POMDP Planning and Execution in an Augmented Space

Marek Grześ and Pascal Poupart
Cheriton School of Computer Science, University of Waterloo
200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada
{mgrzes, ppoupart}@cs.uwaterloo.ca

ABSTRACT

In planning with partially observable Markov decision processes, pre-compiled policies are often represented as finite state controllers or sets of alpha-vectors, which provide a lower bound on the value of the optimal policy. Some algorithms (e.g., HSVI2, SARSOP, GapMin) also compute an upper bound to guide the search and to offer performance guarantees, but they do not derive a policy from this upper bound due to computational reasons. The execution of a policy derived from an upper bound requires a one step lookahead simulation to determine the next best action and the evaluation of the upper bound at the reachable beliefs is complicated and costly (i.e., linear programming or sawtoooth approximation). The first aim of this paper is to show principled and computationally cheap ways of executing upper bound policies which can be even faster than executing lower bound policies based on alpha vectors. The second complementary contribution is a new method to find better upper bound policies that outperforms those obtained by existing algorithms, such as HSVI2, SARSOP, or GapMin, on a suite of benchmarks. Our approach is based on a novel synthesis of augmented and deterministic POMDPs and it facilitates efficient optimization of upper bound policies.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

Keywords

Planning under uncertainty; POMDP; Point-based value iteration

1. INTRODUCTION

Research on efficient POMDP algorithms has advanced significantly in the past decade. The most widely known and esteemed approaches are based on point-based value iteration [13, 19], refinement of lower and upper bounds on the optimal value function [18, 9, 16], and online planning [17] among others. Beyond scalability, there is growing interest in performance guarantees to estimate how far from optimal a policy may be. This has lead to some advances in the computation of upper bounds on the optimal value function since they provide an estimate of the gap between the value of a policy and the value of an optimal policy [16]. Upper bounds can also be used to guide the search in both offline point-based techniques [18, 9, 16] and online search techniques [17]. In addition, they may be used to directly derive a policy, which we will refer to as an *upper bound policy*, however this is not done in practice due to computational reasons.

In the current state of knowledge, the execution of upper bound policies is expensive (often prohibitive) because of the need to evaluate the upper bound value of new beliefs by linear programming or a sawtooth approximation [7], and because of the need to do a lookahead simulation in order to determine the next best action. Lookahead requires the evaluation of the upper bound value at every belief reachable in one step. The first contribution of this paper is an efficient method to execute upper bound POMDP policies without the need to do any lookahead simulation. The second contribution is an analytical justification for improved/optimal methods to select belief points that refine the upper bound, which leads to a powerful algorithm that outperforms HSVI2, SARSOP and GapMin when compared on the most challenging POMDP benchmarks for this class of algorithms [16]. Surprisingly, and contrary to the current perception that the joint use of lower and upper bounds is the main factor for efficient search, our method does not require any lower bound. The entire planning is based on the computation of an upper bound that is tighter than existing upper bounds for the majority of the benchmarks. Our approach is justified by a novel synthesis of augmented [7, 16] and deterministic POMDPs [10, 2], which is introduced in this paper.

2. BACKGROUND

We first introduce key concepts and define the notation used throughout the paper.

2.1 Partially Observable Markov Decision Processes

A partially observable Markov decision process (POMDP) is formally defined by a tuple $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$ which includes a set S of states s, a set A of actions a, a set O of observations o, a transition function $T(s', s, a) = \Pr(s'|s, a)$, an observation function $Z(o, a, s') = \Pr(o|s', a)$, a reward function $R(s, a) \in \Re$, an initial belief $b_0(s) = \Pr(s)$, and a discount factor $0 \leq \gamma \leq 1$. We allow the planning horizon to be either finite or infinite. The goal is to find an optimal policy that maximizes the (discounted) sum of rewards. A policy $\pi : H_t \rightarrow A_t$ can be defined as a mapping from histories $H_t \equiv A_0 \times O_1 \times ... \times A_{t-1} \times O_t$ of past actions and observations to actions A_t, however this definition is problematic for an infinite horizon since histories may be arbitrarily long. The most common approach used to circumvent this issue is to replace histories by finite length sufficient statistics such as beliefs. A belief b is a conditional probability distribution over states, i.e.,

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

$b(s) = \Pr(s|a_0, o_1, \ldots, a_{t-1}, o_t) \forall s \in S$. Since states correspond to corners of the belief simplex, we will sometimes call them **corner beliefs**, and all the other beliefs will be called **interior beliefs**, while the word **belief** alone will refer to both corner and interior beliefs. Therefore, a policy becomes a mapping from beliefs to actions: $\pi : b_t \rightarrow A_t$.

The transition and observation functions T, Z are often combined into $TZ(s', s, o, a) = T(s', s, a)Z(o, s', a) = \Pr(s', o|s, a)$. We will often denote this combined function by $T_{ao}(s', s)$ to emphasize the fact that it defines the state transition induced by an action-observation pair. We will also represent $T_{a,o}$ by a square matrix of size $|S|^2$ for each a, o pair.

When beliefs are used as sufficient statistics, a belief update is required in order to compute a new belief $b_{a,o}$ after executing action a in the current belief and observing o. The Bayesian belief update can be computed based on $T_{a,o}$ as follows:

$$b_{a,o}(s') \leftarrow \sum_s \big\{b(s)T_{a,o}(s, s')\big\} / \sum_{s,s''} \big\{b(s)T_{a,o}(s, s'')\big\}. \quad (1)$$

Algorithms that use belief mappings often exploit the fact that the value V^* of an optimal policy satisfies Bellman's equation:

$$V^*(b) = \max_{a \in A} \Big\{R(b, a) + \gamma \sum_o P(o|b, a)V^*(b_{a,o})\Big\} \quad (2)$$

where

$$R(b, a) = \sum_{s \in S} b(s)R(s, a) \text{ and } \Pr(o|b, a) = \sum_{s,s' \in S} b(s)T_{a,o}(s, s').$$

However, the continuous nature of the belief space prevents us from performing value iteration at all beliefs and therefore the important class of point-based techniques performs point-based Bellman backups only at a finite set of beliefs [13]. An approximation of the value function at all beliefs is obtained by computing the gradient in addition to the value at each belief. This allows the formation of a set of linear value functions that are often represented by α-vectors. The policy π induced by the set of alpha vectors is obtained by computing

$$\pi(b) = a_{best} \text{ where } best = \arg\max_i \ \alpha_i \cdot b \quad (3)$$

An important fact is that such a policy is a lower bound policy and better policies may exist. Upper bounds can confirm whether a policy is near optimal or not, and provides information about areas of the belief space that should be explored further because they may yield higher rewards [18, 9, 16].

