# Closing the Gap: Improved Bounds on Optimal POMDP Solutions

**Pascal Poupart**
David R. Cheriton School of Computer Science
University of Waterloo, Canada
ppoupart@cs.uwaterloo.ca

**Kee-Eung Kim** and **Dongho Kim**
Department of Computer Science
KAIST, Korea
{kekim@cs,dkim@ai}.kaist.ac.kr

**Abstract**

POMDP algorithms have made significant progress in recent years by allowing practitioners to find good solutions to increasingly large problems. Most approaches (including point-based and policy iteration techniques) operate by refining a lower bound of the optimal value function. Several approaches (e.g., HSVI2, SARSOP, grid-based approaches and online forward search) also refine an upper bound. However, approximating the optimal value function by an upper bound is computationally expensive and therefore tightness is often sacrificed to improve efficiency (e.g., sawtooth approximation). In this paper, we describe a new approach to efficiently compute tighter bounds by i) conducting a prioritized breadth first search over the reachable beliefs, ii) propagating upper bound improvements with an augmented POMDP and iii) using exact linear programming (instead of the sawtooth approximation) for upper bound interpolation. As a result, we can represent the bounds more compactly and significantly reduce the gap between upper and lower bounds on several benchmark problems.

## 1 Introduction

Recent years have seen impressive improvements in the scalability of POMDP solvers. However the optimal policy of most problems is still unknown. Since the computational complexity of finite horizon flat POMDPs is PSPACE-Complete (Papadimitriou and Tsitsiklis 1987), it is generally agreed that finding an optimal policy is most likely out of reach for all but tiny problems. As a result, most of the advances have focused on the development of scalable approximate algorithms. On that front, approximate algorithms routinely find good policies for many large problems (Hoey et al. 2010; Thomson and Young 2010). However, how good the policies are is a delicate question. Most policies can be evaluated in simulation, meaning that the expected value of the policy is only known up to some confidence interval that holds only with some probability. Some algorithms (including most point-based value iteration techniques) actually compute a lower bound on the value, which provides a guarantee. However, even if the value of the policy is known, it is not always clear how far from optimal it may be. To that effect some algorithms (e.g., HSVI2 (Smith and Simmons 2005), SARSOP (Kurniawati, Hsu, and Lee 2008), grid-based techniques (Lovejoy 1991; Brafman 1997; Hauskrecht 2000; Zhou and Hansen 2001) and some online search techniques (Ross et al. 2008)) also compute an upper bound on the value, but since this tends to be computationally expensive, tightness is often sacrificed for efficiency.

In practice, there is a need for explicit performance guarantees. A common approach to tackle sequential decision making problems consists of going through several rounds of modelling, policy optimization and policy simulation. After a while, domain experts involved in the modeling step will typically inquire about the optimality of the solution algorithm since a lack of optimality could explain questionable choices of actions and perhaps there is no need to further tweak the model. In general, many people outside of computer science do not trust computers and therefore will be more inclined to question the solution algorithm instead of the model, especially when the model is (partly) specified by a human. Furthermore, before deploying a computer generated policy into an industrial application, decision makers will often demand some kind of guarantee regarding the quality of the policy.

In this paper we describe a new algorithm called GapMin that minimizes the gap between upper and lower bounds by efficiently computing tighter bounds. Although our long-term goal is to compute bounds for factored problems, we restrict ourselves to flat problems in this paper. Note that flat problems are still interesting since the optimal value function of many benchmark problems on Cassandra's POMDP website[1] (some of which have served as benchmarks for more than 15 years) is unknown. Our approach is related to point-based value iteration techniques that perform a heuristic search (e.g., HSVI2 and SARSOP). GapMin differs from its predecessors in three important ways: i) a prioritized breadth first search is performed instead of a depth first search, ii) improvements to the upper bound are efficiently propagated with an augmented POMDP and iii) upper bound interpolation is performed exactly by linear programming instead of using the sawtooth relaxation. Here, i) leads to much more compact representations for the bounds and ii) is a technique borrowed from (Hauskrecht 2000) that reduces the number of upper bound interpolations, which al-



[1] http://www.pomdp.org

lows us to use linear programming at a negligible cost while obtaining tighter upper bounds. We tested the approach on 64 benchmark problems from Cassandra's POMDP website. GapMin finds a near optimal solution (gap smaller than one unit at the 3rd significant digit) for 46 problems in less than 1000 seconds (in comparison to 32 problems for HSVI2 and 31 for SARSOP). GapMin also finds lower and upper bound representations that require significantly fewer $\alpha$-vectors and belief-bound pairs than HSVI2 and SARSOP.

The paper is structured as follows. Sec. 2 existing techniques to compute lower and upper bounds for POMDPs. Sec. 3 describes our new algorithm GapMin. Sec. 4 reports the results of the experiments on the suite of benchmark problems from Cassandra's POMDP website. Finally, Sec. 5 concludes and discusses potential future work.

## 2 Background

In this section, we introduce some notation for partially observable Markov decision processes (POMDPs) and quickly review previous work to compute lower and upper bounds on the optimal value function.

### 2.1 Partially Observable Markov Decision Processes

Consider a POMDP $\mathcal{P}$ specified by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \gamma, b_0 \rangle$ where $\mathcal{S}$ is the set of states $s$, $\mathcal{A}$ is the set of actions $a$, $\mathcal{O}$ is the set of observations $o$, $T$ is the transition function indicating the probability of reaching some state $s'$ when executing an action $a$ in state $s$ (i.e., $T(s', s, a) = \Pr(s'|s, a)$), $Z$ is the observation function indicating the probability of making an observation $o$ after executing action $a$ and reaching state $s'$ (i.e., $Z(o, s', a) = \Pr(o|s', a)$), $R$ is the reward function indicating the utility of executing action $a$ in state $s$ (i.e., $R_a(s) \in \Re$), $\gamma \in (0, 1)$ is the discount factor indicating by how much future rewards should be scaled at each step in the future and $b_0$ is the initial distribution over states (i.e., $b_0(s) = \Pr_0(s)$). Alternatively, we can also specify a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathit{TZ}, R, \gamma, b_0 \rangle$ by combining $T$ and $Z$ into a single function $\mathit{TZ}(s', o, s, a) = T(s', s, a)Z(o, s', a) = \Pr(s', o|s, a)$ that indicates the joint probability of state-observation pairs given previous state-action pairs. This alternative formulation will be useful in Section 3 when we specify an augmented POMDP. Since $T$ and $Z$ appear only as a product in the fast informed bound algorithm described in Section 2.3, it is sufficient to specify *TZ*.

Given a POMDP $\mathcal{P}$, the goal is to find a policy $\pi$ that maximizes the expected total rewards. Since the states are not observable, policies are mappings from histories of past actions and observations to the next action. However, this is not convenient since histories grow with the planning horizon. Alternatively, distributions over the hidden states, called beliefs, can be used as a substitute for histories since they are a finite-length sufficient statistic. The belief $b$ at each time step can be updated based on the action $a$ executed and the observation $o$ received to obtain the belief $b_{ao}$ at the next time step according to Bayes' theorem:

$$b_{ao}(s') \propto \sum_s b(s) \Pr(s'|s, a) \Pr(o|s', a)$$

In this paper, we will assume that policies $\pi : \mathcal{B} \rightarrow \mathcal{A}$ are mappings from beliefs to actions. The value $V^\pi(b)$ of executing a policy $\pi$ from a belief $b$ is the expected sum of the rewards earned, which can be expressed recursively by:

