Speeding up Online POMDP Planning
Unification of Observation Branches by Belief-state Compression via Expected Feature Values

Gavin Rens
Centre for Artificial Intelligence Research, University of KwaZulu-Natal, and CSIR Meraka, South Africa
grens@csir.co.za

Keywords: Online, POMDP, Planning, Heuristic, Optimization, Belief-state compression, Expected feature values

Abstract: A novel algorithm to speed up online planning in partially observable Markov decision processes (POMDPs) is introduced. I propose a method for compressing nodes in belief-decision-trees while planning occurs. Whereas belief-decision-trees branch on actions and observations, with my method, they branch only on actions. This is achieved by unifying the branches required due to the nondeterminism of observations. The method is based on the expected values of domain features. The new algorithm is experimentally compared to three other online POMDP algorithms, outperforming them on the given test domain.

1 INTRODUCTION

Partially observable Markov decision processes (POMDPs) (Monahan, 1982; Lovejoy, 1991) can generate optimal policies that account for stochastic actions and partially observable environments. They are also reward-based, accounting for an agent's preferences (to guide its behavior). Unfortunately, optimal POMDP policies are usually intractable to generate, and they are even less practical when used in dynamic environments.

A strategy for policy generation in dynamic environments that deals with this intractability is *continuous planning* or *agent-centered search* (Koenig, 2001). Agents employing this strategy compute future actions with only *local look-ahead* or *forward-search*. Recently, algorithms for online planning have been developed to deal with intractability of solving POMDPs (Ross et al., 2008).

Two sources for the intractability of solving POMDPs' optimally are usually cited in the literature (Pineau et al., 2003). First is the *curse of dimensionality*, which refers to (in the case of a model with discrete states) a belief space having a dimension equal to the number of states. For instance, a domain modeled with 1000 states has a 1000-dimensional belief space! Several researchers have suggested ways to compress the state space in attempts to deal with this problem (Poupart and Boutilier, 2003; Roy et al., 2005; Li et al., 2007). Second is the *curse of history*, which refers to the number of possible belief-states which must be considered during planning, increases exponentially with the planning horizon. Kurniawati et al. (2011) reduce the effective horizon in robot motion planning by using a particular (offline) point-based POMDP solver. He et al. (2011) tackle the horizon problem for online planning for large systems that need predictions for actions many steps into the future by using *macro-actions*. According to Rens and Ferrein (2013), exponential growth of belief-state size in the number of steps can be considered as a third source of potential intractability in POMDP algorithms—the *curse of outcomes*. They investigated several "fast and frugal" methods for reducing the size of belief-nodes generated during online planning. The most effective method was what they called the *mean-as-threshold* method (explained in § 3).

The search space in online POMDP planning can be thought of graphically as a finite tree (a *belief-decision-trees*), where nodes in alternating tiers represent (i) belief-states branching on the agent's choice of actions to perform, respectively, (ii) perceptions branching on the environment's choice of which observation to supply. An agent's belief-states can contain anything from a few states to thousands of states or more. Furthermore, given some action choice, every possible observation results in a new belief-state.

I introduce a novel approximate algorithm to speed up online POMDP planning by generating only a single most expected state for every action considered, instead of generating a belief-state for every possible observation for every action considered dur-

ing planning. The method unifies the branches which are due to the nondeterminism of perceptions, thus branching only on actions. The unification is based on the expected values of domain features. The new algorithm is experimentally compared to three other online POMDP algorithms, outperforming them on the given test domain—in some cases, significantly.

The required preliminaries are reviewed in Section 2. Section 3 presents the four algorithms which will be compared with each other, including the new proposed algorithm. In Section 4, I test the algorithms on a simple grid-world domain. Some concluding remarks are made in the last section.

2 PRELIMINARIES

In a partially observable Markov decision process (POMDP), the agent can only predict with a likelihood in which state it will end up after performing an action. And due to imperfect sensors, an agent must maintain a probability distribution over the set of possible states.

Formally (Kaelbling et al., 1998), a POMDP is a tuple $\langle S, A, T, R, Z, O, b^0 \rangle$ with a finite set of states $S = \{s_1, s_2, \ldots, s_n\}$, a finite set of actions $A = \{a_1, a_2, \ldots, a_k\}$, the *state-transition function*, where $T(s,a,s')$ is the probability of being in s' after performing action a in state s, the *reward function*, where $R(a,s)$ is the reward gained for executing a while in state s, a finite set of observations $Z = \{z_1, z_2, \ldots, z_m\}$; the *observation function*, where $O(s',a,z)$ is the probability of observing z in state s' resulting from performing action a in some other state; and b^0 is the initial probability distribution over all states in S.

