Symbolic Heuristic Search Value Iteration for Factored POMDPs

Hyeong Seop Sim and **Kee-Eung Kim** and **Jin Hyung Kim**
Department of Computer Science, Korea Advanced Institute of Science and Technology
Daejeon, Korea

Du-Seong Chang and **Myoung-Wan Koo**
HCI Research Department, KT
Seoul, Korea

Abstract

We propose Symbolic heuristic search value iteration (Symbolic HSVI) algorithm, which extends the heuristic search value iteration (HSVI) algorithm in order to handle factored partially observable Markov decision processes (factored POMDPs). The idea is to use algebraic decision diagrams (ADDs) for compactly representing the problem itself and all the relevant intermediate computation results in the algorithm. We leverage Symbolic Perseus for computing the lower bound of the optimal value function using ADD operators, and provide a novel ADD-based procedure for computing the upper bound. Experiments on a number of standard factored POMDP problems show that we can achieve an order of magnitude improvement in performance over previously proposed algorithms.

Partially observable Markov decision processes (POMDPs) are widely used for modeling stochastic sequential decision problems with noisy observations. However, when we model real-world problems, we often need a compact representation since the model may result in a very large number of states when using the flat, table-based representation. Factored POMDPs (Boutilier & Poole 1996) are one type of such compact representation.

Given a factored POMDP, we have to design an algorithm that does not explicitly enumerate all the states in the model. For this purpose, it has gained popularity over the years to use the algebraic decision diagram (ADD) representation (Bahar *et al.* 1993) of all the vectors and matrices used in conventional POMDP algorithms. Hansen & Feng (2000) extended the Incremental Pruning algorithm (Cassandra, Littman, & Zhang 1997) to use ADDs. More recently, Poupart (2005) proposed the Symbolic Perseus algorithm, which is a point-based value iteration using ADDs. Symbolic Perseus was the primary motivation for our work to design a symbolic version of heuristic search value iteration (HSVI) algorithm (Smith & Simmons 2004; 2005) to handle factored POMDPs by using ADDs in a similar manner.

HSVI is also a point-based value iteration algorithm that recursively explores important belief points for approximating the optimal value function. For this purpose, HSVI computes both the lower as well as the upper bound of the value function in each recursive exploration. Our proposed algorithm (hereafter Symbolic HSVI) uses ADD operators to compute the upper and lower bound without explicitly enumerating the states and observations of the POMDP. We describe the implementation of core procedures and the experimental results on a number of benchmark factored POMDP problems.

Procedure: $\pi = \text{HSVI}(\epsilon)$
  Initialize $V_\oplus$ and $V_\ominus$
  while $V_\oplus(b_0) - V_\ominus(b_0) > \epsilon$ **do**
    explore$(b_0, V_\oplus, V_\ominus)$
  end while

Procedure: explore$(b, V_\oplus, V_\ominus)$
  if $V_\oplus(b) - V_\ominus(b) \leq \epsilon\gamma^{-t}$ **then**
    return
  end if
  $a^* \leftarrow \text{argmax}_a \, Q_{V_\oplus}(b, a)$
  $z^* \leftarrow \text{argmax}_z \, P(z|b, a^*)$
    $\cdot[V_\oplus(\tau(b, a^*, z)) - V_\ominus(\tau(b, a^*, z)) - \epsilon\gamma^{-t-1}]$
  explore$(\tau(b, a^*, z^*), V_\oplus, V_\ominus)$
  $V_\ominus \leftarrow V_\ominus \cup \text{backup}(b, V_\ominus)$
  $V_\oplus \leftarrow V_\oplus \cup \{(b, HV_\oplus(b))\}$

Procedure: $\alpha = \text{backup}(b, V_\ominus)$
  $\alpha_{a,z} \leftarrow \text{argmax}_{\alpha_\ominus \in V_\ominus}(\alpha_\ominus \cdot \tau(b, a, z))$
  $\alpha_a(s) \leftarrow R(s, a) + \gamma \sum_{z,s'} \alpha_{a,z}(s') O(s', a, z) T(s, a, s')$
  $\alpha \leftarrow \text{argmax}_{\alpha_a}(\alpha_a \cdot b)$

Figure 1: Top level pseudo-code of HSVI

Overview of POMDPs and HSVI

POMDPs model stochastic control problems with partially observable states. A POMDP is specified as a tuple $\langle S, A, Z, T, O, R, b_0 \rangle$: S is the set of states (s is a state); A is the set of actions (a is an action); Z is the set of observations (z is an observation); $T(s, a, s')$ is the transition probability of changing to state s' from state s by executing action a; $O(s, a, z)$ is the observation probability of observing z after executing action a and reaching in state s; $R(s, a)$ is the immediate reward of executing action a in state s; b_0 is the initial belief, which is the starting probability distribu-