$$V^\pi(b) = R_{\pi(b)}(b) + \gamma \sum_o \Pr(o|b, \pi(b)) V^\pi(b_{\pi(b)o})$$

Here $R_{\pi(b)}(b) = \sum_s b(s)R(s, \pi(b))$ and $\Pr(o|b, \pi(b)) = \sum_{ss'} b(s) \Pr(s'|s, \pi(b)) \Pr(o|s', \pi(b))$. An optimal policy $\pi^*$ has an optimal value function $V^*$ that is at least as high as the value of any other policy for all beliefs (i.e., $V^*(b) \geq V^\pi(b)\ \forall b, \pi$). The optimal value function satisfies Bellman's equation:

$$V^*(b) = \max_a R_a(b) + \gamma \sum_o \Pr(o|b, a) V^*(b_{ao}) \quad (1)$$

Smallwood and Sondik (Smallwood and Sondik 1973) also showed that $V^*$ is piece-wise linear and convex with respect to the belief space. This means that the optimal value function can be represented by a (possibly infinite) set $\Gamma^*$ of $\alpha$-vectors that map each state $s$ to some value $\alpha(s)$ yielding linear functions in the belief space (i.e., $\alpha(b) = \sum_s b(s)\alpha(s)$). The optimal value function is the upper surface of the linear functions defined by the $\alpha$'s (i.e., $V^*(b) = \max_{\alpha \in \Gamma^*} \alpha(b)$). In some situations, it is also useful to consider the value $Q^\pi(b, a)$ of executing an action $a$ at $b$ followed by $\pi$. The optimal $Q$ function (denoted $Q^*$) is also piece-wise linear and convex and therefore can be represented by a set of $\alpha$-vectors.

Some algorithms do not represent policies directly as a mapping from beliefs to actions. Instead they use a value function or $Q$-function to implicitly represent a policy. The action of a specific belief $b$ is the action that leads to the largest value according to $Q$ (i.e., $\pi(b) = \operatorname{argmax}_a Q_a(b)$) or a one step lookahead with $V$:

$$\pi(b) = \operatorname*{argmax}_a R_a(b) + \gamma \sum_o \Pr(o|b, a) V(b_{ao}) \quad (2)$$

Algorithms to optimize a policy can generally be divided in two groups: *offline* algorithms (e.g., most value iteration and policy search algorithms) that pre-compute a policy which is executed with minimal computation at runtime and *online* algorithms that do not pre-compute anything, but instead perform a forward search from the current belief at each step to select the next action to execute. In practice, it is best to combine offline and online techniques to precompute a reasonable policy (or value function), which is then refined online by a forward search. In this paper, we focus on the offline computation of lower and upper bounds for the optimal value function. Such bounds may be used to guide an online search and to provide performance guarantees.

**Algorithm 1** Blind Strategies

**Inputs:** $\mathcal{P}$
**Output:** lower bound $\underline{Q}(s)$
$\underline{Q}_a(s) \leftarrow \min_{s'} R_a(s')/(1-\gamma) \;\; \forall a, s$
**repeat**
  $\underline{Q}_a(s) \leftarrow R_a(s) + \gamma \sum_{s'} \Pr(s'|s,a)\underline{Q}_a(s') \forall a, s$
**until** convergence

**Algorithm 2** Point-based Value Iteration

**Inputs:** $\mathcal{P}$ and $\mathcal{B} = \{b_1, \ldots, b_{|\mathcal{B}|}\}$
**Output:** lower bound $\Gamma$ of $\alpha$-vectors
$\Gamma \leftarrow \{\underline{Q}_a | a \in \mathcal{A}\}$ where $\underline{Q} \leftarrow$ blindStrategies($\mathcal{P}$)
**repeat**
  $\Gamma' \leftarrow \emptyset$
  **for** each $b \in \mathcal{B}$ **do**
    $\alpha_{ao} \leftarrow \text{argmax}_{\alpha \in \Gamma} \alpha(b_{ao})$
    $a^* \leftarrow \text{argmax}_a R_a(b) + \gamma \sum_o \Pr(o|b,a)\alpha_{ao}(b_{ao})$
    $\alpha_b \leftarrow R_{a^*} + \gamma \sum_o \Pr(o|b,a^*)\alpha_{a^*o}$
    $\Gamma' \leftarrow \Gamma' \cup \{\alpha_b\}$
  **end for**
**until** convergence

### 2.2 Lower Bounds

A simple and fast lower bound $\underline{Q}$ on the $Q$-function can be computed by finding the value function of *blind* strategies (Hauskrecht 1997) that ignore all observations by always executing the same action (see Alg. 1). In this algorithm, each vector $\underline{Q}_a(s)$ is the value function of the blind strategy that always executes $a$, which is a lower bound for the optimal $Q$-function.

Point-based value iteration techniques (Pineau, Gordon, and Thrun 2006; Spaan and Vlassis 2005; Smith and Simmons 2005; Kurniawati, Hsu, and Lee 2008; Shani, Brafman, and Shimony 2007) gradually refine a lower-bound of the optimal value function. Given a set $\mathcal{B}$ of belief points $b$, they iteratively compute the value of each belief $b$ with its gradient. Since the optimal value function is convex, they find a set $\Gamma$ of hyperplanes known as $\alpha$-vectors that provide a lower bound on the optimal value function. Alg. 2 describes a generic point-based value iteration technique. Specific implementations differ in how the set of belief points is chosen as well as the order in which the value (and gradient) of each belief point is updated. Since the only relevant beliefs are those that are reachable from the initial belief $b_0$, a popular approach consists of growing the set of belief points with the beliefs visited while executing a heuristic policy (Smith and Simmons 2005; Kurniawati, Hsu, and Lee 2008; Shani, Brafman, and Shimony 2007). In particular, when this policy is obtained by a one step lookahead (Eq. 2) with respect to a decreasing upper bound of the value function, then convergence to the optimal value function is guaranteed (Smith and Simmons 2005). This approach can be further refined to focus on the beliefs reachable by the optimal policy by adapting the belief set as the heuristic policy changes (Kurniawati, Hsu, and Lee 2008).

**Algorithm 3** Fast Informed Bound

**Inputs:** $\mathcal{P}$
**Output:** upper bound $\bar{Q}$
$\bar{Q}_a(s) \leftarrow \max_{sa} R_a(s)/(1-\gamma) \;\; \forall as$
**repeat**
  $\bar{Q}_a(s) \leftarrow R_a(s) +$
    $\gamma \sum_o \max_{a'} \sum_{s'} \Pr(s'|s,a)\Pr(o|s',a)\bar{Q}_{a'}(s') \;\; \forall as$
**until** convergence

### 2.3 Upper Bounds

Alg. 3 describes the fast informed bound (FIB) (Hauskrecht 2000), which is a simple and fast upper bound $\bar{Q}$ on the optimal $Q$-function. The update in the second last line of Alg. 3 yields an upper bound because the maximization over $a'$ is taken independently for each state $s$ instead of each belief $b$. Note also that the transition and observation functions only appear as a product, hence the product could be replaced by *TZ*$(s', o, s, a)$.

In some situations, we can compute an upper bound on the value function at specific belief points. Let $\bar{\mathcal{V}} = \{\langle b_1, v_1\rangle, \ldots, \langle b_n, v_n\rangle\}$ denote a set of belief-bound pairs such that $\bar{V}(b_i) = v_i$ returns an upper bound $v_i$ at $b_i$. Since $\bar{\mathcal{V}}$ is only defined at a specific set of beliefs, we will call this set the domain of $\bar{V}$ (i.e. $dom(\bar{V})$).