A belief-state b is a set of pairs $\langle s, p \rangle$ where each state s in b is associated with a probability p. All probabilities must sum up to one, hence, b forms a probability distribution over the set S of all states. To update the agent's beliefs about the world, a state estimation function $SE(z,a,b) = b_n$ is defined as

$$b_n(s') = \frac{O(s',a,z) \sum_{s \in S} T(s,a,s') b(s)}{Pr(z|a,b)}, \qquad (1)$$

where a is an action performed in 'current' belief-state b, z is the resultant observation and $b_n(s')$ denotes the probability of the agent being in state s' in 'new' belief-state b_n. Note that $Pr(z \mid a,b)$ is a normalizing constant.

Let the *planning horizon* h (also called the *look-ahead depth*) be the number of future steps the agent plans ahead each time it selects its next action. $V^*(b,h)$ is the *optimal* value of future courses of actions the agent can take with respect to a finite horizon h starting in belief-state b. This function assumes that at each step, the action which will maximize the state's value will be selected. $V^*(b,h)$ is defined as

$$\max_{a \in A} \Big[\rho(a,b) + \gamma \sum_{z \in Z} Pr(z \mid a,b) V^*(SE(z,a,b), h-1) \Big],$$

where $\rho(a,b)$ is defined as $\sum_{s \in S} R(a,s) b(s)$, $0 \leq \gamma < 1$ is a factor to discount the value of future rewards and $Pr(z \mid a,b)$ denotes the probability of reaching belief-state $b_n = SE(z,a,b)$. While V^* denotes the optimal state value, function Q^* denotes the optimal action value: $Q^*(a,b,h) = \rho(a,b) + \gamma \sum_{z \in Z} Pr(z \mid a,b) V^*(SE(z,a,b), h-1)$ is the value of executing a in the current belief-state, plus the total expected value of belief-states reached thereafter.

Algorithm 1: Basic

```
   Input: b: belief-state, h: horizon
   Output: an action, the action's Q value
1  b': belief-state, initially empty
2  if h = 0 then
3      return stop, 0
4  if h > 0 then
5      maxVal ← −∞
6      foreach a ∈ A do
7          sum ← 0
8          foreach z ∈ Z do
9              b' ← SE(z,a,b)
10             _,V ← Basic(b',h−1)
11             sum ← sum + Pr(z | a,b)V
12         value ← ρ(a,b) + γ·sum
13         if value > maxVal then
14             maxVal ← value
15             bestAct ← a
16     return bestAct, maxVal
```

Algorithm 1 is the basic online POMDP algorithm to select the next best action. (_ at line 10 is a placeholder for the best action so far returned by the algorithm, but not used there.)

3 ONLINE POMDP ALGORITHMS

3.1 Mean as Threshold

The basic algorithm is augmented by compressing the belief-states in a very simple and intuitive way. The idea (Rens and Ferrein, 2013) is to reduce the size of a belief-state by retaining only a small number of representative states. As the number of states in a belief re-

duces, performing belief-update on the 'compressed' belief will be significantly faster.

For each belief-node generated in the belief-decision-tree, a *subset* of states with probabilities above a certain threshold are retained. Using the mean μ_b of the probabilities of the states in b as a threshold seems to be a reasonable heuristic for selecting states with probabilities *relatively* high compared to all the states in the node. They define the set $mp(b)$ of *most probable* states of a set b as $\{\langle s,p\rangle \in b \mid p \geq \mu_b\}$. The mean-as-threshold (MT) method is then to replace $b' \leftarrow SE(z,a,b)$ at line 9 of Algorithm 1 by $b' \leftarrow mt(SE(z,a,b))$, where

$$mt(b) = \{\langle s, p/\alpha\rangle \mid (s,p) \in mp(b), \alpha = \sum_{\langle s,p'\rangle \in mp(b)} p'\}.$$

3.2 Monte Carlo Sampling

Every action leads to a set of observations with a probability distribution related to the likelihood of the observation being perceived. Given an action, the belief-state in which it was performed and a possible observation, a new belief-state can be generated. Instead of expanding the belief-nodes for each observation associated with each action at each stage during planning, a relatively small subset of nodes can be generated from a relatively small subset of observations sampled.