2.2 Upper Bounds

The easiest way to compute an upper bound on a POMDP's optimal policy is to use an MDP value function or a value function of a relaxation of the original POMDP that assumes that more information is available, i.e., that the process under consideration is *less* partially observable. With more information, better decision can be made and values computed for such relaxations are, therefore, upper bounds. The simplest relaxation is the QMDP update rule:

$$Q(s, a) = R_a(s) + \gamma \sum_o \sum_{s'} T_{a,o}(s, s') \max_{a'} Q(s', a') \ \forall s, a \quad (4)$$

which computes Q-values of the underlying MDP, assuming that states are completely observable. We can select actions based on such Q-values and determine an upper bound $\bar{V}$ as follows:

$$\bar{V}(b) = \max_a \sum_s b(s)Q(s, a). \quad (5)$$

A tighter upper bound called the *fast informed bound* (FIB) [7] can be obtained with the following recursion:

$$Q(s, a) = R(s, a) + \gamma \sum_o \max_{a'} \sum_{s'} T_{a,o}(s, s')Q(s', a') \forall s, a \quad (6)$$

Unlike QMDP, which assumes that the current state is observable, FIB only assumes that the previous state is observable. It uses less information than QMDP but more information than the original POMDP which is why it yields a tighter upper bound on the optimal value of the POMDP. Overall, QMDP and FIB maintain explicit upper bounds on the values of the states (corner beliefs) and values of interior beliefs can be approximated with Eq. 5.

Further improvement to the quality of upper bounds can be gained by explicitly storing better values at some interior beliefs (better values can be obtained by lookahead). We will call **anchor beliefs** the corner and interior beliefs for which we store explicit values since they "anchor" the upper bound and allow us to bound the value at other beliefs. Let $G = \{\langle b_1, v(b_1)\rangle, \ldots, \langle b_n, v(b_n)\rangle\}$ be the set of all anchor beliefs, b_i, and their upper bound, v_i. We can compute an upper bound $\bar{V}^G(b)$ at any belief b outside of the anchor set by linear programming [7]. The linear program interpolates $\bar{V}^G(b)$ by finding a convex combination c of the anchor beliefs (i.e., distribution $c(b_i) = Pr(b_i)$ over the anchor beliefs) such that $\sum_i c(b_i)b_i(s) = b(s) \forall s$ and $\sum_i c(b_i)v(b_i) = \bar{V}^G(b)$ is minimal. Alternatively, one can approximate the linear program with the so-called *sawtooth approximation* [7] by computing convex combination of the corner belief and a single interior anchor belief. An upper bound policy π^G can be derived from $\bar{V}^G$ by performing a one step lookahead and evaluating $\bar{V}^G$ at $|A||O|$ reachable beliefs:

$$\pi^G(b) = \arg\max_a \Big\{\sum_s b(s)R(s, a) + \gamma \sum_{o \in O} P(o|b, a)\bar{V}^G(b_{a,o})\Big\} \quad (7)$$

Past research [7] has shown that the quality of upper bounds can be improved when value iteration is applied to the entire set of anchor beliefs in G. Such an iterative process would require recomputing—in every iteration—convex combinations of all beliefs reachable in one step from any anchor belief in G. One idea mentioned in [7] is that those convex combinations can be cached and applied in several or even all iterations. Caching of convex combinations leads to the formulation of augmented POMDPs.

2.3 Augmented POMDPs

When value iteration is applied to the set G and caching is used, this process can be viewed as computing the QMDP or FIB bounds for an equivalent POMDP called the *augmented POMDP*. The states of the augmented POMDP are the anchor beliefs, which consist of the original states (corner beliefs) augmented with some interior beliefs. An explicit formulation $\langle S', A', O', TZ', R', \Pr_0(b), \gamma'\rangle$ of augmented POMDPs was considered in [16] where $S' = \{b|(b, v) \in G\}$, $A' = A$, $O' = O$, $\gamma' = \gamma$ and $R'(b, a) = \sum_s b(s)R(s, a)$. The initial distribution $Pr_0(b) = c(b)$ corresponds to the interpolation of b_0 obtained by the convex combination c of anchor beliefs that corresponds to b_0. Similarly, $TZ'(b, b', o, a) = c(b')Pr(o|b, a)$ corresponds to the interpolation of $b_{a,o}$ obtained by a convex combination c of the anchor beliefs times the probability of the observation o. While the augmented POMDP has more states ($|G|$ of them), the states become more observable, which allows QMDP and FIB to produce tighter bounds. In the limit, when all reachable beliefs are treated as anchor beliefs and therefore added to the state space of the augmented POMDP, the augmented POMDP becomes a fully observable belief MDP for which QMDP and FIB compute the exact optimal value function.

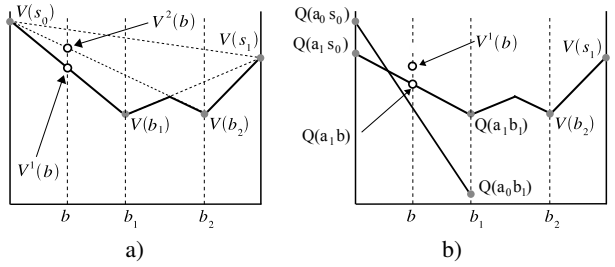

Figure 1: The sawtooth upper bound

3. DIRECT POLICY APPROXIMATION

In this section we explain how upper bound policies can be directly executed without any lookahead simulation, which reduces the online time to select actions. This is particularly important in real-time applications where actions must be selected in miliseconds and resource constrained applications where a reduction in online computation increases battery life [3]. When upper bounds are considered, and the Q-function is available, the bound on the value of a non-anchor belief can be computed using Eq. 5 where the maximization over actions tells us which action is best for the belief b. As a result, the policy can be executed without lookahead. In existing algorithms such as HSVI2, SARSOP and GapMin, the upper bound is not based on a Q-function, but instead it corresponds to $\bar{V}^G$ for a set of anchor beliefs. In order to select actions based on this sort of upper bound, Eq. 7 is used to perform a one-step lookahead simulation.

In what follows, we show how to obtain Q-values using G, which will allow us to execute upper bound policies without lookahead. We illustrate the approach with the sawtooth approximation of the linear program normally used to interpolate upper bound values at non-anchor beliefs. Note that the approach applies to the linear program too.

The classic sawtooth upper bound interpolation is illustrated in Fig. 1a and shows how the value of an arbitrary belief $b \notin G$ can be interpolated by using values of beliefs in G [7, 16]. In our example, the upper bound $\bar{V}^G(b) = V^1(b)$ where $V^1(b)$ is the value of b on the hyperplane defined by $V(s_0)$ and $V(b_1)$. $V^1(b)$ is preferred to $V^2(b)$ because it yields a tighter, i.e., lower, upper bound. Since the upper bound values at anchor beliefs are not with respect to actions like in Q-functions, there is no information about the quality of actions in b.