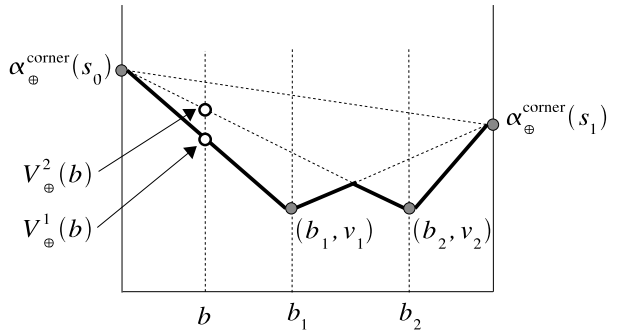

Figure 2: The sawtooth upper bound

tion on the states at the initial time step. Since the state is hidden, POMDP algorithms often make use of belief, which is the probability distribution b over the current states, for computing the policy π that maximizes the long-term discounted reward $V^{\pi}(b) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|b, \pi\right]$.

In the following subsections, we briefly overview the HSVI algorithm (a point-based POMDP algorithm) for the comprehensibility of presentation. We limit our discussion to the improved version of HSVI, namely HSVI2, which we used as the basis for Symbolic HSVI. For more details, we refer the readers to (Smith & Simmons 2004) for the original version of HSVI, and (Smith & Simmons 2005) for HSVI2. The outline of HSVI is shown in Figure 1.

Bound initialization

HSVI computes the initial lower bound by evaluating the value of blind policies of form "always execute action a". The evaluation of each blind policy is carried out by Markov decision process (MDP) policy evaluation:

$$\alpha_{\ominus,t+1}^{a}(s) = R(s,a) + \gamma \sum_{s'} T(s,a,s')\alpha_{\ominus,t}^{a}(s') \quad (1)$$

The initial lower bound is given by the set of all $\alpha_{\ominus}^{a}$'s for each action a: $V_{\ominus} = \{\alpha_{\ominus}^{a} | a \in A\}$.

For computing the initial upper bound, HSVI uses the fast informed bound method (Hauskrecht 2000):

$$\alpha_{\oplus,t+1}^{a}(s) = R(s,a) + \gamma \sum_{z} \max_{a'} \sum_{s'} T(s,a,s')O(s',a,z)\alpha_{\oplus,t}^{a'}(s') \quad (2)$$

Alternatively, the initial upper bound can be obtained by computing the value function of fully observable MDP, as in the original version of HSVI:

$$\alpha_{\oplus,t+1}^{a}(s) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} \alpha_{\oplus,t}^{a'}(s') \quad (3)$$

Once $\alpha_{\oplus}^{a}$'s are computed for each action a, the upper bound should be given by the set of all belief simplex corner points for each state s and their corresponding upper bound value $\alpha_{\oplus}^{\text{corner}}(s) = \max_a \alpha_{\oplus}^{a}(s)$. Hence, $V_{\oplus} = \{(e_s, \max_a \alpha_{\oplus}^{a}(s)) | s \in S\}$ where e_s is the unit vector with the s-th element $e_s(s) = 1$. However, as we will explain shortly, the initial upper bound is stored as a vector $\alpha_{\oplus}^{\text{corner}} = [\alpha_{\oplus}^{\text{corner}}(s_0), \ldots, \alpha_{\oplus}^{\text{corner}}(s_n)]$ in order to differentiate between the interior belief points ($b(s) < 1, \forall s$) and the corner belief points (*e.g.*, $b = e_s$).

Procedure: $V_{\oplus}(b)$
  $V_{\oplus}^{\text{interior}} = \{(b_i, v_i) | b_i$ is an interior point of the belief simplex, and v_i is the upper bound value at $b_i\}$
  $\alpha_{\oplus}^{\text{corner}} = [\alpha_{\oplus}^{\text{corner}}(s_0), \ldots, \alpha_{\oplus}^{\text{corner}}(s_n)]$
  $V_{\oplus}^{\text{corner}}(b) \leftarrow b \cdot \alpha_{\oplus}^{\text{corner}}$
  for $(b_i, v_i) \in V_{\oplus}^{\text{interior}}$ **do**
    $\phi \leftarrow \min\{b(s)/b_i(s) | s \in S, b_i(s) > 0\}$
    $V_{\oplus}^{i}(b) \leftarrow V_{\oplus}^{\text{corner}}(b) + \phi(v_i - b_i \cdot \alpha_{\oplus}^{\text{corner}})$
  end for
  $V_{\oplus}(b) = \min_i V_{\oplus}^{i}(b)$

Figure 3: Pseudo-code for the sawtooth upper bound

Upper and lower bound update

HSVI uses the set of α-vectors for representing the lower bound, as with the case of most POMDP algorithms. The lower bound is updated by the backup operation, which is the standard point-based Bellman update as given in Figure 1; $\tau(b, a, z)$ is the successor of belief point b, which will be defined in the next section.