It is often useful to infer an upper bound on the beliefs outside of the domain of $\bar{V}$. Since the optimal value function is convex, we can interpolate between the beliefs of the domain by solving a linear program. In particular, Alg. 4 shows how to compute the smallest upper bound possible for any belief $b$ given upper bounds $\bar{Q}$ and $\bar{V}$ on the optimal $Q$-function and value function. In addition to computing a bounding value $v^*$, the algorithm returns the lowest convex combination $c^*$ of beliefs (i.e., distribution $c^*(\bar{b})$ of beliefs $\bar{b} \in dom(\bar{V})$). However, since linear programs are computationally expensive, a *sawtooth* approximation (Hauskrecht 2000) (Alg. 5) is used in most state of the art algorithms including HSVI2 and SARSOP. This approximation finds the best interpolation that involves one interior belief with $|\mathcal{S}|-1$ extreme points of the belief simplex (denoted by $e_s$ in Alg. 5). The computation time is only $O(|dom(\bar{V})||\mathcal{S}|)$ and the approximation becomes exact in the limit when $dom(\bar{V})$ contains the entire belief space. So there is a tradeoff: a polynomial amount of computation is saved by avoiding linear programs, but more belief-bound pairs may be necessary to achieve the same level of accuracy. In the worst case, the increase in the number of belief-bound pairs may be exponential since it takes exponentially many beliefs to densely cover an $|\mathcal{S}|$-dimensional space. Alternatively, one can reduce the number of interpolations by caching the distributions $c^*$ that are repeatedly computed at the same beliefs (Hauskrecht 2000). We will apply this technique to mitigate the cost of LP interpolations, while ensuring a bound that is as tight as possible.

## 3 Closing the Gap

We propose a new algorithm called GapMin that minimizes the gap between lower and upper bounds on the optimal

**Algorithm 4** UB (LP upper bound interpolation)

**Inputs:** $\mathcal{P}$, $b$, $\bar{Q}$ and $\bar{V}$
**Outputs:** upper bound $v^*$ and distribution $c^*$
$v \leftarrow \max_a \sum_s b(s)\bar{Q}_a(s)$
LP: $c^* \leftarrow \mathrm{argmin}_c \sum_{\bar{b}\in dom(\bar{V})} c(\bar{b})\bar{V}(\bar{b})$
s.t. $\sum_{\bar{b}\in dom(\bar{V})} c(\bar{b})\bar{b}(s) = b(s) \;\; \forall s$
$c(\bar{b}) \geq 0 \;\; \forall \bar{b} \in dom(\bar{V})$
$v^* \leftarrow \min(v, \sum_{\bar{b}\in dom(\bar{V})} c^*(\bar{b})\bar{V}(\bar{b}))$

**Algorithm 5** UB (sawtooth upper bound interpolation)

**Inputs:** $\mathcal{P}$, $b$, $\bar{Q}$ and $\bar{V}$
**Outputs:** upper bound $v^*$ and distribution $c^*$
$v \leftarrow \max_a \sum_s b(s)\bar{Q}_a(s)$
**for** each $\bar{b} \in dom(\bar{V}) \setminus \{e_s | s \in \mathcal{S}\}$ **do**
$c(\bar{b}) \leftarrow min_s b(s)/\bar{b}(s)$
$f(\bar{b}) \leftarrow \bar{V}(\bar{b}) - \sum_s \bar{b}(s)\bar{V}(e_s)$
**end for**
$\bar{b}^* \leftarrow \mathrm{argmin}_{\bar{b}}\, c(\bar{b})f(\bar{b})$
$v^* \leftarrow \min(v, c(\bar{b}^*)f(\bar{b}^*) + \sum_s b(s)\bar{V}(e_s))$
$c^*(e_s) \leftarrow b(s) - \sum_{\bar{b}\in dom(\bar{V})\setminus\{e_s|s\in\mathcal{S}\}} \bar{b}(s)c(\bar{b}) \;\; \forall s$
$c^*(\bar{b}^*) \leftarrow c(\bar{b}^*)$ and $c^*(\bar{b}) \leftarrow 0 \;\; \forall \bar{b} \neq \bar{b}^*$

value function. The algorithm gradually increases a lower bound by point-based value iteration similar to previous techniques. It distinguishes itself from previous algorithms in the upper bound computation and the exploration technique. The upper bound $\bar{V}$ is gradually decreased by finding belief points for which the upper bound is not tight and adding them to the domain of $\bar{V}$. Each time some new belief-bound pairs are added to $\bar{V}$, the reduction is propagated to other reachable beliefs. This can be done efficiently by constructing an augmented POMDP and computing the fast informed bound of this augmented POMDP. As a result, we do not need to interpolate between the belief-bound pairs of $\bar{V}$ too often and using LP-interpolation instead of sawtooth interpolation does not make a big difference in the overall running time.

GapMin (Alg. 6) executes four major steps repeatedly: a) it finds belief points $\underline{\mathcal{B}}'$ at which the lower bound is not optimal and belief-bound pairs $\bar{\mathcal{V}}'$ that improve the upper bound, b) point-based value iteration is then performed to update the set $\Gamma$ of $\alpha$-vectors that represent the lower bound, c) an augmented POMDP $\mathcal{P}'$ is constructed with the new belief-bound pairs and d) the improvements induced by the new belief-bound pairs are propagated throughout the upper bound by computing the fast informed bound of $\mathcal{P}'$. GapMin is reminiscent of policy iteration techniques in the sense that it alternates between finding beliefs at which the bounds can be improved and then propagating the improvements through the bounds by policy evaluation-like techniques. Similar to HSVI2 and SARSOP, the bounds in GapMin are also guaranteed to converge to the optimal value function in the limit.

Alg. 7 describes a search for beliefs at which the lower or upper bound is not tight. This search is done in a breadth-

**Algorithm 6** Gap minimization

**Inputs:** $\mathcal{P}$
**Output:** lower bound $\Gamma$ and upper bound $\bar{Q}, \bar{V}$
$\bar{Q} \leftarrow$ fastInformedBound($\mathcal{P}$)
$\bar{V} \leftarrow \{\langle e_s, \max_a \bar{Q}_a(s)\rangle | s \in \mathcal{S}\}$
$\Gamma \leftarrow$ blindStrategies($\mathcal{P}$)
$\underline{\mathcal{B}} \leftarrow \emptyset$ and $\bar{\mathcal{B}} \leftarrow \emptyset$
**repeat**
$[\underline{\mathcal{B}}', \bar{V}'] \leftarrow$ suboptimalBeliefs($\mathcal{P}, \bar{Q}, \bar{V}, \Gamma$)
$\underline{\mathcal{B}} \leftarrow \underline{\mathcal{B}} \cup \underline{\mathcal{B}}'$
$\Gamma \leftarrow$ pointBasedValueIteration($\mathcal{P}, \underline{\mathcal{B}}$)
$\bar{V} \leftarrow \bar{V} \cup \bar{V}'$
$\mathcal{P}' \leftarrow$ augmentedPOMDP($\mathcal{P}, \bar{Q}, \bar{V}$)
$\bar{Q}' \leftarrow$ fastInformedBound($\mathcal{P}'$)
$\bar{Q}_a(s) \leftarrow \bar{Q}'_a(e_s) \;\; \forall as$
$\bar{V}(b) \leftarrow \max_a \bar{Q}'_a(b) \;\; \forall b \in dom(\bar{V})$
**until** convergence

first manner with a priority queue that ranks beliefs according to a score that measures the gap between the upper and lower bound at the belief weighted by the probability of reaching this belief. In contrast, HSVI2 and SARSOP perform their search in a depth-first manner, which tends to find beliefs that are deeper, but less significant for the overall bounds. Hence, it is often the case that fewer beliefs are needed to construct equally tight bounds when the beliefs are found by a breadth-first search. The search selects actions according to a one-step lookahead with the upper bound $\bar{V}$. This is the same action selection strategy as for HSVI2 and SARSOP, which ensures that actions are tried until they become suboptimal. This guarantees that upper and lower bounds will converge to the optimal value function in the limit. The beliefs reached based on each observation are scored by measuring the gap between the upper and lower bound weighted by the probability of reaching that belief. The beliefs with a gap lower than some tolerance threshold (adjusted based on the discount factor and search depth) are discarded since their contribution to the gap of the initial belief is negligible. The remaining beliefs are inserted in the priority queue in order of decreasing score. At each visited belief, we verify whether the lower and upper bounds can be tightened by a one-step look ahead search. The search terminates when the queue is empty or a predetermined number of suboptimal beliefs have been found. It returns a set $\underline{\mathcal{B}}$ of beliefs for which the lower bound can be improved and set $\bar{V}$ of belief-bound pairs that improve the upper bound. The priority queue ensures that beliefs are examined in decreasing order of potential contribution to the gap of the initial belief.