Our example in Fig. 1b shows an extended version of Fig. 1a where Q-values instead of V-values are shown for points s_0 and b_1. This is possible, because Q-values for every anchor belief point are computed using QMDP or FIB. This suggests that we can potentially tighten upper bounds by using Q-values of beliefs in G, but more importantly, to produce upper bounds that depend on a. Specifically, when V-values are used, the value of b is upper bounded by $V^1(b)$ which is higher than $Q(a_1, b)$. Furthermore, $Q(a_1, b)$ is based on one particular action, hence ranking of actions in b is provided. The improvement is facilitated by the fact that bounds based on Q-values require the same actions for all states that define the bounding hyperplane whereas standard sawtooth always selects V-values which may be obtained by different actions for different states. Overall, this is a graphical justification of the idea for the execution of upper bound policies without lookahead at the cost of $|A|$ interpolations (one for each action).

The number of interpolations can be reduced to one by observing that it is sufficient to compute a single convex combination c of the current belief, b, with respect to the anchor beliefs. This convex combination corresponds to the embedding of the current belief into the belief space of the augmented POMDP. Once our current belief is with respect to the augmented state space, we can perform all necessary computations in the augmented space and the upper bound policy can be computed as follows:

$$c \leftarrow \text{convex combination of } b \text{ with respect to } G \quad (8)$$

$$\pi^G(c) = \arg\max_a \textstyle\sum_{\{b_g|(b_g,v)\in G\}} c(b_g)Q(b_g, a). \quad (9)$$

In this section, we showed how the upper bound policy can be executed without lookahead where only one interpolation of the current belief is computed. In the next section, we introduce another approach, which in addition to being elegant and well-motivated algebraically, does not require any interpolation.

4. POLICY EXECUTION IN AUGMENTED SPACE

In the previous section, we developed an approach that embeds the current belief into the augmented space. Once we are in the augmented space, the selection of an action can be done cheaply by avoiding any lookahead, but at every time step we must compute a convex combination by interpolation to embed the current belief. This embedding is expensive because the interpolation must be done by linear programming (or the sawtooth approximation) in order to obtain a convex combination of anchor beliefs that is equivalent to the current belief. In this section, we go one step further, and we observe that the augmented POMDP can be simulated directly and the entire execution of a policy can be transferred to the augmented space.

Before we explain the validity of the above approach, we show the algebraic relationship between the original and augmented belief spaces. Beliefs, c, in the augmented space are essentially convex combinations of anchor beliefs that constitute corner beliefs in the augmented space. Hence, having any vector c, the corresponding belief, b, in the original space can be computed using the matrix $D = \begin{bmatrix} b_1, \ldots, b_n \end{bmatrix}$, of size $|S||G|$, in which anchor beliefs, b_i, are arranged column-wise. The following equation $b = Dc$ would project the augmented belief, c, onto the original space. The inverse of this operation consists of embedding an original belief in the augmented space by linear programming as described in [7].

THEOREM 1. *Given the same sequence of observations, the simulation of original and augmented POMDPs under the same upper bound policy, and the same initial beliefs generates the same sequence of convex combinations (beliefs in the augmented space) whether inference is performed in the original space and each belief is embeded in the augmented space, or we directly perform inference in the augmented space.*

PROOF. This follows from the fact that the original and augmented POMDPs are equivalent and therefore inference in either space willl yield the same results. □

When a policy is executed in the augmented space, the augmented belief c is updated at every time step and an optimal action can be computed directly by maximizing over $|A|$ dot products, $c \cdot Q_a$, because Q-values are available for every action and every anchor belief.

The complexity of executing various types of POMDP policies is in Tab. 1. The augmented space has a more expensive belief update, however the next two sections will show that matrices, $T_{a,o}$, become sparser when the set G grows, which mitigates the dependence on $|G|$. The cost of querying a lower bound policy depends on the number of α vectors which is often large ($|\alpha| \gg |A|$). The

policy	belief update	action selection
lower bound	$\mathcal{O}(\|S\|^2)$	$\mathcal{O}(\|\alpha\|\|S\|)$
original UB	$\mathcal{O}(\|S\|^2)$	$\mathcal{O}(\|A\|\|O\|(\|S\|^2 + \|G\|^{3.5}L))$
orig UB w/ embedding	$\mathcal{O}(\|S\|^2)$	$\mathcal{O}(\|A\|\|G\| + \|G\|^{3.5}L)$
augmented UB	$\mathcal{O}(\|S\|\|G\|)$	$\mathcal{O}(\|A\|\|G\|)$

Table 1: Complexity of POMDP policy execution. *Original UB* and *augmented UB* refer to the execution of the upper bound policy in the original and augmented space respectively. *Orig UB w/ embedding* refers to the execution of an upper bound policy in the original space where we embed the current belief at each time step.

cost of selecting an action while working in the original space or embedding the current belief in the augmented space is dominated by the cost of solving a linear program $O(|G|^{3.5}L)$ where L is the number of bits to describe the LP. The above complexity results show that the execution time of a particular policy is problem dependent. Nonetheless, taking into account sparsity of augmented POMDPs, we expect significant gains for the approaches that we introduce in this paper. Furthermore, we will show that executing an upper bound policy in the augmented space is not more time consuming than the execution of lower bound policies.

It is interesting to note that the Q-vectors in the augmented space bare some similarities with α-vectors in lower bound policies [14] and finite state controllers [5]. In both cases, they allow an action to be selected by computing some dot products, however they differ in the sense that Q-vectors do not guarantee any value for the resulting policy while alpha-vectors constitute a lower bound that guarantees a minimum value. In the next section, we explain how to grow the set of anchor beliefs in order to tighten the resulting upper bound until it matches the optimal value function. We will first introduce a generalized notion of deterministic POMDPs since the addition of anchor beliefs tends to make the augmented POMDP more deterministic.

5. DETERMINISTIC POMDPS

Deterministic POMDPs were first considered in Littman's thesis [10] who defined deterministic POMDPs as those that have deterministic transitions and deterministic observations, i.e., all entries in T and Z matrices are either zero or one. Littman showed that deterministic POMDPs can be mapped to MDPs with an exponential number of states. Bonet indicated recently in [2] that the LAO* [6] algorithm can be used to reduce the reachable state space when the initial belief state is provided. An important property of deterministic POMDPs that was exploited in these approaches is that the MDP solution of the original POMDP provides an optimal POMDP solution whenever the initial belief state is a corner belief in the POMDP. When this is not the case, then additional planning is required in order to optimize the actions for interior beliefs that are reachable from the initial belief. Quasi-deterministic POMDPs were also considered in [1] where a POMDP is quasi-deterministic when its actions are deterministic, but its observations can be stochastic. In this paper we introduce a more general notion of deterministic POMDPs called *AO-deterministic POMDPs*.