The upper bound is represented as the set of pairs $(b, V_{\oplus}(b))$, so that $V_{\oplus}(b)$ is guaranteed to be always greater than or equal to the true optimal value at belief point b. Given such set of pairs, the upper bound is defined by the convex hull of $V_{\oplus}(b)$'s. Although it requires linear programming to find the exact value for the upper bound, HSVI instead uses the approximation of the estimate for the upper bound suggested by (Hauskrecht 2000). The idea is shown in Figure 2. Suppose that two belief points b_1 and b_2 are known to have upper bound values v_1 and v_2, respectively. Instead of computing the convex hull formed by v_1, v_2, and $\alpha_{\oplus}^{\text{corner}}$'s, HSVI uses the *sawtooth*-shaped upper bound shown as the thick solid line. To compute the upper bound value at a new belief point b, we compute the value predicted by one interior belief point and the corner points. For example, if the interior belief point is b_1, the predicted value is given by

$$V_{\oplus}^{1}(b) = b_1 \cdot \alpha_{\oplus}^{\text{corner}} + \min\left[\frac{b(s_0)}{b_1(s_0)}, \frac{b(s_1)}{b_1(s_1)}\right](v_1 - b_1 \cdot \alpha_{\oplus}^{\text{corner}})$$

following the formula for the proportional difference. We repeat for b_2, and the final upper bound value predicted from the sawtooth bound is

$$V_{\oplus}(b) = \min[V_{\oplus}^{1}(b), V_{\oplus}^{2}(b)]$$

Figure 3 shows the pseudo-code for computing the upper bound value based on the sawtooth bound.

The upper bound at belief point b is updated by operator H, defined as

$$Q_{V_{\oplus}}(b,a) = \sum_{s} R(s,a)b(s) + \gamma \sum_{z} P(z|b,a)V_{\oplus}(\tau(b,a,z))$$

$$HV_{\oplus}(b) = \max_a Q^{V_{\oplus}}(b,a) \quad (4)$$

where $P(z|b,a) = \sum_{s'} O(s',a,z) \sum_{s} b(s)T(s,a,s')$

Belief point exploration

Given the current upper and lower bounds of the optimal value ($V_{\oplus}$ and $V_{\ominus}$) and the belief point b, HSVI uses the

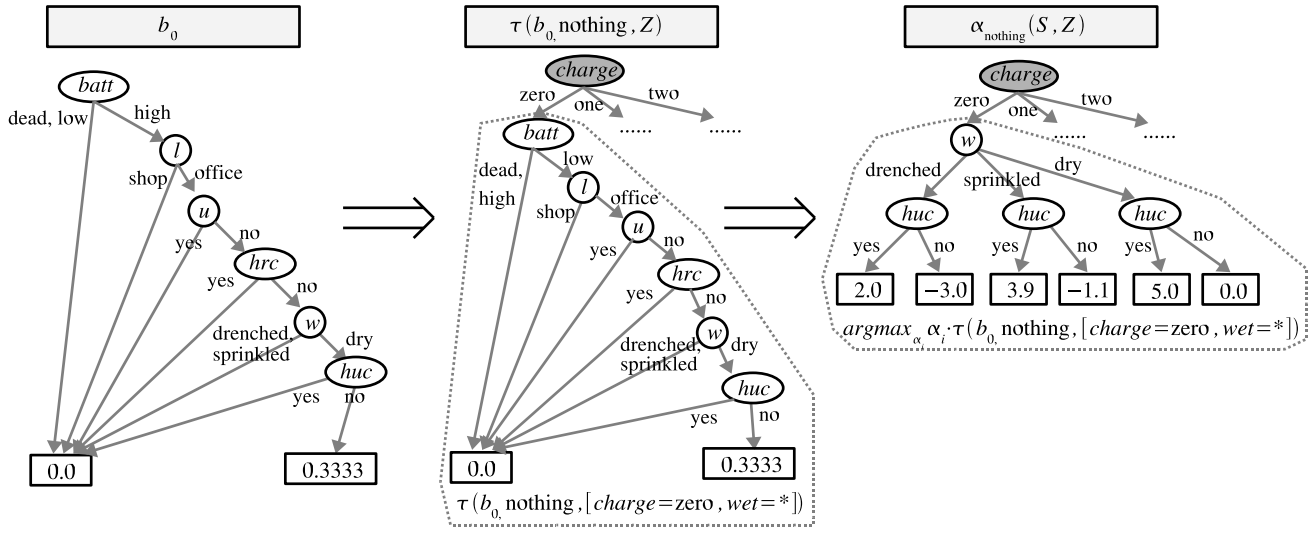

Figure 4: Calculating the successors of belief point and computing $\alpha_a(\mathcal{S}, \mathcal{Z})$. Since the whole ADD is too large to show in this paper, we show how the relevant operations are done on only one abstract observation $(charge = \text{zero})$.