Given a set of belief-bound pairs $\bar{V}$, we can propagate any improvement to the upper bound by repeatedly computing the following update for each $\langle b, v\rangle \in \bar{V}$:

$$v = \max_a R_a(b) + \gamma \sum_o \Pr(o|b,a)\mathit{UB}(b_{ao}, \bar{Q}, \bar{V})$$

However, notice that the number of calls to the upper bound interpolation function *UB* is $|\mathcal{A}||\mathcal{O}|$ per update and to fully

**Algorithm 7** Suboptimal Beliefs

**Inputs:** $\mathcal{P}$, $\bar{Q}$, $\bar{V}$, $\Gamma$ and $tolerance$
**Output:** lower bound beliefs $\underline{\mathcal{B}}$ and upper bound $\bar{V}$
$score \leftarrow \max_{\alpha\in\Gamma} \alpha(b_{ao}) - UB(b, \bar{Q}, \bar{V})$
$queue \leftarrow \{\langle b, gap, 1, 0\rangle\}$
**while** $queue \neq \emptyset$ **do**
  $\langle b, score, prob, depth\rangle \leftarrow$ pop($queue$)
  $a^* \leftarrow \text{argmax}_a R_a(b) + \gamma \sum_o \Pr(o|b,a) UB(b_{ao}, \bar{Q}, \bar{V})$
  $\overline{val}^* \leftarrow R_{a^*}(b) + \gamma \sum_o \Pr(o|b,a^*) UB(b_{a^*o}, \bar{Q}, \bar{V})$
  $\overline{val} \leftarrow UB(b, \bar{Q}, \bar{V})$
  **if** $\overline{val} - \overline{val}^* > tolerance$ **then**
    $\bar{V} \leftarrow \bar{V} \cup \{\langle b, \overline{val}^*\rangle\}$
  **end if**
  $\underline{val}^* \leftarrow R_{a^*}(b) + \gamma \sum_o \Pr(o|b,a^*) \max_{\alpha\in\Gamma} \alpha(b_{a^*o})$
  $\underline{val} \leftarrow \max_{\alpha\in\Gamma} \alpha(b)$
  **if** $\underline{val}^* - \underline{val} > tolerance$ **then**
    $\underline{\mathcal{B}} \leftarrow \underline{\mathcal{B}} \cup \{b\}$
  **end if**
  $depth \leftarrow depth + 1$
  **for** each $o \in \mathcal{O}$ **do**
    $gap \leftarrow \max_{\alpha\in\Gamma} \alpha(b_{ao}) - UB(b_{a^*o}, \bar{Q}, \bar{V})$
    **if** $\gamma^{depth} gap > tolerance$ **then**
      $prob_o \leftarrow prob \Pr(o|b, a^*)$
      $score \leftarrow prob_o \gamma^{depth} gap$
      $queue \leftarrow$ insert($queue$, $\langle b_{a^*o}, score, prob_o, depth\rangle$)
    **end if**
  **end for**
**end while**

propagate an improvement we may need to compute thousands' of updates. When the interpolation is done by linear programming, this is quite expensive, which is why HSVI2 and SARSOP use the sawtooth interpolation procedure. We follow an alternative approach (Hauskrecht 2000) that drastically reduces the number of calls to the interpolation function.

We noticed that we repeatedly make calls to the interpolation for the same beliefs $b_{ao}$ and that the optimal convex combination $c^*$ returned by UB tends to be the same even when the input $\bar{V}$ changes. Since $\bar{V}$ induces a convex function, $c^*$ is always a convex combination of beliefs that form a small convex hull of the desired belief. While $\bar{V}$ changes, the resulting convex combination rarely changes. Hence, one can cache the resulting $c^*$ for each call to UB (one call for each $b_{ao}$). Given $c^*$, we can quickly compute $v^* = \sum_{\langle b,v\rangle\in\bar{V}} c^*(b)v$, which is computationally negligible in comparison to solving an LP. We can then quickly propagate improvements at the cost of only one LP-interpolation per belief $b_{ao}$. The propagation won't be as good as if we resolved the LP for each interpolation, but it is very close in practice. Note that the LPs are resolved periodically (after each search for new beliefs where the bounds are not tight), so this does not affect the asymptotic convergence of the bounds to the optimal value function.

It turns out that propagating improvements with

**Algorithm 8** Augmented POMDP

**Inputs:** $\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \gamma, b_0\rangle$, $\bar{Q}$ and $\bar{V}$
**Output:** $\mathcal{P}' = \langle \mathcal{S}', \mathcal{A}, \mathcal{O}, TZ', R', \gamma, b_0'\rangle$
$\mathcal{S}' \leftarrow dom(\bar{V})$
**for** each $b \in \mathcal{S}'$, $a \in \mathcal{A}$, $o \in \mathcal{O}$ **do**
  $[val, c] \leftarrow UB(b_{ao}, \bar{Q}, \bar{V})$
  $TZ'(b', b, a, o) \leftarrow c(b') \Pr(o|b, a) \;\; \forall b' \in \mathcal{S}'$
**end for**
$R'_a(b) \leftarrow \sum_s b(s) R_a(s) \;\; \forall a \in \mathcal{A}, b \in \mathcal{S}'$
$\langle val, b_0'\rangle \leftarrow UB(b_0, \bar{Q}, \bar{V})$

LP caching is equivalent to solving a discrete belief MDP (Lovejoy 1991; Hauskrecht 2000). The interpolation essentially re-maps each $b_{ao}$ to a convex combination of beliefs in the domain of $\mathcal{V}$. Since the domain of $\bar{V}$ always contains the extreme points of the belief simplex, which correspond to each state, we can view this belief MDP as an augmented POMDP with additional states corresponding to the interior beliefs of the domain of $\mathcal{V}$. Alg. 8 describes how to construct this augmented POMDP. The combined transition and observation function *TZ* is obtained by the convex combination of each reachable belief $b_{ao}$ according to $\bar{V}$. Finally, we perform the propagation of the improvements by computing the fast informed bound of this augmented POMDP according to Alg. 3.

# 4 Experiments

We experimented with the suite of benchmark problems posted on Cassandra's POMDP website.[2] Out of the 68 problems, we discarded four of them (1d.noisy, 4x4.95, baseball and bulkhead.A) due to parsing issues and report results for the remaining 64 problems. Whenever the discount factor was 1, we changed it to 0.999 and whenever there was no start belief, we set it to a uniform distribution over the entire state space. We compare GapMin with sawtooth (ST) and LP interpolation to HSVI2 and SARSOP by running the implementations provided in the ZMDP[3] and APPL[4] packages.