DEFINITION 1. *A POMDP is* AO-deterministic *when all $T_{a,o}$ matrices have at most one non-zero entry in every row.*

Deterministic POMDPs are necessarily AO-deterministic because the product of deterministic transition and observation distributions yields deterministic $T_{a,o}$ matrices with at most one non-zero entry per row. Similarly, quasi-deterministic POMDPs are necessarily AO-deterministic since a deterministic transition distribution guarantees that there will be at most one non-zero entry per row in each $T_{a,o}$. In contrast, there exist some AO-deterministic POMDPs that are not deterministic or quasi-deterministic. Consider a POMDP with a stochastic transition distribution. As a result, it is not deterministic nor quasi-deterministic. If the observation distribution assigns a non-zero probability to a single distinct observation for each state, then each $T_{a,o}$ will have at most one non-zero entry per row. It is also possible to construct transition and observation distributions that are both stochastic, but the resulting $T_{a,o}$ matrices have a single non-zero entry per row. Hence, the class of AO-deterministic POMDPs is a superset of the deterministic and quasi-deterministic POMDPs. The *baseball* problem[1] is an example of a common POMDP benchmark that is AO-deterministic, but not deterministic nor quasi-deterministic.

Similar to previous definitions, AO-deterministic POMDPs can be shown to have the property that policies that are optimal for their underlying MDPs are also optimal at the corner beliefs. This means that policies derived from QMDP and FIB upper bounds are optimal at the corner beliefs and in fact all reachable beliefs when the initial belief is a corner belief. This will become important in the next section when we show how to gradually augment a POMDP with anchor beliefs until it becomes AO-deterministic.

THEOREM 2. *Policies that are optimal for the underlying MDP of an AO-deterministic POMDP are also optimal at the corner beliefs of this POMDP.*

PROOF. When each row in every $T_{a,o}$ matrix has at most one non-zero entry, corner beliefs always transition in a deterministic way to a corner belief, which means that once the underlying state is known, we can always determine the resulting state after each a, o-pair. This is equivalent to working with a fully observable process. Hence a policy that is optimal for the underlying MDP has access to the same information (identity of the state) even when the original POMDP is considered and therefore it is optimal at the corner beliefs. □

COROLLARY 1. *Policies derived from QMDP or FIB upper bounds are optimal at the corner beliefs when the POMDP is AO-deterministic.*

PROOF. QMDP policies are optimal for the underlying MDP and therefore are optimal at the corner beliefs by Thm. 2. We can also show that the FIB and QMDP upper bounds are identical for AO-deterministic POMDPs, which means that FIB policies are also optimal at the corner beliefs. FIB applies the update rule shown in Eq. 6. If for all s only one s' is possible after any a, o-pair, the FIB equation can be simplified to

$$Q(s,a) = R_(s,a) + \gamma \sum_o \max_{a'} T_{a,o}(s,s'(s))Q(s',a') \quad (10)$$

which further reduces to

$$Q(s,a) = R_(s,a) + \gamma \sum_o T_{a,o}(s,s'(s))V(s') \quad (11)$$

because there is only one s' inside $\max_{a'}$. Hence, $V(s')$ can be used to remove the dependence on a and $\max_{a'}$. The resulting update rule is a QMDP update rule as in Eq. 4. □

6. A SYNTHESIS OF DETERMINISTIC AND AUGMENTED POMDPS

Our formulation of AO-deterministic POMDPs that were introduced in Sec. 5 has important implications for augmented POMDPs

[1] http://pomdp.org/examples/index.shtml

and how one may design an algorithm to gradually augment the set of anchor beliefs and obtain tighter upper bounds. In particular, the current perception is that only the beliefs reachable from the initial POMDP belief are important to consider as anchor beliefs [8]. We show that beliefs reachable from corner beliefs can also be good anchor belief candidates.

The process of improving an upper bound can be seen as a loop over three steps: (1) selection of new anchor beliefs, (2) reconstruction of the augmented POMDP with new anchor beliefs, (3) computation of FIB on the augmented POMDP. This is exactly how GapMin is implemented [16]; though, it selects beliefs according to a forward search from the initial belief only. An interesting question is when should we stop adding new anchor beliefs and is it possible to make this decision without consulting any lower bound? The answer is in our definition of AO-deterministic POMDPs. An important feature of augmented POMDPs is that whenever—for an anchor belief b—the reachable belief $b_{a,o}$ is added to G, the transition from b to $b_{a,o}$ becomes deterministic in the corresponding $T_{a,o}$ matrix, i.e., in $T_{a,o}$, the row that corresponds to b, will have only one non-zero entry equal to $\Pr(o|b,a)$ at $b_{a,o}$. Therefore, the process of refining upper bounds through the addition of new anchor beliefs, can be seen as *a gradual determinisation of the corresponding augmented POMDP*. The determinisation is important, because after adding $b_{a,o}$ to the augmented POMDP, the upper bound at b becomes closer to V^*. An exact Bellman backup is used to propagate the upper bound at $b_{a,o}$ back to b. In general, anchor beliefs do not necessarily need to be those that are reachable from the initial belief (as it is done in SARSOP, HSVI2, or GapMin). Beliefs reachable from the corner beliefs are also good candidates since they help to determinise augmented POMDPs.

Overall, considering the properties of AO-deterministic POMDPs, tightening an upper bound can be seen as a dual process. The first process tries to make the augmented POMDP deterministic for corner beliefs and this way the QMDP solution of such a POMDP would be optimal for any initial belief that belongs to G, whereas for any belief not in G the second process with forward search would be required (as introduced in [10, 2]). When $\bar{V}^G(b_0)$ is considered, sometimes better improvements can be achieved through the addition of anchor beliefs found using forward search from the initial belief, whereas in other cases determinisation of the augmented POMDP by a search from the corner beliefs may lead to tighter bounds. The process that reduces $\bar{V}^G(b_0)$ the fastest is domain dependent. In particular, in domains that are AO-deterministic, only forward search is important because there is no need to add beliefs reachable from the corners since that part of the POMDP is already deterministic. Based on this discussion, we propose an algorithm (shown in Alg. 1) to refine upper bounds. When the POMDP is deterministic and $b_0 \in G$ then we know that the upper bound solution is optimal. In contrast, when $b_o \notin G$, then the only thing that makes sense is a forward search from b_0. When the POMDP is not AO-deterministic, we sample the set H of N corner belief states from G where we sample only among corners that have at least one $b_{a,o} \notin G$. Here, the sampling is based on the occupancy frequency (OCF) [15]. Intuitively, the more often the corner belief is visited (when the policy starts in b_0), the more critical it will be to improve its upper bound estimate. Hence, corners whose occupancy frequency with respect to b_0 is higher are ranked higher and are more likely to be sampled. After that, for every belief in H, we sample one trajectory and add to G the first belief that was not in G. When the execution of the above algorithm terminates, we update the augmented POMDP with new anchor beliefs. Next, FIB is computed on a new, larger augmented POMDP and the above procedure is executed again.