In the approach of McAllester and Singh (1999), the probabilities $Pr(z \mid a,b)$ are approximated using the observed frequencies in the sample: Let $Z_c^S(a,b) = \{z_1, z_2, \ldots, z_c\}$ be c samples obtained from Z proportional to $Pr(\cdot \mid a,b)$. Let $N_z(Z_c^S(a,b))$ be the number of times observation z occurs in $Z_c^S(a,b)$. Then $\frac{N_z(Z_c^S(a,b))}{c}$ is an estimate of $Pr(z \mid a,b)$.

Whereas computing

$$Pr(z \mid a,b) = \sum_{s' \in S} O(s',a,z) \sum_{s \in S} T(s,a,s')b(s)$$

exactly is in $O(|S|^2)$, estimating it by looking at the frequency of z's occurrence in the set of samples is only in $O(log|S| + log|Z|)$. McAllester and Singh (1999) relate sample size c to the optimality of this approximation in a theorem. Determining a desirable value for c theoretically is beyond the scope of this paper. "[...] a few samples is often sufficient to obtain a good estimate as the observations that have the most effect on $Q^*(b,a)$ (i.e. those which occur with high probability) are more likely to be sampled" (Ross et al., 2008). In the experiments, I found that using even only one sample produces good returns. Planning time becomes impractical when using more than three or four samples.

A disadvantage of this approach is that actions cannot be pruned (Monte Carlo estimation is not guaranteed to correctly propagate the lower and upper bounds up the tree). Thus, each action executable in a belief-state must be considered. That means that the Monte Carlo approach may be difficult to apply in domains where the number of actions $|A|$ is large (Ross et al., 2008).

The Monte Carlo (MC) sampling approach is thus to replace lines 7 to 10 of Algorithm 1 by

foreach $z \in Z_S \mid N_z(Z_c^S(a,b)) > 0$ **do**
$b' \leftarrow SE(z,a,b)$
$\_, V \leftarrow \mathrm{MC}(b', h-1, c)$
$sum \leftarrow sum + \frac{N_z(Z_c^S(a,b))}{c} V$

where $\mathrm{MC}(\cdots)$ is the modified basic algorithm.

3.3 Real-Time Belief Space Search

Branch-and-Bound (BB) pruning is a general search technique used to prune nodes that are known to be suboptimal in the search tree, thus preventing the expansion of unnecessary lower nodes in lower parts of the tree. To achieve this, an estimate is maintained on the value $Q^*(a,b)$ of each action a, for every belief-node b in the tree. These estimates are computed by first evaluating an upper bound heuristic for the fringe nodes (at the fixed depth D) of the current tree. The bound is then propagated to parent nodes according to the following equation:

$$\delta(b,h) = \begin{cases} Hr(b), & \text{if } h = 0 \\ \Delta(b,h), & \text{otherwise} \end{cases}$$

with $\Delta(b,h)$ is defined as

$$\max_{a \in A} [\rho(a,b) + \gamma \sum_{z \in Z} Pr(z \mid a,b)\delta(SE(z,a,b), h-1)],$$

where heuristic function $Hr(b)$ is an over-optimistic estimate on $V^*(b)$. Similarly, heuristic function $Hr(a,b)$ is an over-optimistic estimate on $Q^*(a,b)$. Of course, $Hr(b)$ and $Hr(a,b)$ are domain-specific functions.

The idea behind Branch-and-Bound pruning is as follows. If a given action a in a belief b has an upper bound $Hr(a,b)$ which is lower than the current value of b determined so far using $\delta(b,h)$, then we know that a cannot be on the path yielding the best value, meaning that a is suboptimal in belief b. Hence, all belief-nodes reached from b via action a will be pruned from the tree (Ross et al., 2008). The heuristic function used in the experiments is

$$Hr(a,b) = \sum_{s \in S} Q^{MDP}(s,a)b(s),$$

where $Q^{MDP}(s,a)$ is the QMDP approximation—an upper bound on the Q-value of a in s. According to $V^*(b) = \max_a Q^*(a,b)$, the function $Hr(b)$ is then $Hr(b) = \max_{a \in A} Hr(a,b)$.

I implemented a version of Real-Time Belief Space Search (RTBSS) first proposed by Paquet et al. (2005). It follows the branch-and-bound strategy to prune suboptimal states from the tree. The 0 at line 3 of Algorithm 1 is replaced by $Hr(b)$. Replace **foreach** $a \in A$ at line 6 by **foreach** $a \in Sort(b,A)$. $Sort(b,A)$ is a list of the actions in $\{a_1, a_2, \ldots, a_{|A|}\}$, sorted such that if $i < j$, then $Hr(a_i,b) \geq Hr(a_j,b)$. Actions are sorted in this manner in an attempt to prune branches as soon as possible (Paquet et al., 2005). And finally, the procedure corresponding to lines 6 to 14 are executed only if $Hr(a,b) > maxVal$.