We ran the four algorithms on each problem to compare the quality of the lower and upper bounds as well as the size of their representations. Each run was terminated as soon as the gap between the lower and upper bound was less than one unit at the 3rd significant digit or when 1000 seconds was reached. GapMin found a near optimal policy (gap less than one unit at the third significant digit) for 46 problems (out of 64) in comparison to 32 for HSVI2 and 31 for SARSOP. In Tables 1 and 2, we report the results for the 33 problems that were not solved (near) optimally by *all* solvers. For each problem, (near) optimal gaps are highlighted and when none of the techniques find a (near) optimal gap, the smallest gap is highlighted. Among the 18 problems that were not solved (near) optimally by *any* solver, GapMin with LP interpolation found the smallest gap for 7 problems in compar-

[2]http://www.cassandra.org/pomdp/index.shtml
[3]http://www.cs.cmu.edu/~trey/zmdp/
[4]http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/

Table 1: Results: comparison of the gap, lower bound (LB), upper bound (UB), # of $\alpha$-vectors ($|\Gamma|$) to represent the lower bound, # of belief-bound pairs ($|V|$) to represent the upper bound and time (seconds) for runs terminated after 1000 seconds or when the gap is less than one unit at the 3rd significant digit.

| problem | algorithm | gap | LB | UB | $\|\Gamma\|$ | $\|V\|$ | time |
|---|---|---|---|---|---|---|---|
| aloha.10 | hsvi2 | 9.0 | 535.4 | 544.4 | 4729 | n.a. | 997 |
| $\|\mathcal{S}\| = 30$ | sarsop | 9.5 | 535.2 | 544.7 | 48 | 2151 | 1000 |
| $\|\mathcal{A}\| = 9, \|\mathcal{O}\| = 3$ | gapMin ST | 10.3 | 534.1 | 544.4 | 136 | 510 | 673 |
| $\gamma = 0.999$ | **gapMin LP** | **7.6** | 536.5 | 544.2 | 152 | 383 | 968 |
| aloha.30 | **hsvi2** | **38** | 1212 | 1249 | 2062 | n.a. | 1000 |
| $\|\mathcal{S}\| = 90$ | sarsop | 74 | 1177 | 1252 | 86 | 1245 | 999 |
| $\|\mathcal{A}\| = 29, \|\mathcal{O}\| = 3$ | gapMin ST | 113 | 1136 | 1249 | 44 | 701 | 800 |
| $\gamma = 0.999$ | gapMin LP | 111 | 1136 | 1247 | 46 | 442 | 799 |
| cheng.D3-1 | hsvi2 | 11 | 6417 | 6428 | 16 | n.a. | 997 |
| $\|\mathcal{S}\| = 3$ | sarsop | 15 | 6417 | 6432 | 10 | 1836 | 1000 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | **gapMin ST** | **10** | 6412 | 6422 | 8 | 33 | 26 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 6412 | 6422 | 8 | 8 | 25 |
| cheng.D3-2 | **hsvi2** | **10** | 8240 | 8250 | 8 | n.a. | 404 |
| $\|\mathcal{S}\| = 3$ | sarsop | 12 | 8240 | 8252 | 6 | 866 | 1000 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | **gapMin ST** | **10** | 8235 | 8245 | 3 | 21 | 15 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 8235 | 8245 | 3 | 7 | 22 |
| cheng.D3-3 | hsvi2 | 105 | 7457 | 7562 | 13 | n.a. | 991 |
| $\|\mathcal{S}\| = 3$ | sarsop | 129 | 7457 | 7585 | 8 | 2437 | 999 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | **gapMin ST** | **10** | 7452 | 7462 | 7 | 149 | 56 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 7452 | 7462 | 7 | 15 | 37 |
| cheng.D3-4 | hsvi2 | 41 | 5827 | 5868 | 15 | n.a. | 993 |
| $\|\mathcal{S}\| = 3$ | sarsop | 48 | 5827 | 5875 | 5 | 1799 | 1000 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | **gapMin ST** | **10** | 5822 | 5832 | 8 | 65 | 78 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 5822 | 5832 | 5 | 16 | 37 |
| cheng.D3-5 | hsvi2 | 26 | 8673 | 8698 | 63 | n.a. | 990 |
| $\|\mathcal{S}\| = 3$ | sarsop | 34 | 8673 | 8706 | 10 | 2704 | 1000 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | **gapMin ST** | **10** | 8668 | 8678 | 9 | 28 | 34 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 8668 | 8678 | 10 | 8 | 15 |
| cheng.D4-1 | hsvi2 | 167 | 6715 | 6882 | 19 | n.a. | 999 |
| $\|\mathcal{S}\| = 4$ | sarsop | 180 | 6715 | 6894 | 10 | 6222 | 1000 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 4$ | **gapMin ST** | **10** | 6710 | 6720 | 11 | 476 | 553 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 6711 | 6721 | 11 | 45 | 288 |
| cheng.D4-2 | hsvi2 | 63 | 8381 | 8443 | 22 | n.a. | 995 |
| $\|\mathcal{S}\| = 4$ | sarsop | 71 | 8378 | 8450 | 8 | 2321 | 999 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 4$ | **gapMin ST** | **10** | 8376 | 8386 | 12 | 323 | 135 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 8376 | 8386 | 13 | 48 | 115 |
| cheng.D4-3 | hsvi2 | 55 | 7661 | 7715 | 20 | n.a. | 997 |
| $\|\mathcal{S}\| = 4$ | sarsop | 60 | 7660 | 7721 | 11 | 4720 | 1000 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 4$ | **gapMin ST** | **10** | 7656 | 7666 | 10 | 144 | 91 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 7656 | 7666 | 10 | 37 | 68 |
| cheng.D4-4 | hsvi2 | 65 | 7670 | 7735 | 18 | n.a. | 997 |
| $\|\mathcal{S}\| = 4$ | sarsop | 69 | 7669 | 7738 | 6 | 1371 | 1000 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 4$ | **gapMin ST** | **10** | 7665 | 7675 | 16 | 362 | 313 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 7665 | 7675 | 11 | 40 | 109 |
| cheng.D4-5 | hsvi2 | 91 | 7884 | 7975 | 35 | n.a. | 994 |
| $\|\mathcal{S}\| = 4$ | sarsop | 96 | 7884 | 7980 | 14 | 2584 | 1000 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 4$ | **gapMin ST** | **10** | 7879 | 7889 | 19 | 453 | 415 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 7879 | 7889 | 17 | 46 | 197 |
| cheng.D5-1 | hsvi2 | 59 | 6549 | 6608 | 19 | n.a. | 996 |
| $\|\mathcal{S}\| = 5$ | sarsop | 64 | 6549 | 6613 | 9 | 3002 | 999 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | **gapMin ST** | **10** | 6544 | 6554 | 1 | 125 | 26 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 6544 | 6554 | 1 | 22 | 25 |
| cit | hsvi2 | 0.0951 | 0.7430 | 0.8381 | 3739 | n.a. | 975 |
| $\|\mathcal{S}\| = 284$ | **sarsop** | **0.0491** | 0.7909 | 0.8399 | 3108 | 1368 | 967 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$ | gapMin ST | 0.8378 | 0.0000 | 0.8378 | 1 | 123 | 802 |
| $\gamma = 0.990$ | gapMin LP | 0.8378 | 0.0000 | 0.8378 | 1 | 104 | 855 |
| ejs1 | hsvi2 | 7.8 | 421.3 | 429.1 | 13 | n.a. | 991 |
| $\|\mathcal{S}\| = 3$ | sarsop | 48.8 | 421.3 | 470.1 | 9 | 37237 | 1000 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 2$ | **gapMin ST** | **0.4** | 421.1 | 421.5 | 9 | 23 | 52 |
| $\gamma = 0.999$ | **gapMin LP** | **0.3** | 421.2 | 421.6 | 9 | 11 | 65 |
| ejs2 | hsvi2 | 91 | 1781 | 1872 | 8 | n.a. | 997 |
| $\|\mathcal{S}\| = 2$ | sarsop | 115 | 1781 | 1896 | 7 | 12629 | 1000 |
| $\|\mathcal{A}\| = 2, \|\mathcal{O}\| = 2$ | **gapMin ST** | **10** | 1777 | 1787 | 6 | 21 | 22 |
| $\gamma = 0.999$ | **gapMin LP** | **10** | 1776 | 1786 | 6 | 5 | 13 |
| ejs4 | hsvi2 | 20.2 | -133.6 | -113.4 | 7 | n.a. | 999 |
| $\|\mathcal{S}\| = 3$ | sarsop | 22.8 | -133.6 | -110.8 | 2 | 5107 | 1000 |
| $\|\mathcal{A}\| = 2, \|\mathcal{O}\| = 2$ | **gapMin ST** | **1.0** | -134.1 | -133.1 | 2 | 76 | 26 |
| $\gamma = 0.999$ | **gapMin LP** | **1.0** | -134.1 | -133.1 | 2 | 7 | 13 |
| fourth | hsvi2 | 0.3758 | 0.2416 | 0.6174 | 3345 | n.a. | 994 |
| $\|\mathcal{S}\| = 1052$ | **sarsop** | **0.3300** | 0.2875 | 0.6175 | 3595 | 888 | 975 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$ | gapMin ST | 0.6176 | 0.0000 | 0.6176 | 1 | 20 | 532 |
| $\gamma = 0.990$ | gapMin LP | 0.6176 | 0.0000 | 0.6176 | 1 | 21 | 669 |
| hallway2 | hsvi2 | 0.5250 | 0.3612 | 0.8862 | 2393 | n.a. | 997 |
| $\|\mathcal{S}\| = 92$ | sarsop | 0.5247 | 0.3737 | 0.8984 | 262 | 1519 | 992 |
| $\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 17$ | **gapMin ST** | **0.3718** | 0.4173 | 0.7891 | 294 | 460 | 940 |
| $\gamma = 0.950$ | gapMin LP | 0.4279 | 0.3621 | 0.7900 | 153 | 256 | 759 |
| hallway | hsvi2 | 0.250 | 0.945 | 1.195 | 1367 | n.a. | 996 |
| $\|\mathcal{S}\| = 60$ | sarsop | 0.210 | 0.995 | 1.206 | 456 | 1713 | 998 |
| $\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 21$ | **gapMin ST** | **0.078** | 1.008 | 1.086 | 290 | 549 | 765 |
| $\gamma = 0.950$ | gapMin LP | 0.085 | 1.003 | 1.089 | 159 | 299 | 845 |