following heuristic to explore the belief points (*i.e.*, select a belief point among successors of b). First, HSVI computes the upper bound for the value of executing action a in belief point b, which we denote as $Q_{V_\oplus}(b, a)$ given in Equation 4, and selects action a^* that yields the maximum upper bound value. Second, HSVI selects an observation z^* so that the difference of bounds is greatest at the successor belief point $b' = \tau(b, a^*, z^*)$ defined as

$$b'(s') = C \cdot O(s', a^*, z^*) \sum_s b(s) T(s, a^*, s')$$

where C is a normalizing constant so that $\sum_{s'} b'(s') = 1$.

Symbolic HSVI for Factored POMDPs

POMDP problems are traditionally represented in terms of matrices and vectors. However, real-world problems often exhibit large numbers of states and observations, and hence it is impractical to use such "flat" representation. Factored POMDPs (Boutilier & Poole 1996) use factored representation of states and observations by introducing state and observation variables. The idea is to compactly represent the transition probabilities, the observation probabilities, and the rewards in terms of these variables. We use ADDs (Bahar *et al.* 1993) to represent the problem, as well as all the relevant intermediate computation results used in the algorithm. For details on using ADDs in other POMDP algorithms, we refer the readers to (Hansen & Feng 2000) for the Incremental Pruning algorithm and (Poupart 2005) for the Perseus algorithm.

In this section, we explain our Symbolic HSVI algorithm, which extends HSVI algorithm to handle factored POMDPs. We will use the following notations for the ease of presentation: $\mathcal{S}$ for the set of state variables for the current time step, $\mathcal{S}'$ for the set of state variables for the next time step, and $\mathcal{Z}$ for the set of observation variables. We explicitly attach the variables to notations to signify the fact that a vector or a matrix is represented as an ADD, *e.g.*, $b(\mathcal{S})$ is a belief point represented as an ADD. Some of the ADD operators used frequently in the algorithm are the product $(\cdot)$, sum $(+)$, and existential abstraction $(\sum_{\mathcal{S}})$.

Bound initialization

The initializations of the lower and upper bounds are carried out using SPUDD (Hoey *et al.* 1999). Specifically, assuming an ADD representation for the $\alpha^a_{\ominus,t}(\mathcal{S})$, we use ADD product, existential abstraction, and sum operators for computing the lower bound in Equation 1:

$$\alpha^a_{\ominus,t+1}(\mathcal{S}) = R_a(\mathcal{S}) + \gamma \textstyle\sum_{\mathcal{S}'} T_a(\mathcal{S}, \mathcal{S}') \alpha^a_{\ominus,t}(\mathcal{S}')$$

Similarly, we can compute the upper bound by fast informed bound method in Equation 2 using ADD product, existential abstraction, sum, and max operators:

$$\alpha^a_{\oplus,t+1}(\mathcal{S}) = R_a(\mathcal{S}) + \gamma \sum_{\mathcal{Z}} \max_{a'} \sum_{\mathcal{S}'} T_a(\mathcal{S}, \mathcal{S}') O_{a'}(\mathcal{S}', \mathcal{Z}) \alpha^{a'}_{\oplus,t}(\mathcal{S}')$$

However, as noted in (Smith & Simmons 2005), fast informed bound does not significantly increase the performance of HSVI. We observed the same trend in Symbolic HSVI, so instead we implemented the fully observable MDP bound in Equation 3:

$$\alpha^a_{\oplus,t+1}(\mathcal{S}) = \max_a \left[R_a(\mathcal{S}) + \gamma \textstyle\sum_{\mathcal{S}'} T_a(\mathcal{S}, \mathcal{S}') \alpha^a_{\oplus,t}(\mathcal{S}') \right]$$

Upper and lower bound update

As with other ADD-based POMDP algorithms, Symbolic HSVI uses ADDs to represent α-vectors. The evaluation of lower bound at a belief point $b(\mathcal{S})$ is done by $\max_{\alpha_\ominus(\mathcal{S}) \in V_\ominus} \sum_{\mathcal{S}} b(\mathcal{S}) \alpha_\ominus(\mathcal{S})$ which involves ADD product and existential abstraction operators.