3.4 Observation Unification

The OUCEF algorithm (2) is my contribution. It came about due to the following insight. Suppose a person's beliefs about his/her current situation were comparable to a POMDP state. One could imagine that in a real-time/online setting, when considering future courses of action (including perceptions), one does not consciously maintain different sets of belief-states for the possible observations one could perceive after some action. Rather, one can imagine, that given (mental) models of stochastic actions and observations, a single projected state is formed (for every action considered). That is, only the most expected state is considered, given a sequence of actions.

Our idea for unifying all belief-states which would have been generated for each observation in $Z = \{z_1, z_2, \ldots, z_n\}$ after action a performed in belief-state b is to select, for each feature, the feature value closest to the expected value of the feature (note that the *expected* value may not exist). Let $s(f)$ be the value of feature f in state s. Formally, given a set $B(a,b) = \{b^a_{z_1}, b^a_{z_2}, \ldots, b^a_{z_n}\}$ of projected belief-states, the expected value $\hat{v}(a,b,f)$ of feature f is[1]

$$\begin{aligned}
\hat{v}(a,b,f) &= \sum_{b^a_z \in B(a,b), \langle s', p \rangle \in b^a_z} s'(f) Pr(z \mid a,b) p \\
&= \sum_{z \in Z, s' \in S} s'(f) Pr(z \mid a,b) \times \\
&\qquad \frac{O(s',a,z) \sum_{s \in S} T(s,a,s') b(s)}{Pr(z \mid a,b)} \quad \text{(from (1))} \\
&= \sum_{z \in Z, s' \in S} s'(f) O(s',a,z) \sum_{s \in S} T(s,a,s') b(s),
\end{aligned}$$

[1] Recall that $Pr(z \mid ,a,b)$ can be viewed as the probability of reaching belief-state b^a_z from b.

Algorithm 2: OUCEF

Input: b: belief-state, h: horizon
Output: an action, the action's Q value

```
1  if h = 0 then
2      return stop, 0
3  if h > 0 then
4      maxVal ← −∞
5      foreach a ∈ A do
6          _, V = OUCEF({⟨cef(a,b), 1⟩}, h − 1)
7          value ← ρ(a,b) + γ·V
8          if value > maxVal then
9              maxVal ← value
10             bestAct ← a
11     return bestAct, maxVal
```

As mentioned in the section about the Monte Carlo Sampling approach, computing $Pr(z \mid a,b)$ is in $O(|S|^2)$, a relatively intensive computation. Notice that it has been cancelled out of the calculation of $\hat{v}(a,b,f)$.

Every state $s \in S$ is defined by the value the state assigns to the set of features F considered for the domain. Let $\{v^f_1, v^f_2, \ldots, v^f_m\}$ be the finite set of discrete values that feature f can take. Hence, a state s can be expressed as $\{\langle f, v^f_s \rangle \mid f \in F, v^f_s \in \{v^f_1, v^f_2, \ldots, v^f_m\}\}$. I propose that the most expected state $s^* = cef(a,b)$, given some action a is executed in some b, is identified by determining the feature values closest to the expected feature values:

$$cef(a,b) \stackrel{def}{=} \{\langle f, snapTo(\hat{v}(a,b,f)) \rangle \mid f \in F\},$$

where $snapTo(\hat{v}(a,b,f))$ is the value in $\{v^f_1, v^f_2, \ldots, v^f_m\}$ closest to $\hat{v}(a,b,f)$.

If a feature f_{qal} is qualitative, then its values can be converted to numbers for the period of calculation of $\hat{v}(a,b,f_{qal})$. For instance, given a POMDP with only a 'direction' feature *dir* with possible values in $\{North, East, West, South\}$, one can associate 1 with *North*, 2 with *East*, 3 with *West* and 4 with *South*. If there are two observations, then there are two possible new belief-states, each with four states, for any action executed in any belief-state. Assume, for example, that the two new belief-states are $b_1 = \{\langle\{(dir,1)\}, 0.1\rangle, \langle\{(dir,2)\}, 0.2\rangle, \langle\{(dir,3)\}, 0.3\rangle, \langle\{(dir,4)\}, 0.4\rangle\}$ and $b_2 = \{\langle\{(dir,1)\}, 0.3\rangle, \langle\{(dir,2)\}, 0.3\rangle, \langle\{(dir,3)\}, 0.3\rangle, \langle\{(dir,4)\}, 0.1\rangle\}$, due to performing some a' in some b' and perceiving z'_1 with some probability p', respectively, perceiving z'_2 with some probability $1 - p'$. Also assume that the probability of reaching b_1 is 0.4 and of reach-

Table 1: Results for $h = 3$ and $sf = 0.8$.