Algorithm 1: Anchor Beliefs: Our Method ($N = 50$ in all experiments)

Data: S, G, $\bar{V}^G$, OCF, N, Q- in augmented space
$G_{new} \leftarrow \emptyset$
if *POMDP is AO-deterministic* **then**
  for *i=1 to N* **do**
    if $b_0 \in G$ **then**
      return G_{new}; `/* nothing to improve */`
    else
      $b \leftarrow$ FORWARDSEARCH or LAO*
      add b into G_{new}
else
  $H \leftarrow$ SAMPLECORNERS(G, OCF, N); `/* sample among corners with non-deterministic transitions only */`
  for *all corner beliefs* $b \in H$ **do**
    repeat
      $c \leftarrow$ embed b into augmented space
      $a^* \leftarrow$ action for c using augmented Q-values
      sample observation o according to $P(o|b,a^*)$
      $b \leftarrow b_{a,o}$
    until $b \notin G \cup G_{new}$
    add b into G_{new}
return G_{new}

Interestingly, almost 15 years ago Hauskrecht [7] investigated a very similar direction, however his motivation was different; at least, it did not make connections with deterministic POMDPs. In the algorithm for adaptive selection of anchor beliefs, Hauskrecht sampled beliefs reachable from corners and added reachable beliefs to G. This process also leads to determinisation of the augmented POMDP according to our definition. His motivation for this procedure (shown in detail in Alg. 2) was that when the successors of current anchor beliefs are added to G, the values of corner beliefs may be reduced and this may subsequently reduce the values of beliefs that interpolate the corner beliefs. In every iteration, Alg. 2

Algorithm 2: Anchor Beliefs: Hauskrecht's method

Data: S, G, $\bar{V}^G$
$G_{new} \leftarrow \emptyset$
for *all corner beliefs* $b \in S$ **do**
  repeat
    $a^* \leftarrow$ LOOK-AHEAD(b,$\bar{V}^G$)
    sample observation o according to $P(o|b,a)$
    $b \leftarrow b_{a,o}$
  until $b \notin G \cup G_{new}$
  add b into G_{new}
return G_{new}

samples one trajectory starting form every original corner $b \in S$ (S is the set of corners of the original POMDP). Note, that this algorithm will not find any useful anchor beliefs when the original POMDP is deterministic.

Properties of augmented POMDPs and the fact that they converge to deterministic augmented POMDPs, when more beliefs are added, could be used as another way of illustrating that planning in infinite horizon POMDPs is undecidable [11] since infinitely many anchor beliefs may be required.

7. EXPERIMENTS

Even though POMDP planning with infinite horizon is undecidable [11], Poupart *et. al* [16] reduced the gap between lower and

upper bounds to one unit at the third significant digit for many standard POMDP benchmarks[2]. We report experiments for the POMDPs that were challenging for all algorithms tested in [16], i.e., problems where existing algorithms cannot close the gap to one unit at the third significant digit. Note that even POMDPs with a small number of states may be difficult to solve as suggested by the size of the remaining gap.

7.1 Policy Execution Times

When bounding algorithms (i.e., SARSOP, HSVI2, GapMin) perform planning, they need to execute lower and/or upper bound policies. *Hence, fast execution of upper bound policies can lead to faster planning.* Before we evaluate the quality of upper bounds, we compare the execution time of several upper bound polices; we also include a lower bound policy from SARSOP for reference.

The experiment compares the execution time of 1000 episodes of some policies optimized for 1000 seconds. This includes SARSOP's lower bound policy and three upper bound policies obtained by running Alg. 1 with the sawtooth approximation: a standard lookahead policy in the original space (UB-Orig-lookahead), embedding of beliefs in the augmented space (UB-C), and upper bound policy in the augmented space (UB-Augmented). Fig. 2 shows the running times as multiples (in logarithmic scale) of the time taken by a cheap default QMDP policy. The results confirm that the execution of upper bound policies in the original space with lookahead is time consuming. The time to execute lower bound policies (SARSOP) is also considerably large. While the execution of lower bound policies involve only the computation of dot products between beliefs and α-vectors, the process becomes time consuming when the number of α-vectors is large. UB-C turned out to be the fastest in most cases; though, execution in the augmented space was faster in two cases. Overall the time to execute upper bound polices can be reduced significantly with the techniques described in this paper. Even though the policy in the augmented space was not faster than the UB-C policy in general, it was still faster than other policies in many cases. This is a useful insight because augmented POMDPs provide an elegant way of using anchor beliefs, which are important whenever upper bounds are required for branch-and-bound [12, 4] or bounding planners [16]. Since UB-C was the fastest in this experiment, we used it in Alg. 1 as part of the most competitive solution in the next experiment.

The simulated quality (averaged over 1000 trials) of the policies compared in Fig. 2 is shown in Fig. 3. QMDP was surprisingly good on more than half of the benchmarks. Even if bounding POMDP planners have difficulties to tighten the gap, upper bound policies—QMDP in particular—can be competitive in a practice. In some situations, the upper bound policy in the augmented space (UB-Augmented) showed superior performance, which is nice given that it is the second fastest policy. Policies derived from upper bounds can be poor since they do not directly optimize a lower bound. This happened on a few domains in Fig. 3, e.g., on learning.c2-c4. In such situations, one cannot rely on upper bound policies. Nevertheless, better upper bounds are advantageous for both branch-and-bound and bounding point-based planners.

7.2 Upper Bounds

Upper bounds are useful to evaluate how far from optimal a policy may be and to guide the search for good controllers in branch and bound [12, 4]. Hence, there is a need to tighten upper bounds. We compare the upper bounds from Sec. 6 to the state-of-the-art.

We used the source code of SARSOP, HSVI2, and GapMin available at their authors' websites and all algorithms were executed on the same machine. Our method is named Aug-OCF and corresponds to Alg. 1 whereas Aug-H represents Alg. 2. Out of 24 problems (selection criterion explained at the beginning of Sec. 7), 21 of them were not AO-deterministic according to Def. 1. Hence, for those problems, our algorithm selects anchor beliefs based on the occupancy frequency and ranks corners as shown in Alg. 1. Table 3 shows the results. The column UB shows the upper bound computed by each method. Our algorithm Aug-OCF computed the best upper bounds on the majority of benchmarks. Interestingly, an old idea suggested by Hauskrecht [7] and implemented in Aug-H turned out to be very competitive.