In order to establish the backup operator for updating the lower bound, we first compute the ADD of *all* successors from the current belief point $b(\mathcal{S})$ by executing action a:

$$\tau(b(\mathcal{S}), a, \mathcal{Z}) = \frac{\sum_{\mathcal{S}} T_a(\mathcal{S}, \mathcal{S}') O_a(\mathcal{S}', \mathcal{Z}) b(\mathcal{S})}{\sum_{\mathcal{S}, \mathcal{S}'} T_a(\mathcal{S}, \mathcal{S}') O_a(\mathcal{S}', \mathcal{Z}) b(\mathcal{S})}$$

Procedure: $V_{\oplus}(b(\mathcal{S}))$
$V_{\oplus}^{\text{interior}} = \{(b_i(\mathcal{S}), v_i) | b_i(\mathcal{S})$ is an interior point of the belief simplex, and v_i is the upper bound value at $b_i(\mathcal{S})\}$
$\alpha_{\oplus}^{\text{corner}}(\mathcal{S}) =$ upper bound from the initialization step
$V_{\oplus}^{0}(b(\mathcal{S})) \leftarrow \sum_{\mathcal{S}} b(\mathcal{S})\alpha_{\oplus}^{\text{corner}}(\mathcal{S})$
for $(b_i(\mathcal{S}), v_i) \in V_{\oplus}^{\text{interior}}$ **do**
  $\phi \leftarrow \min\{b(\mathcal{S})/b_i(\mathcal{S})\}$
  $V_{\oplus}^{i}(b(\mathcal{S})) \leftarrow V_{\oplus}^{0}(b(\mathcal{S})) + \phi(v_i - \sum_{\mathcal{S}} b_i(\mathcal{S})\alpha_{\oplus}^{\text{corner}}(\mathcal{S}))$
end for
$V_{\oplus}(b) = \min_i V_{\oplus}^{i}(b)$

Figure 5: Using ADDs for the sawtooth upper bound

where the fraction is ADD division operator. Note that $\tau(b(\mathcal{S}), a, \mathcal{Z})$ will contain observation variables in the diagram, hence it is effectively partitioning the observation space into a set of abstract observations that lead to the same successor from $b(\mathcal{S})$. Figure 4 shows an example of computing such successor ADD representing $\tau(b_0(\mathcal{S}), \text{nothing}, \mathcal{Z})$ for the coffee delivery problem (Poupart 2005).

Next, for each abstract observation z and the associated successor (which is also a belief point represented as ADD), we compute

$$\alpha_{a,z}(\mathcal{S}) \leftarrow \text{argmax}_{\alpha_{\ominus} \in V_{\ominus}}(\textstyle\sum_{\mathcal{S}} \alpha_{\ominus}(\mathcal{S})\tau(b(\mathcal{S}), a, z))$$

using the ADD product and existential abstraction operators. The results are combined to yield $\alpha_a(\mathcal{S}, \mathcal{Z})$, the ADD representing $\alpha_{a,z}$ for all observations (see the rightmost decision diagram in Figure 4). We finally compute

$$\alpha_a(\mathcal{S}) \rightarrow R_a(\mathcal{S}) + \gamma \textstyle\sum_{\mathcal{S}',\mathcal{Z}} T_a(\mathcal{S}, \mathcal{S}')O_a(\mathcal{S}, \mathcal{Z})\alpha_a(\mathcal{S}', \mathcal{Z})$$
$$\alpha(\mathcal{S}) \leftarrow \text{argmax}_{\alpha_a(\mathcal{S})} (\textstyle\sum_{\mathcal{S}} \alpha_a(\mathcal{S})b(\mathcal{S}))$$

using appropriate ADD operators.

The evaluation of the upper bound value at belief point $b(\mathcal{S})$ is also done using ADD operators. Note that we use ADD division operator to compute the minimum of belief proportions (ϕ) without explicitly enumerating the states (Figure 5).

