dissertation, Carnegie Mellon University.

Sondik, E. J. 1971. *The Optimal Control of Partially Observable Markov Processes*. Ph.D. Dissertation, Stanford University.

Spaan, M. T., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*.