Table 2: Results continued (1000 seconds limit).

| problem | algorithm | gap | LB | UB | $\|\Gamma\|$ | $\|V\|$ | time |
|---|---|---|---|---|---|---|---|
| iff | hsvi2 | 0.924 | 8.931 | 9.855 | 7134 | n.a. | 999 |
| $\|\mathcal{S}\| = 104$ | sarsop | 0.775 | 9.095 | 9.871 | 6811 | 1991 | 997 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 22$ | gapMin ST | 0.722 | 9.214 | 9.936 | 544 | 741 | 785 |
| $\gamma = 0.999$ | **gapMin LP** | **0.660** | 9.261 | 9.920 | 532 | 831 | 940 |
| learning.c2 | hsvi2 | 0.090 | 1.549 | 1.639 | 4082 | n.a. | 996 |
| $\|\mathcal{S}\| = 12$ | sarsop | 0.093 | 1.556 | 1.648 | 4903 | 2054 | 996 |
| $\|\mathcal{A}\| = 8, \|\mathcal{O}\| = 3$ | gapMin ST | 0.078 | 1.553 | 1.631 | 810 | 2038 | 893 |
| $\gamma = 0.999$ | **gapMin LP** | **0.024** | 1.558 | 1.582 | 470 | 582 | 885 |
| 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 |
| $\|\mathcal{A}\| = 12, \|\mathcal{O}\| = 3$ | gapMin ST | 0.214 | 2.442 | 2.655 | 446 | 1387 | 944 |
| $\gamma = 0.999$ | **gapMin LP** | **0.180** | 2.441 | 2.622 | 515 | 518 | 947 |
| 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 |
| $\|\mathcal{A}\| = 16, \|\mathcal{O}\| = 3$ | gapMin ST | 0.363 | 3.308 | 3.671 | 349 | 894 | 858 |
| $\gamma = 0.999$ | gapMin LP | 0.353 | 3.306 | 3.658 | 500 | 365 | 989 |
| machine | hsvi2 | 3.49 | 63.18 | 66.66 | 662 | n.a. | 982 |
| $\|\mathcal{S}\| = 256$ | sarsop | 3.57 | 63.18 | 66.75 | 150 | 2742 | 998 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 16$ | **gapMin ST** | **2.98** | 62.93 | 65.90 | 77 | 476 | 817 |
| $\gamma = 0.990$ | gapMin LP | 3.20 | 62.39 | 65.59 | 67 | 292 | 856 |
| milos-aaai97 | hsvi2 | 18.31 | 49.15 | 67.46 | 3965 | n.a. | 998 |
| $\|\mathcal{S}\| = 20$ | sarsop | 19.61 | 49.74 | 69.35 | 3699 | 4465 | 997 |
| $\|\mathcal{A}\| = 6, \|\mathcal{O}\| = 8$ | gapMin ST | 17.67 | 49.89 | 67.55 | 1212 | 1889 | 774 |
| $\gamma = 0.900$ | **gapMin LP** | **15.42** | 49.97 | 65.39 | 581 | 1144 | 730 |
| mit | hsvi2 | 0.0939 | 0.7910 | 0.8849 | 5539 | n.a. | 1000 |
| $\|\mathcal{S}\| = 204$ | sarsop | 0.0665 | 0.8189 | 0.8854 | 2820 | 1861 | 999 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$ | **gapMin ST** | **0.0388** | 0.8447 | 0.8835 | 152 | 143 | 806 |
| $\gamma = 0.990$ | gapMin LP | 0.0554 | 0.8279 | 0.8833 | 120 | 130 | 859 |
| pentagon | hsvi2 | 0.1920 | 0.6341 | 0.8261 | 4361 | n.a. | 997 |
| $\|\mathcal{S}\| = 212$ | **sarsop** | **0.1311** | 0.6962 | 0.8273 | 3196 | 1228 | 971 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$ | gapMin ST | 0.8258 | 0.0000 | 0.8258 | 1 | 191 | 990 |
| $\gamma = 0.990$ | gapMin LP | 0.8258 | 0.0000 | 0.8258 | 1 | 121 | 893 |
| query.s2 | hsvi2 | 4.2 | 490.7 | 495.0 | 1366 | n.a. | 992 |
| $\|\mathcal{S}\| = 9$ | sarsop | 5.5 | 490.7 | 496.3 | 113 | 2992 | 999 |
| $\|\mathcal{A}\| = 2, \|\mathcal{O}\| = 3$ | **gapMin ST** | **1.0** | 490.4 | 491.4 | 37 | 1916 | 224 |
| $\gamma = 0.990$ | **gapMin LP** | **1.0** | 490.5 | 491.5 | 31 | 212 | 57 |
| query.s3 | hsvi2 | 26.2 | 546.8 | 573.1 | 1203 | n.a. | 997 |
| $\|\mathcal{S}\| = 27$ | sarsop | 28.1 | 546.8 | 574.8 | 112 | 3132 | 999 |
| $\|\mathcal{A}\| = 3, \|\mathcal{O}\| = 3$ | gapMin ST | 10.8 | 546.7 | 557.5 | 154 | 4066 | 686 |
| $\gamma = 0.990$ | **gapMin LP** | **7.0** | 546.7 | 553.7 | 119 | 1323 | 706 |
| query.s4 | hsvi2 | 51.9 | 569.5 | 621.4 | 2846 | n.a. | 999 |
| $\|\mathcal{S}\| = 81$ | sarsop | 54.3 | 569.1 | 623.4 | 166 | 6782 | 1000 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 3$ | gapMin ST | 46.1 | 569.6 | 615.6 | 377 | 2601 | 958 |
| $\gamma = 0.990$ | **gapMin LP** | **43.2** | 569.5 | 612.7 | 169 | 921 | 939 |
| sunysb | **hsvi2** | **0.2396** | 0.5566 | 0.7963 | 4370 | n.a. | 997 |
| $\|\mathcal{S}\| = 300$ | sarsop | 0.3233 | 0.4748 | 0.7980 | 3537 | 1229 | 986 |
| $\|\mathcal{A}\| = 4, \|\mathcal{O}\| = 28$ | gapMin ST | 0.7962 | 0.0000 | 0.7962 | 1 | 99 | 930 |
| $\gamma = 0.990$ | gapMin LP | 0.7961 | 0.0000 | 0.7961 | 1 | 107 | 974 |
| tiger-grid | hsvi2 | 0.388 | 2.138 | 2.525 | 3394 | n.a. | 990 |
| $\|\mathcal{S}\| = 36$ | sarsop | 0.262 | 2.267 | 2.529 | 945 | 2165 | 997 |
| $\|\mathcal{A}\| = 5, \|\mathcal{O}\| = 17$ | **gapMin ST** | **0.106** | 2.296 | 2.402 | 386 | 506 | 912 |
| $\gamma = 0.950$ | gapMin LP | 0.132 | 2.271 | 2.402 | 255 | 435 | 923 |