[2]http://www.cassandra.org/pomdp/examples/index.shtml

Table 2: The quality of upper bounds (UB) after 1000 seconds of planning (AO-deterministic POMDPs).

problem	algorithm	gap	LB	UB	$\|\Gamma\|$	$\|\bar{V}\|$	time	UB	$\|\bar{V}\|$	time
baseball	hsvi2	1e-3	0.6412	**0.6412**	991	n.a.	999			
$\|\mathcal{S}\| = 7681$	sarsop	7e-4	0.6412	0.6419	1453	1694	400	0.6412	3878	2346
$\|\mathcal{A}\|$=6 $\|\mathcal{O}\|$=9	GapMin	5.01	0.6346	5.6500	1	1	281	0.6434	52	15219
$\gamma = 0.999$	Aug-OCF			0.6413		3051	970			
rockSample-7_8	hsvi2	3.56	20.91	**24.46**	4752	n.a.	998			
$\|\mathcal{S}\| = 12545$	sarsop	4.12	20.91	25.02	3119	2473	999	24.46	8520	9806
$\|\mathcal{A}\|$=13 $\|\mathcal{O}\|$=2	GapMin	25.07	7.35	32.42	1	1	6.18	26.84	30	13855
$\gamma = 0.950$	Aug-OCF			24.81		3351	978			
underwaterNav	hsvi2	23.4	729.9	**753.3**	3545	n.a.	1000			
$\|\mathcal{S}\| = 2653$	sarsop	23.4	731.0	754.4	7918	2820	999	754.0	7947	10014
$\|\mathcal{A}\|$=6 $\|\mathcal{O}\|$=103	GapMin	80.2	675.06	755.3	1	1	742	754.8	115	10113
$\gamma = 0.950$	Aug-OCF			754.6		1830	471.0			

The three remaining POMDPs were AO-deterministic. Two of them, underwaterNav and rockSample_7-8 have both deterministic actions and deterministic observations whereas baseball has only deterministic observations. Aug-H did not compute any interior anchor belief for those AO-deterministic domains because it starts its sampling process from corner beliefs and in the case of AO-deterministic POMDPs interior beliefs are never reached. For this reason, Aug-H does not appear in the results in Table 2. Aug-OCF performed only forward search from b_0 according to Alg. 1 (the algorithm determines beforehand that expanding initial corners will not find any additional anchor beliefs). Our forward search is a variation of LAO*. The results show that HSVI2 was the best within 1000 seconds time limit. This fact is not surprising because LAO* requires an efficient implementation that does not always adhere to its original motivation; the original paper [6] introduces one such variation called efficient LAO*. Since current research on bounding algorithms for POMDPs has focused on forward search from b_0, HSVI2, SARSOP and GapMin can be seen as fast, efficient implementations of LAO*-type of algorithms, and there is no surprise that HSVI2 was the best in Table 2. This work shows how to tighten the upper bounds for non AO-deterministic POMDPs as reported in Table 3. To further highlight the quality of our results, Table 3 contains additional upper bounds computed by HSVI2, SARSOP, and GapMin, when these algorithms are given an order of magnitude more time. It can be seen that in most cases after 10^4 seconds the upper bound found by these methods is still not as good as what our proposed algorithms find in 10^3 seconds.

8. CONCLUSION AND FUTURE WORK

Tightening upper bounds on the optimal value function remains an important challenge [8, 16]. In this paper, we introduced the notion of AO-deterministic POMDPs, which generalizes previous notions of deterministic POMDPs, and combined it with the notion of augmented POMDPs that are used in bounding planners. We showed that the process of improving upper bounds makes the augmented POMDP deterministic, i.e., by improving the upper bound, one makes the augmented POMDP more deterministic. This explanation allowed us to design a straightforward solution that yields

Table 3: The quality of upper bounds (UB) after 1000 seconds of planning (non AO-deterministic POMDPs).