Algorithm	ic/s	s/a	ic
MT	.62	0.29	6.52
MC (c=1)	.12	1.21	5.18
MC (c=2)	.10	1.68	5.84
MC (c=3)	.07	2.39	6.16
RTBSS	.12	1.27	5.50
OUCEF	1.48	0.09	4.80

Table 2: Results for $h = 4$ and $sf = 0.8$.

Algorithm	ic/s	s/a	ic
MT	.17	1.33	7.98
MC (c=1)	.02	6.67	5.06
MC (c=2)	.01	12.57	5.52
MC (c=3)	.01	15.22	5.56
RTBSS	0.03	4.60	4.52
OUCEF	.40	0.38	5.42

Table 3: Results for $h = 3$ and $sf = 0.95$.

Algorithm	ic/s	s/a	ic
MT	.97	0.26	9.04
MC (c=1)	.15	1.17	6.44
MC (c=2)	.11	1.67	6.90
MC (c=3)	.08	2.36	6.78
RTBSS	.17	0.95	5.96
OUCEF	1.69	0.09	5.48

Table 4: Results for $h = 4$ and $sf = 0.95$.

Algorithm	ic/s	s/a	ic
MT	.20	1.26	9.04
MC (c=1)	.03	6.87	6.98
MC (c=2)	.02	11.11	6.96
MC (c=3)	.01	15.57	6.96
RTBSS	.04	3.94	5.18
OUCEF	.60	0.37	7.94

ing b_2 is 0.6.[2] The expected value of *dir* is thus $\hat{v}(a',b',dir) = 0.4(1 \times 0.1 + 2 \times 0.2 + 3 \times 0.3 + 4 \times 0.4) + 0.6(1 \times 0.3 + 2 \times 0.3 + 3 \times 0.3 + 4 \times 0.1) = 2.52$. 2.52 is the closest to $dir = 3$ and we estimate that the agent should believe it is facing West after a' performed in b'. Hence, $cef(a',b') = \{(dir, snapTo(\hat{v}(a',b',dir))\} = \{(dir, West)\}$.

Algorithm 2 summarizes the approach.

4 EXPERIMENTS

To assess the proposed algorithm, four sets of experiments were performed in a six-by-six grid-world. In this world, the agent's task is to collect twelve randomly scattered items. States are quadruples (x,y,d,t), with $x,y \in \{1,\cdots,6\}$ being the coordinates of the agent's position in the world, $d \in \{North, East, West, South\}$ the direction it is facing, and $t \in \{0,1\}$, $t = 1$ if an item is present in the cell with the agent, else $t = 0$. The agent can perform five actions $\{left, right, forward, see, collect\}$, meaning, turn left, turn right, move one cell forward, see whether an item is present and collect an item. The only observation possible when executing one of the physical actions is *obsNil*, the null observation, and *see* has possible observations from the set $\{0,1\}$ for whether the agent sees the presence of an item (1) or not (0).

The same reward function is used in all cases. The value of the reward function is essentially proportional to the inverse of the Manhattan distance from the cell in which the agent is currently located to the nearest item to be collected. The agent also receives rewards for executing a *collect* action when there is an item in the same cell.

Next, I define the possible outcomes for each action: When the agent turns left or right, it can get stuck in the same direction, turn 90° or overshoots by 90°. When the agent moves forward, it moves one cell in the direction it is facing or it gets stuck and does not move. The agent can see an item or see nothing (no item in the cell), and collecting is deterministic (if there is an item present, it will be collected with certainty, if the agent executes *collect*). All actions except *collect* are designed so that (i) the correct outcome is achieved 95% of the time and incorrect outcomes are achieved 5% of the time or (ii) the correct outcome is achieved 80% of the time and incorrect outcomes are achieved 20% of the time. So that the agent does not get lost too quickly, I have included an automatic localization action, that is, a sensing action returns information about the agent's approximate location. The action is automatic because the agent cannot choose to perform it or not to perform it; the agent localizes itself after every regular/chosen action is executed. However, just as with regular actions, the localization sensor is noisy, and it correctly reports the agent's location with probability 0.95 or 0.80, else the sensor reports a location adjacent to the agent with probability uniformly distributed over 0.05, resp., 0.2

For each experiment, 50 trials were run with the agent starting in random locations and performing 36 actions per trial. The look-ahead depth is set to either $h = 3$ or $h = 4$, and for MC, the sample size is set to $c = 1$, $c = 2$ or $c = 3$. Only results for $c = 1$ are reported here, because MC was most effective in this case.