ison to 5 problems for GapMin with sawtooth interpolation, 4 problems for SARSOP and 2 problems for HSVI2. In general, the GapMin variants require much fewer $\alpha$-vectors and belief-bound pairs to represent the upper and lower bounds than HSVI2 and SARSOP, which demonstrates the effectiveness of the breadth-first search.

Figure 1 compares the lower and upper bounds for long running times on 8 of the 18 problems that were not solved optimally by any of the solvers. The circles correspond to GapMin with LP interpolation, stars to GapMin with sawtooth interpolation, solid lines to HSVI2 and the dash-dotted lines to SARSOP. The GapMin variants clearly outperform HSVI2 and SARSOP on 5 of the problems (hallway, hallway2, machine, mit and tiger-grid). GapMin with LP interpolation also finds a tighter upper bound, but a slightly looser lower bound for iff. It did not perform well on cit and pentagon. This can be explained by the fact that the upper bound for those two problems was already quite good and most of the work was about tightening the lower bound for which GapMin has no advantage over HSVI2 and SARSOP.

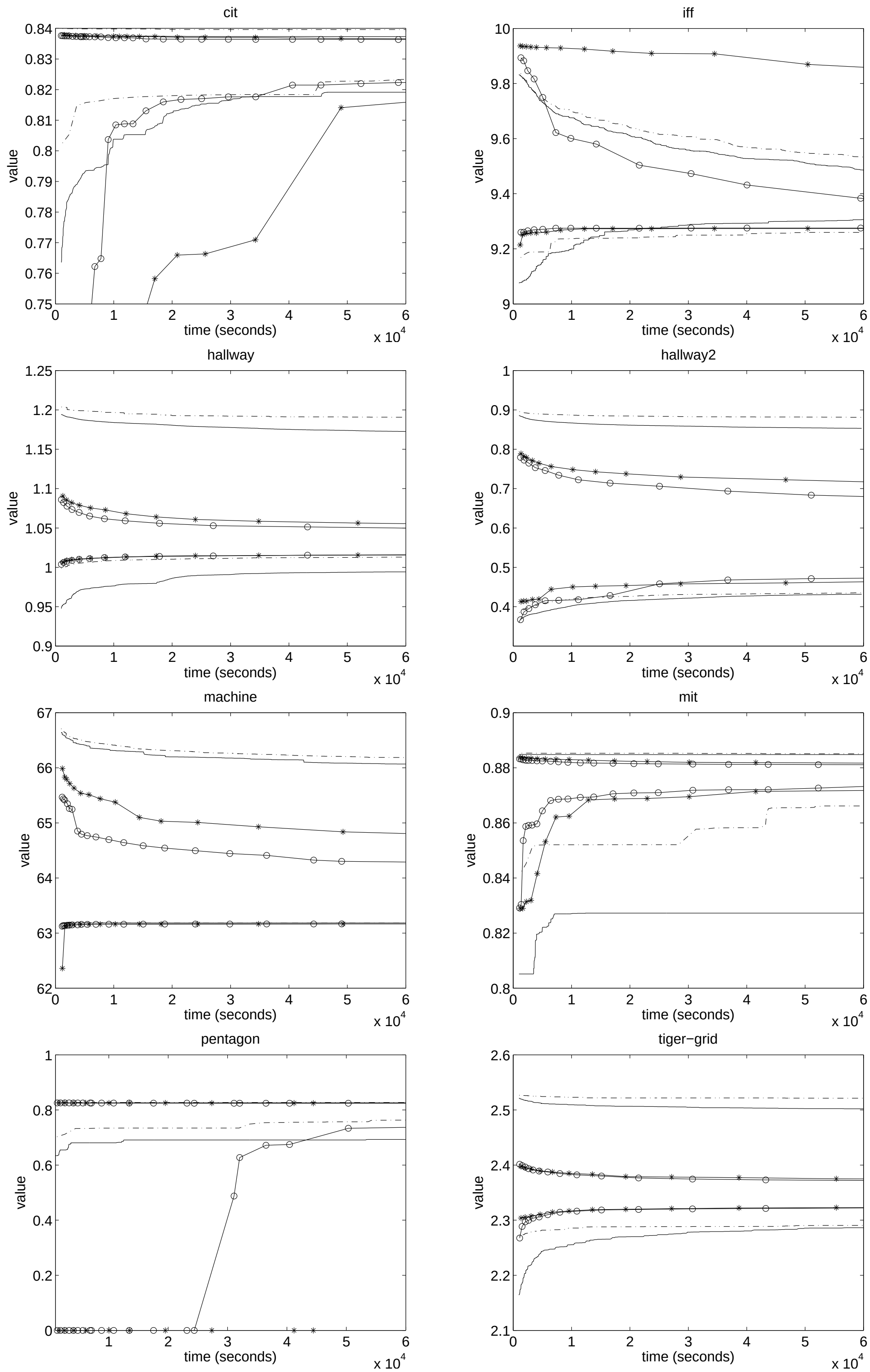


Figure 1: Comparison of lower and upper bounds for GapMin with LP interpolation (circles), GapMin with sawtooth interpolation (stars), HSVI2 (solid line) and SARSOP (dash-dotted line)

Table 3: Results with 50000 seconds limit.