problem	algorithm	gap	LB	UB	$\|\Gamma\|$	$\|V\|$	time	UB	$\|V\|$	time
aloha.10	hsvi2	9.0	535.4	544.4	4729	n.a.	997	544.1	n.a.	10001.4
$\|\mathcal{S}\| = 30$	sarsop	9.5	535.2	544.7	48	2151	1000	544.3	8035	10000.5
$\|\mathcal{A}\| = 9, \|\mathcal{O}\| = 3$	GapMin	10.7	533.5	544.2	81	223	972	544.0	1140	10741.3
$\gamma = 0.999$	Aug-H			539.6 ± 0.01		1999.1 ± 21.7	981.9 ± 3.6			
	Aug-OCF			**539.0** ± 0.01		3345 ± 22.8	984.5 ± 2.8			
aloha.30	hsvi2	38	1212	1249	2062	n.a.	1000	1247.5	n.a.	10011.6
$\|\mathcal{S}\| = 90$	sarsop	74	1177	1252	86	1245	999	1249.1	5859	10000.1
$\|\mathcal{A}\| = 29, \|\mathcal{O}\| = 3$	GapMin	112	1135	1248	34	212	883	1244.3	1258	11941.7
$\gamma = 0.999$	Aug-H			1244.0 ± 0.03		1603.1 ± 11.4	907.9 ± 4.5			
	Aug-OCF			**1242.9** ± 0.07		1780 ± 36.8	973.8 ± 6.8			
cit	hsvi2	0.0951	0.7430	0.8381	3739	n.a.	975	0.8376	n.a.	10024.4
$\|\mathcal{S}\| = 284$	sarsop	0.0491	0.7909	0.8399	3108	1368	967	0.8395	4484	10337.5
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$	GapMin	0.8379	0.0	0.8379	1	75	882	0.8373	239	10647.5
$\gamma = 0.990$	Aug-H			0.8368 ± 0.0		3450.0 ± 43.6	900.2 ± 22.2			
	Aug-OCF			**0.8358** ± 0.0		1910.0 ± 25.3	957.2 ± 5.3			
fourth	hsvi2	0.3758	0.2416	0.6174	3345	n.a.	994	0.6170	n.a.	10003
$\|\mathcal{S}\| = 1052$	sarsop	0.3300	0.2875	0.6175	3595	888	975	0.6175	2756	10299.1
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$	GapMin	0.6175	0.0	0.6175	1	19	594	0.6169	97	11659.1
$\gamma = 0.990$	Aug-H			0.615 ± 0.0		4176.0 ± 0.0	643.6 ± 14.2			
	Aug-OCF			**0.613** ± 0.0		1735.0 ± 26.5	973.9 ± 5.5			
hallway2	hsvi2	0.5250	0.3612	0.8862	2393	n.a.	997	0.8696	n.a.	10003.1
$\|\mathcal{S}\| = 92$	sarsop	0.5247	0.3737	0.8984	262	1519	992	0.8877	4029	10002.5
$\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 17$	GapMin	0.4495	0.3497	**0.7992**	122	218	835.5			
$\gamma = 0.950$	Aug-H			0.897 ± 0.0		1349.6 ± 11.5	896.2 ± 17.6			
	Aug-OCF			0.805 ± 0.0		861.0 ± 6.3	944.1 ± 12.1			
hallway	hsvi2	0.250	0.945	1.195	1367	n.a.	996	1.185	n.a.	10026.6
$\|\mathcal{S}\| = 60$	sarsop	0.210	0.995	1.206	456	1713	998	1.196	5117	10002.6
$\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 21$	GapMin	0.132	0.989	1.122	94	176	974	1.091	344	2035.3
$\gamma = 0.950$	Aug-H			1.186 ± 0.0		1189.7 ± 13.0	947.1 ± 13.3			
	Aug-OCF			**1.095** ± 0.0		951.0 ± 7.0	946.1 ± 11.5			
iff	hsvi2	0.924	8.931	9.855	7134	n.a.	999	9.828	n.a.	1142.4
$\|\mathcal{S}\| = 104$	sarsop	0.775	9.095	9.871	6811	1991	997	9.827	2684	1566.5
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 22$	GapMin	0.683	9.249	9.932	402	595	955	9.801	1501	3261.5
$\gamma = 0.999$	Aug-H			9.945 ± 0.0		3407.4 ± 18.8	983.8 ± 2.8			
	Aug-OCF			**9.828** ± 0.01		2961 ± 19.7	973.2 ± 5.6			
learning.c2	hsvi2	0.090	1.549	1.639	4082	n.a.	996	1.616	n.a.	10087.7
$\|\mathcal{S}\| = 12$	sarsop	0.093	1.556	1.648	4903	2054	996	1.624	7044	10017
$\|\mathcal{A}\| = 8, \|\mathcal{O}\| = 3$	GapMin	0.054	1.552	**1.607**	245	276	925			
$\gamma = 1.000$	Aug-H			1.996 ± 0.0		1733.7 ± 17.8	989.7 ± 2.8			
	Aug-OCF			1.777 ± 0.0		4471 ± 85.7	987.1 ± 2.1			
learning.c3	hsvi2	0.250	2.364	**2.614**	4229	n.a.	988			
$\|\mathcal{S}\| = 24$	sarsop	0.222	2.446	2.668	981	4094	997	2.647	8300	10008.8
$\|\mathcal{A}\| = 12, \|\mathcal{O}\| = 3$	GapMin	0.217	2.422	2.640	370	243	933	2.607	500	1809.9
$\gamma = 1.000$	Aug-H			2.992 ± 0.0		2293.4 ± 19.2	992.1 ± 2.5			
	Aug-OCF			2.762 ± 0.0		4021 ± 20.2	984.8 ± 2.7			
learning.c4	hsvi2	0.567	3.055	**3.622**	4569	n.a.	999			
$\|\mathcal{S}\| = 48$	sarsop	0.321	3.358	3.679	923	3717	982	3.668	7811	10101.7
$\|\mathcal{A}\| = 16, \|\mathcal{O}\| = 3$	GapMin	0.418	3.291	3.710	280	339	974	3.615	399	2285.3
$\gamma = 1.000$	Aug-H			3.987 ± 0.0		3086.2 ± 14.5	977.3 ± 5.3			
	Aug-OCF			3.749 ± 0.0		3731 ± 36.1	986 ± 3.9			
machine	hsvi2	3.49	63.18	66.66	662	n.a.	982	66.34	n.a.	10003.5
$\|\mathcal{S}\| = 256$	sarsop	3.57	63.18	66.75	150	2742	998	66.4	9846	10004.6
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 16$	GapMin	3.48	62.38	65.87	58	208	898	64.64	1174	12147.0
$\gamma = 0.990$	Aug-H			64.68 ± 0.0		972.0 ± 0.0	809.0 ± 4.1			
	Aug-OCF			**63.84** ± 0.01		965.0 ± 12.3	918.7 ± 17.1			
milos-aaai97	hsvi2	18.31	49.15	67.46	3965	n.a.	998	65.25	n.a.	10003
$\|\mathcal{S}\| = 20$	sarsop	19.61	49.74	69.35	3699	4465	997	67.5	13560	10004.3
$\|\mathcal{A}\| = 6, \|\mathcal{O}\| = 8$	GapMin	15.56	49.95	65.52	488	927	888	62.79	3741	11710.0
$\gamma = 0.900$	Aug-H			53.7 ± 0.0		1634.0 ± 10.9	969.7 ± 2.8			
	Aug-OCF			**52.9** ± 0.05		2941 ± 35.2	980.2 ± 5.3			
mit	hsvi2	0.0939	0.7910	0.8849	5539	n.a.	1000	0.8848	n.a.	10013.5
$\|\mathcal{S}\| = 204$	sarsop	0.0665	0.8189	0.8854	2820	1861	999	0.8853	6015	10090.3
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$	GapMin	0.1677	0.7163	0.8840	53	74	927	0.8825	493	11326.9
$\gamma = 0.990$	Aug-H			0.8827 ± 0.0		3292.8 ± 24.7	957.6 ± 17.1			
	Aug-OCF			**0.8818** ± 0.0		1460 ± 31.4	950.4 ± 8.8			
pentagon	hsvi2	0.1920	0.6341	0.8261	4361	n.a.	997	0.8258	n.a.	10020.1
$\|\mathcal{S}\| = 212$	sarsop	0.1311	0.6962	0.8273	3196	1228	971	0.8273	3960	10089.6
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$	GapMin	0.8258	0.0	0.8258	1	111	784	0.8249	336	12240.9
$\gamma = 0.990$	Aug-H			0.8242 ± 0.0		3304.8 ± 25.8	964.0 ± 5.8			
	Aug-OCF			**0.8239** ± 0.0		1785 ± 14.2	960 ± 8.2			
query.s3	hsvi2	26.2	546.8	573.1	1203	n.a.	997	571.8	n.a.	10000.1
$\|\mathcal{S}\| = 27$	sarsop	28.1	546.8	574.8	112	3132	999	573.9	10556	10000.6
$\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$	GapMin	8.01	546.6	554.7	123	1088	938	550.2	3120	7702.4
$\gamma = 0.990$	Aug-H			551.6 ± 0.02		1960.2 ± 10.2	979.7 ± 3.5			
	Aug-OCF			**550.2** ± 0.05		2801 ± 80.3	972.6 ± 3.9			
query.s4	hsvi2	51.9	569.5	621.4	2846	n.a.	999	620.4	n.a.	10002.9
$\|\mathcal{S}\| = 81$	sarsop	54.3	569.1	623.4	166	6782	1000	622.8	23742	10014.1
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 3$	GapMin	45.0	569.4	614.5	137	631	956	605.2	2945	13881.0
$\gamma = 0.990$	Aug-H			589.4 ± 0.06		1660.5 ± 12.8	892.0 ± 3.6			
	Aug-OCF			**586.4** ± 0.03		1871 ± 10.4	949.6 ± 5.7			
sunysb	hsvi2	0.2396	0.5566	0.7963	4370	n.a.	997	0.7957	n.a.	10076.6
$\|\mathcal{S}\| = 300$	sarsop	0.3233	0.4748	0.7980	3537	1229	986	0.7979	3853	10070.2
$\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$	GapMin	0.7962	0.0	0.7962	1	96	958	0.7948	294	10049.1
$\gamma = 0.990$	Aug-H			0.7919 ± 0.0		3474.8 ± 49.7	898.4 ± 20.4			
	Aug-OCF			**0.7908** ± 0.0		1760 ± 11.8	960.7 ± 7.1			
tiger-grid	hsvi2	0.388	2.138	2.525	3394	n.a.	990	2.510	n.a.	10017.7
$\|\mathcal{S}\| = 36$	sarsop	0.262	2.267	2.529	945	2165	997	2.523	6354	10043.1
$\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 17$	GapMin	0.246	2.174	2.420	103	168	934	2.397	634	3418.2
$\gamma = 0.950$	Aug-H			2.403 ± 0.0		1139.4 ± 8.5	960.2 ± 8.9			
	Aug-OCF			**2.398** ± 0.0		1111 ± 19.7	935.3 ± 9.9			
tagAvoid	hsvi2	3.207	-6.150	-2.943	2896	n.a.	1000	-3.378	n.a.	10001.3
$\|\mathcal{S}\| = 870$	sarsop	3.455	-6.142	-2.686	9324	8049	989	-3.298	18099	10085.4
$\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 30$	GapMin	12.70	-14.0	-1.291	77	310	773	-2.436	1800	10017.0
$\gamma = 0.950$	Aug-H			-0.672 ± 0.0		5840.3 ± 55.8	949.0 ± 5.5			
	Aug-OCF			**-3.660** ± 0.0		6861.0 ± 50.8	990.5 ± 1.4			
cppo3	hsvi2	10.89	12.96	23.84	3773	n.a.	999	23.83	n.a.	10004.5
$\|\mathcal{S}\| = 180$	sarsop	9.69	14.69	24.38	242	3420	998	24.38	8879	10053.8
$\|\mathcal{A}\| = 6, \|\mathcal{O}\| = 6$	GapMin	6.87	15.43	22.30	497	1495	976	21.66	1624	14156.6
$\gamma = 0.900$	Aug-H			21.28 ± 0.01		2808.0 ± 27.8	920.6 ± 23.3			
	Aug-OCF			**20.71** ± 0.03		1221 ± 34.7	937 ± 12			