Let *sf* denote the stochasticity factor; it corresponds to the percentage correct outcomes of actions and observations. I thus set $sf = 0.8$ or $sf = 0.95$. For each experiment, I measure the average number

[2] In the following calculation, $Pr(z \mid a,b)$ is used to simplify the example. Else, defining and using the observation and transition functions would be necessary.

of items collected (ic), the average time (in seconds) it takes to plan for one action (s/a) and the average number of items collected per second (ic/s). ic/s is taken as the performance measure in these experiments. Arguably, ic could have been used as the performance measure, however, I feel that speed of planning should be considered when measuring performance of online algorithms. Tables 1 thru 4 report the results.

5 CONCLUDING REMARKS

I presented a novel online POMDP algorithm which performs at least twice as good as the other algorithms on a particular grid-world problem. The basic algorithm with mean-as-threshold belief-state compression always collected the most items (ic). However, because it takes more than twice (for $h = 3$) or three times (for $h = 4$) as long as OUCEF to select the next action, its effectiveness is significantly below OUCEF's (w.r.t. ic/s). OUCEF's nominal performance (ic) is comparable with that of the other algorithms over the four experiment parameter combinations.

The effectiveness of the OUCEF algorithm is due to (i) unifying the branches due to nondeterministic observations by collecting all belief-nodes at the ends of these branches into one set B, and then (ii) selecting the state most representative of B, by calculating the expected values of the features of the states in B.

The aspect of this work most in need of attention is to validate the approach on different benchmark problems. It might be the case that the OUCEF algorithm is well suited to the kind of grid-world problems presented here, but to few other problems. Or it might be suited to many kinds of problems. This paper is, however, a first step in introducing and testing the algorithm. At the very least, the new ideas presented here might lead other researchers to new insights in their online POMDP algorithm. A theoretical analysis of the optimality of OUCEF is also required and could lead to interesting insights.

REFERENCES

He, R., Brunskill, E., and Roy, N. (2011). Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research (JAIR)*, 40:523–570.

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

Koenig, S. (2001). Agent-centered search. *Artificial Intelligence Magazine*, 22:109–131.

Kurniawati, H., Du, Y., Hsu, D., and Lee, W. (2011). Motion planning under uncertainty for robotic tasks with long time horizons. *International Journal of Robotics Research*, 30(3):308–323.

Li, X., Cheung, W., Liu, J., and Wu, Z. (2007). A novel orthogonal NMF-based belief compression for POMDPs. In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML-07)*, pages 537–544, New York, NY, USA. ACM Press.

Lovejoy, W. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66.

McAllester, D. and Singh, S. (1999). Approximate planning for factored POMDPs using belief state simplification. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 409–416, San Francisco, CA. Morgan Kaufmann.

Monahan, G. (1982). A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16.

Paquet, S., Tobin, L., and Chaib-draa, B. (2005). Real-time decision making for large POMDPs. In *Advances in Artificial Intelligence: Proceedings of the Eighteenth Conference of the Canadian Society for Computational Studies of Intelligence*, volume 3501 of *Lecture Notes in Computer Science*, pages 450–455. Springer Verlag.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032.

Poupart, P. and Boutilier, C. (2003). Value-directed compression of POMDPs. In *Advances in Neural Information Processing Systems (NIPS 2003)*, pages 1547–1554. MIT Press, Massachusetts/England.

Rens, G. and Ferrein, A. (2013). Belief-node condensation for online pomdp algorithms. In *Proceedings of IEEE AFRICON 2013*, pages 1270–1274, Red Hook, NY 12571 USA. Institute of Electrical and Electronics Engineers, Inc.

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

Roy, N., Gordon, G., and Thrun, S. (2005). Finding approximate POMDP solutions through belief compressions. *Journal of Artificial Intelligence Research (JAIR)*, 23:1–40.