| problem | algorithm | gap | LB | UB | $\lvert\Gamma\rvert$ | $\lvert V\rvert$ | time |
|---|---|---|---|---|---|---|---|
| cit | hsvi2 | 0.0182 | 0.8192 | 0.8373 | 29803 | n.a. | 49760 |
| $\lvert\mathcal{S}\rvert = 284$ | sarsop | 0.0169 | 0.8228 | 0.8396 | 21168 | 9337 | 49916 |
| $\lvert\mathcal{A}\rvert = 4, \lvert\mathcal{O}\rvert = 28$ | gapMin ST | 0.0226 | 0.8141 | 0.8367 | 739 | 681 | 48931 |
| $\gamma = 0.990$ | **gapMin LP** | **0.0149** | 0.8215 | 0.8364 | 648 | 614 | 45473 |
| hallway | hsvi2 | 0.179 | 0.994 | 1.173 | 15374 | n.a. | 49951 |
| $\lvert\mathcal{S}\rvert = 60$ | sarsop | 0.178 | 1.013 | 1.191 | 3053 | 12869 | 49992 |
| $\lvert\mathcal{A}\rvert = 5, \lvert\mathcal{O}\rvert = 21$ | gapMin ST | 0.043 | 1.015 | 1.058 | 947 | 2611 | 34828 |
| $\gamma = 0.950$ | **gapMin LP** | **0.036** | 1.016 | 1.051 | 851 | 1904 | 43184 |
| hallway2 | hsvi2 | 0.4211 | 0.4319 | 0.8530 | 18505 | n.a. | 49983 |
| $\lvert\mathcal{S}\rvert = 92$ | sarsop | 0.4482 | 0.4336 | 0.8818 | 1901 | 10908 | 49973 |
| $\lvert\mathcal{A}\rvert = 5, \lvert\mathcal{O}\rvert = 17$ | gapMin ST | 0.2620 | 0.4605 | 0.7225 | 1647 | 2809 | 46687 |
| $\gamma = 0.950$ | **gapMin LP** | **0.2256** | 0.4680 | 0.6936 | 1135 | 1798 | 36766 |
| iff | hsvi2 | 0.199 | 9.302 | 9.501 | 40984 | n.a. | 50000 |
| $\lvert\mathcal{S}\rvert = 104$ | sarsop | 0.290 | 9.259 | 9.549 | 54016 | 12237 | 49966 |
| $\lvert\mathcal{A}\rvert = 4, \lvert\mathcal{O}\rvert = 22$ | gapMin ST | 0.634 | 9.273 | 9.908 | 1614 | 4502 | 34472 |
| $\gamma = 0.999$ | **gapMin LP** | **0.156** | 9.275 | 9.431 | 1626 | 6231 | 40046 |
| machine | hsvi2 | 2.89 | 63.18 | 66.07 | 7857 | n.a. | 49998 |
| $\lvert\mathcal{S}\rvert = 256$ | sarsop | 3.02 | 63.18 | 66.20 | 996 | 22591 | 49963 |
| $\lvert\mathcal{A}\rvert = 4, \lvert\mathcal{O}\rvert = 16$ | gapMin ST | 1.67 | 63.17 | 64.84 | 139 | 3807 | 49261 |
| $\gamma = 0.990$ | **gapMin LP** | **1.14** | 63.17 | 64.30 | 173 | 1988 | 49036 |
| mit | hsvi2 | 0.0575 | 0.8273 | 0.8848 | 34461 | n.a. | 49942 |
| $\lvert\mathcal{S}\rvert = 204$ | sarsop | 0.0196 | 0.8655 | 0.8851 | 20662 | 12097 | 49616 |
| $\lvert\mathcal{A}\rvert = 4, \lvert\mathcal{O}\rvert = 28$ | gapMin ST | 0.0105 | 0.8714 | 0.8819 | 861 | 984 | 41564 |
| $\gamma = 0.990$ | **gapMin LP** | **0.0091** | 0.8721 | 0.8812 | 832 | 1051 | 43680 |
| pentagon | hsvi2 | 0.1349 | 0.6910 | 0.8258 | 29033 | n.a. | 49924 |
| $\lvert\mathcal{S}\rvert = 212$ | **sarsop** | **0.0702** | 0.7570 | 0.8271 | 21950 | 7534 | 49994 |
| $\lvert\mathcal{A}\rvert = 4, \lvert\mathcal{O}\rvert = 28$ | gapMin ST | 0.8249 | 0.0000 | 0.8249 | 1 | 713 | 44437 |
| $\gamma = 0.990$ | gapMin LP | 0.1497 | 0.6747 | 0.8244 | 425 | 846 | 40436 |
| tiger-grid | hsvi2 | 0.217 | 2.286 | 2.502 | 28182 | n.a. | 49948 |
| $\lvert\mathcal{S}\rvert = 36$ | sarsop | 0.231 | 2.290 | 2.522 | 5333 | 12504 | 49987 |
| $\lvert\mathcal{A}\rvert = 5, \lvert\mathcal{O}\rvert = 17$ | gapMin ST | 0.055 | 2.322 | 2.377 | 2404 | 3752 | 38675 |
| $\gamma = 0.950$ | **gapMin LP** | **0.052** | 2.321 | 2.373 | 2404 | 3778 | 43254 |

In Table 3, we report the size of the lower and upper bound representations found by the algorithms at 50000 seconds for the same 8 problems as in Fig. 1. The GapMin variants clearly find more compact representations than SARSOP and HSVI2. Also, GapMin with LP interpolation slightly outperforms GapMin with sawtooth interpolation.

Finally, we discuss the running time of GapMin. It is interesting to note that GapMin is implemented in Matlab and uses CPLEX to solve LPs where as HSVI2 and SARSOP are heavily optimized C implementation that avoid linear programs. Nevertheless GapMin performs very well as evident from the experiments. This can be explained by the fact that the breadth-first search finds more important beliefs for the bounds and therefore fewer $\alpha$-vectors and belief-bound pairs are necessary to represent the bounds. Furthermore, we reduced the number of upper bound interpolations, which allows linear programming to be used at a negligible cost while improving the tightness of the upper bounds.

## 5 Conclusion

In this paper, we described a new algorithm called GapMin that strives to compute tight upper and lower bounds of the optimal value function. It addresses the need for performance guarantees that practitioners often encounter. GapMin differs from previous state of the art point-based approaches by performing a prioritized breadth-first search, efficiently propagating upper bound improvements with an augmented POMDP and computing exact interpolations by linear programming. When tested on the suite of benchmark problems from Cassandra's POMDP website, GapMin found a near optimal solution (gap smaller than one unit at the third significant digit) in less than 1000 seconds for 46 problems (out of 64) in comparison to 32 problems for HSVI2 and 31 for SARSOP. GapMin also finds representations for the lower and upper bounds that are 1.5 to 50 times more compact than HSVI2 and SARSOP for the more difficult problems (Table 3). Our next step is to extend GapMin to factored POMDPs. The main issue is that LP interpolation yields linear programs with exponentially many variables and constraints. However, it should be possible to use column and constraint generation techniques similar to what has been done to tackle factored MDPs by linear programming (Guestrin et al. 2003; Schuurmans and Patrascu 2001).

## References


Brafman, R. I. 1997. A heuristic variable grid solution method for pomdps. In *AAAI*, 727–733.

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research* 19:399–468.

Hauskrecht, M. 1997. Incremental methods for computing bounds in partially observable Markov decision processes. In *AAAI*, 734–739.

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

Hoey, J.; Poupart, P.; von Bertoldi, A.; Craig, T.; Boutilier, C.; and Mihailidis, A. 2010. Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process. *Computer Vision and Image Understanding* 114(5):503–519.

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

Lovejoy, W. S. 1991. Computationally feasible bounds for partially observed markov decision processes. *Oper. Res.* 39:162–175.

Papadimitriou, C., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Math. Oper. Res.* 12:441–450.

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

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

Schuurmans, D., and Patrascu, R. 2001. Direct value-approximation for factored MDPs. In *NIPS*, 1579–1586.

Shani, G.; Brafman, R. I.; and Shimony, S. E. 2007. Forward search value iteration for POMDPs. In *IJCAI*, 2619–2624.

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

Smith, T., and Simmons, R. G. 2005. Point-based POMDP algorithms: Improved analysis and implementation. In *UAI*, 542–547.

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

Thomson, B., and Young, S. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language* 24(4):562–588.

Zhou, R., and Hansen, E. A. 2001. An improved grid-based approximation algorithm for pomdps. In *IJCAI*, 707–716.