The upper bound update operator H (Equation 4) is computed using ADDs similarly as in the backup operator: We first compute $\tau(b(\mathcal{S}), a, \mathcal{Z})$ to obtain the successors of $b(\mathcal{S})$ and the associated abstract observations, and evaluate the upper bound values. The results are combined to yield $V_{\oplus}(\tau(b(\mathcal{S}), a, \mathcal{Z}))$, an ADD with observation variables as intermediate nodes and upper bound values as leaf nodes. Once this is done, we compute

$$Q_{V_{\oplus}}(b(\mathcal{S}), a) = \textstyle\sum_{\mathcal{S}} R_a(\mathcal{S})b(\mathcal{S}) + \gamma \sum_{\mathcal{Z}} P(\mathcal{Z}|b(\mathcal{S}), a)V_{\oplus}(\tau(b(\mathcal{S}), a, \mathcal{Z}))$$
$$HV_{\oplus}(b(\mathcal{S})) = \max_a Q_{V_{\oplus}}(b(\mathcal{S}), a)$$

where $P(\mathcal{Z}|b(\mathcal{S}), a) = \sum_{\mathcal{S}'} O_a(\mathcal{S}', \mathcal{Z}) \sum_{\mathcal{S}} b(\mathcal{S})T_a(\mathcal{S}, \mathcal{S}')$.

Belief point exploration

Given the aforementioned ADD-based procedures for computing $Q_{V_{\oplus}}(b(\mathcal{S}), a)$, $V_{\oplus}(b(\mathcal{S}))$, and $V_{\ominus}(b(\mathcal{S}))$, we can select action a^* in a straight-forward way:

$$a^* = \text{argmax}_a Q_{V_{\oplus}}(b(\mathcal{S}), a)$$

In order to select $\tau(b(\mathcal{S}), a^*, z^*)$, we first compute $\tau(b(\mathcal{S}), a^*, \mathcal{Z})$, which is an ADD representing all possible successors from $b(\mathcal{S})$ by executing action a^* and the associated abstract observations. For each successor $b'(\mathcal{S})$, we compute $V_{\oplus}(b'(\mathcal{S}))$ and $V_{\ominus}(b'(\mathcal{S}))$, and calculate the excess uncertainty $V_{\oplus}(b'(\mathcal{S})) - V_{\ominus}(b'(\mathcal{S})) - \epsilon\gamma^{-t-1}$. The successor that maximizes the excess uncertainty weighted by the probability of the associated abstract observation is selected for the recursive exploration.

Additional performance optimization

HSVI additionally employs a number of techniques to speed up the algorithm. We mention two noteworthy techniques that are also used in symbolic HSVI: (1) HSVI caches the value of $Q_{V_{\oplus}}(b, a)$ for each explored belief point b in order to avoid re-computation of Equation 4 in the later exploration. For example, if a_1 had the greatest upper bound and a_2 had the second, if the updated $Q_{V_{\oplus}}(b, a_1)$ is greater than cached $Q_{V_{\oplus}}(b, a_2)$, there is no need to re-compute $Q_{V_{\oplus}}(b, a_2)$ for determining action; (2) HSVI aggressively prunes α-vectors by combining passive bounded pruning and pairwise pruning (Smith 2007).

Symbolic HSVI also adopts the factored belief approximation technique (Boyen & Koller 1998) used in symbolic Perseus; since the size of the ADD for the belief point can quickly become very large as recursive exploration depth increases, Symbolic HSVI breaks the correlations between state variables and represents the belief point as a product of independent marginals. This approximation technique has an additional benefit when we compute the upper bound value; we can obtain ϕ by calculating the product of the minimum of belief marginal ratios, instead of using the ADD division operator. For example, if we assume Boolean state variables,

$$\phi \leftarrow \prod_{s \in \mathcal{S}} \min \left[\frac{b(s)}{b_i(s)}, \frac{b(\overline{s})}{b_i(\overline{s})} \right]$$

where s represents the state variable s being true, and $\overline{s}$ being false.

Experiments

In this section, we report the experimental results of Symbolic HSVI on the following benchmark factored POMDP problems: Tiger (Kaelbling, Littman, & Cassandra 1998), Coffee Delivery and Handwashing (Poupart 2005), and n-City Ticketing (Williams, Poupart, & Young 2005) spoken dialogue management problem [1] with varying error rates of speech recognition $p_e = \{0.0, 0.1, 0.2\}$.

We compared the performance of Symbolic HSVI with the following POMDP algorithms: Perseus (Spaan & Vlassis 2005), HSVI (Smith & Simmons 2005) with the same set of performance optimizations used in Symbolic HSVI, and Symbolic Perseus (Poupart 2005). Since Perseus and HSVI cannot handle factored POMDPs, we prepared the equivalent flat versions for some of the problems (n-City Ticketing problems). In addition, since Symbolic HSVI was im-

[1] The original problem gives the penalty of 10 when issuing a wrong ticket, but here we give the penalty of 100.