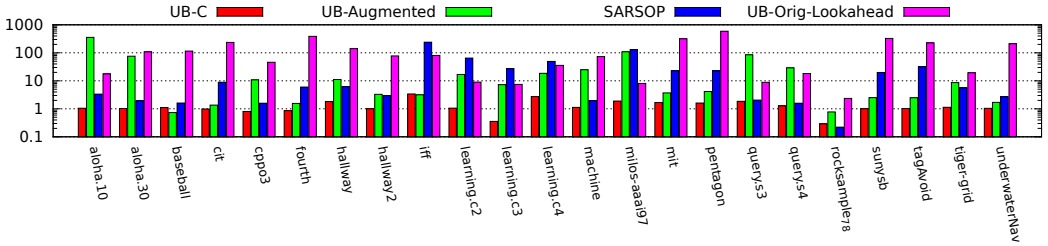

Figure 2: The ratio of execution time of selected policies to QMDP execution time.

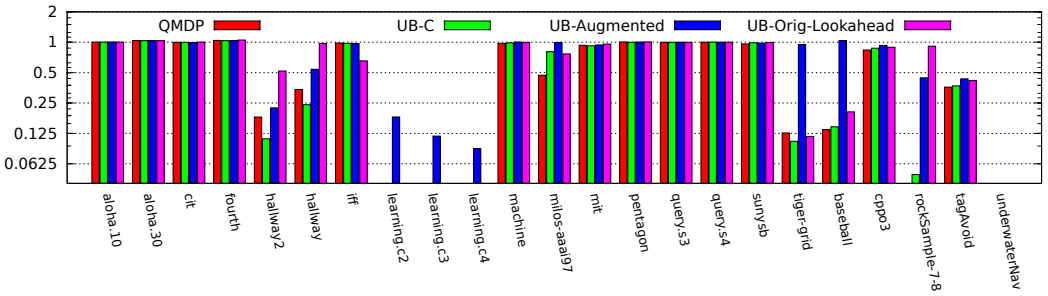

Figure 3: The ratio of simulated values of selected policies to simulated values of SARSOP lower bound policies.

tighter bounds than state-of-the-art bounding planners on the majority of hard benchmarks from [16]. The key to design our algorithms was our synthesis of AO-deterministic and augmented POMDPs. Furthermore, our results were obtained without using any lower bounds to guide the search. In future work, it would be interesting to see how our ideas could be combined with existing planners that optimize both bounds at the same time. We believe that this could lead to further improvements.

Until now, the execution of upper bound policies was not practical due to prohibitive computational costs although upper bounds are computed during planning [8, 16]. We showed two new methods that avoid traditional lookahead and embed beliefs in the augmented space. As a result, the time to execute upper bound policies is now on par and sometimes lower than that of lower bound policies. In the future we would like to explore the use of our methods in the context of branch and bound search techniques [12, 17, 4] since upper bounds are critical for pruning.

Acknowledgement

This research was sponsored by NSERC and MITACS.

9. REFERENCES

[1] C. Besse and B. Chaib-draa. Quasi-deterministic partially observable Markov decision processes. In *ICONIP*, pages 237–246, 2009.

[2] B. Bonet. Deterministic POMDPs revisited. In *UAI*, pages 59–66, 2009.

[3] M. Grześ, P. Poupart, and J. Hoey. Controller compilation and compression for resource constrained applications. In *Proc. of ADT*, 2013.

[4] M. Grześ, P. Poupart, and J. Hoey. Isomorph-free branch and bound search for finite state controllers. In *IJCAI*, 2013.

[5] E. Hansen. An improved policy iteration algorithm for partially observable MDPs. In *NIPS*, 1998.

[6] E. A. Hansen and S. Zilberstein. LAO * : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

[7] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[8] D. Hsu, W. Sun, and L. N. Rong. What makes some POMDP problems easy to approximate? In *NIPS*, 2007.

[9] H. Kurniawati, D. Hsu, and W. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *RSS*, 2008.

[10] M. L. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Department of Computer Science, Brown University, March 1996. CS-96-09.

[11] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite horizon partially observable decision problems. In *Proc. of AAAI*, pages 541–548, 1999.

[12] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *UAI*, pages 417–426, 1999.

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

[14] J. Pineau, G. J. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[15] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto, Toronto, Canada, 2005.

[16] P. Poupart, K.-E. Kim, and D. Kim. Closing the gap: Improved bounds on optimal POMDP solutions. In *ICAPS*, 2011.

[17] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[18] T. Smith and R. Simmons. Point-based POMDP algorithms: improved analysis and implementation. In *UAI*, 2005.

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