Problem & Size ($\|S\|/\|A\|/\|Z\|; \|\mathcal{S}\|/\|\mathcal{Z}\|$)	R	Time	$[\epsilon_R]$ $\|\Gamma\|$
Tiger $(2/2/2; 1/1)$			$[\pm 0.42]$
Perseus	6.28	0.07	5
HSVI	6.28	0.03	5
Symbolic Perseus	6.23	0.13	5
Symbolic HSVI	6.23	0.06	5
Coffee Delivery $(432/10/6; 7/2)$			$[\pm 1.41]$
Symbolic Perseus	37.00	80.0	52
Symbolic HSVI	36.87	15.7	50
Handwashing $(180/6/6; 4/1)$			$[\pm 4.19]$
Symbolic Perseus	76.44	3.3	14
Symbolic HSVI	78.42	0.7	27
2-City Ticketing, $\mathbf{p_e = 0.0}$ $(397/10/11; 5/1)$			$[\pm 0.016]$
Perseus	8.74	4.9	4
HSVI	8.74	1.3	4
Symbolic Perseus	8.74	3.6	4
Symbolic HSVI	8.74	1.0	4
2-City Ticketing, $\mathbf{p_e = 0.1}$ $(397/10/11; 5/1)$			$[\pm 0.078]$
Perseus	7.41	6.5	9
HSVI	7.47	1.4	5
Symbolic Perseus	7.46	4.8	7
Symbolic HSVI	7.39	1.7	5
2-City Ticketing, $\mathbf{p_e = 0.2}$ $(397/10/11; 5/1)$			$[\pm 0.18]$
Perseus	6.92	3.7	8
HSVI	6.95	1.6	12
Symbolic Perseus	6.77	6.7	11
Symbolic HSVI	6.81	1.6	25
3-City Ticketing, $\mathbf{p_e = 0.0}$ $(1945/16/18; 5/1)$			$[\pm 0.023]$
Perseus	8.09	115.9	16
HSVI	8.09	55.6	18
Symbolic Perseus	8.09	48.1	14
Symbolic HSVI	8.09	0.9	13
3-City Ticketing, $\mathbf{p_e = 0.1}$ $(1945/16/18; 5/1)$			$[\pm 0.13]$
Perseus	6.06	637.4	93
HSVI	6.04	227.3	258
Symbolic Perseus	6.12	221.3	101
Symbolic HSVI	6.02	5.7	145
3-City Ticketing, $\mathbf{p_e = 0.2}$ $(1945/16/18; 5/1)$			$[\pm 0.24]$
Perseus	5.22	1260.1	150
HSVI	5.30	1072.9	626
Symbolic Perseus	5.25	707.1	163
Symbolic HSVI	5.30	10.5	200

Figure 6: Performance comparison results of Symbolic HSVI on factored POMDP problems. For the n-City Ticketing problems, we prepared the equivalent flat versions in order to test Perseus and HSVI. R is the average cumulative reward of 3000 simulations, T is the execution time in seconds until convergence, and $|\Gamma|$ is the number of α-vectors in the final result.

plemented in Java, for fair comparisons, we re-implemented other POMDP algorithms in Java using the published source code of each algorithm. All the performance evaluation is done on a Linux platform with the Intel Xeon 2.33GHz CPU and 8GB memory. Figure 6 shows the performance comparison results.

The execution times of the algorithms were gathered in the following way: for HSVI algorithms, we executed multiple times with various goal widths (ϵ) and obtained an estimate of the policy quality (cumulative reward) through simulations. When the estimate is within 95% confidence interval of the optimal, we stopped the algorithm and measured the execution time. For Perseus algorithms, we continued execution until the value function difference was within 0.001, and measured the shortest execution time by varying the number of initial belief points. Figure 7 shows the solution quality versus wallclock time for four of the problems.

In summary, Symbolic HSVI was able to achieve the same level of solution quality as other algorithms, while significantly reducing the execution time. The speedup was most distinct in large problems, specifically the 3-City Ticketing problems, where Symbolic HSVI was, on average, 68x faster than HSVI, 121x faster than Perseus, and 53x faster than Symbolic Perseus.

Conclusion and Future Work

In this paper, we proposed Symbolic HSVI, an extension of the HSVI algorithm to handle factored POMDPs by using ADDs. Experiments on a number of benchmark problems show that Symbolic HSVI is able to achieve an order of magnitude speedup over previously proposed POMDP algorithms in large POMDPs with factored representations.

Symbolic HSVI leverages existing techniques for using ADDs to update the lower bound, and provides a novel ADD-based procedure for computing the upper bound. The upper bound computation in Symbolic HSVI is particularly efficient when assuming the sawtooth approximation method and factored belief points. By combining the belief point exploration strategy of HSVI and the efficient representation of ADDs, Symbolic HSVI scales better to larger problems.

We are currently working on applying Symbolic HSVI to real-world scale dialogue management problems represented as factored POMDPs. They are similar to the n-City Ticketing problems in the task objectives (*i.e.*, gather information from noisy observations), however the state dynamics are obtained from a real corpus.

Symbolic HSVI could also be extended to handle factored action space representations. Using variables for the factored representation of action space has been previously studied for MDPs (Boutilier, Dean, & Hanks 1999) but not yet in sufficient depth for POMDPs. The ADD-based procedures in this paper for computing the upper and lower bound could be easily extended to factored action space as well.

Acknowledgments

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) grant (R01-2007-000-

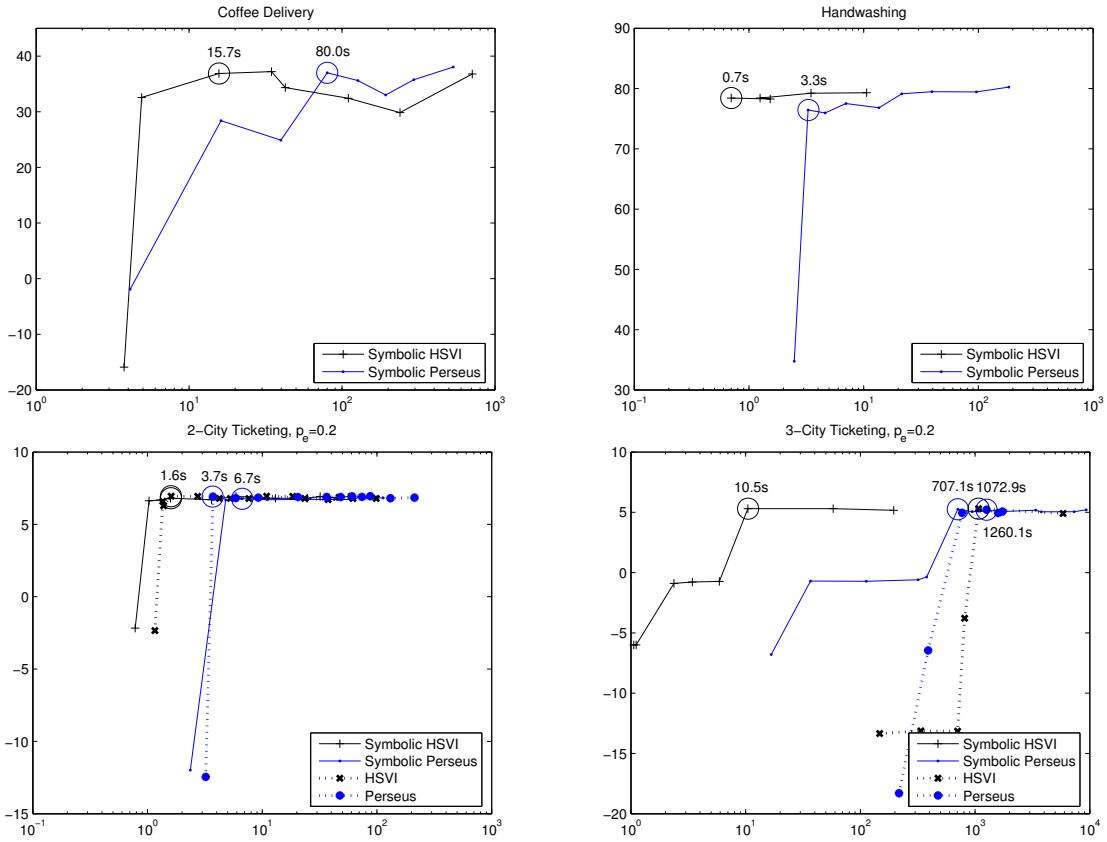

Figure 7: Solution quality versus wallclock time

21090-0) funded by the Korea government (MOST)

References

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proceedings of IEEE/ACM International Conference on CAD-1993*.

Boutilier, C., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of AAAI-1996*.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *Proceedings of UAI-1998*.

Cassandra, A.; Littman, M. L.; and Zhang, N. L. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of UAI-1997*.

Hansen, E. A., and Feng, Z. 2000. Dynamic programming for POMDPs using a factored state representation. In *Proceedings of AIPS-2000*.

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

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of UAI-1999*.

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

Poupart, P. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. Ph.D. Dissertation, University of Toronto.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of UAI-2004*.

Smith, T., and Simmons, R. 2005. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of UAI-2005*.

Smith, T. 2007. *Probabilistic Planning for Robotic Exploration*. Ph.D. Dissertation, Carnegie Mellon University.

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

Williams, J. D.; Poupart, P.; and Young, S. 2005. Factored partially observable Markov decision processes for dialogue management. In *Proceedings of IJCAI-2005 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.