Covering Number for Efficient Heuristic-Based POMDP Planning

Zongzhang Zhang ZHANGZZ@COMP.NUS.EDU.SG
David Hsu DYHSU@COMP.NUS.EDU.SG
Wee Sun Lee LEEWS@COMP.NUS.EDU.SG
Department of Computer Science, National University of Singapore, Singapore 117417, Singapore

Abstract

The difficulty of POMDP planning depends on the size of the search space involved. Heuristics are often used to reduce the search space size and improve computational efficiency; however, there are few theoretical bounds on their effectiveness. In this paper, we use the *covering number* to characterize the size of the search space reachable under heuristics and connect the complexity of POMDP planning to the effectiveness of heuristics. With insights from the theoretical analysis, we have developed a practical POMDP algorithm, *Packing-Guided Value Iteration* (PGVI). Empirically, PGVI is competitive with the state-of-the-art point-based POMDP algorithms on 65 small benchmark problems and outperforms them on 4 larger problems.

1. Introduction

Partially observable Markov decision processes (POMDPs) provide a rich mathematical model for planning under uncertainty (Kaelbling et al., 1998). However, POMDPs are computationally intractable to solve exactly (Madani et al., 1999). In the past decade, enormous progress has been made in computing approximate POMDP solutions (Pineau et al., 2003; Smith & Simmons, 2005; Kurniawati et al., 2008; Ross et al., 2008; Bonet & Geffner, 2009; Silver & Veness, 2010; Zhang & Chen, 2012; Grześ et al., 2013; Shani et al., 2013).

On the theoretical front, the *covering number* of the reachable space has been proposed to quantify the complexity of POMDP planning (Hsu et al., 2007), particularly, for *point-based methods* (Smith & Simmons, 2005; Pineau et al., 2006; Shani et al., 2007; Kurniawati et al., 2008; Shani et al., 2013). Intuitively, the covering number is the minimum number of fixed-size balls required to cover the search space so that all points in the search space lie within some ball. Both theoretical and empirical results support the covering number as a promising complexity measure for POMDP planning and learning (Hsu et al., 2007; Zhang et al., 2012).

In practice, many well-known POMDP planning algorithms use the *lower and upper bounds* of the optimal value function (Hauskrecht, 2000; Smith & Simmons, 2005; Kurniawati et al., 2008), or just the *upper bound* as *heuristics* (Hauskrecht, 2000; Bonet & Geffner, 2009) in guiding the search towards the optimally reachable space. Although these algorithms have made impressive progress in computing approximate solutions by using heuristics and have been successfully applied in several practical domains (Hsiao et al., 2007; Pineau et al., 2006; Hsu et al., 2008; Kurniawati et al., 2011; Grześ et al., 2013), few existing works have analyzed the relationship between the quality of the heuristics and the complexity of POMDP planning.

In this paper, we fill this gap by connecting the size of search spaces reachable under heuristic, as measured by the covering number, to the complexity of POMDP planning. We consider two cases, when only an upper bound heuristic is available and when both upper and lower bound heuristics are available. We show that an ϵ-optimal solution can be computed in time *polynomial* in the covering number for the both cases. This suggests one avenue of handling practical problems: use domain knowledge to find good upper and lower bounds that effectively reduce the covering number of the reachable space under the heuristics.

One key idea behind our theoretical analysis is to build a separate *packing* of sampled beliefs at each level of the search tree to control the number of beliefs at level d, so that it does not grow exponentially in d. Packing is closely related to covering and is used to create a covering in the proofs. We exploit this proof idea in building a practical point-based algorithm, *Packing-Guided Value Iteration* (PGVI). In addition to providing theoretical guarantees, packing helps PGVI to identify interesting parts of the space that is sparsely packed and to sample new beliefs

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

and perform point-based backup there. Compared with state-of-the-art point-based POMDP algorithms, PGVI is very efficient on 65 small benchmark problems and 4 larger robotic problems.

2. Preliminaries

A POMDP models an agent taking a sequence of actions in a partially observable stochastic environment to maximize its total reward (Kaelbling et al., 1998). A discrete and discounted POMDP model can be formally defined by a tuple $(S, A, Z, T, \Omega, R, \gamma)$. In the tuple, S, A and Z are the finite and discrete state space, action space, and observation space, respectively. At each time step, the agent takes some action $a \in A$ and moves from a start state s to an end state s'. The end state s' is given by a state-transition function $T(s, a, s') : S \times A \times S \rightarrow [0, 1]$, where $T(s, a, s') = Pr(s'|s, a)$. The agent then makes an observation to gather information on its current state. The outcome of observing $z \in Z$ is given by an observation function $\Omega(a, s', z) : A \times S \times Z \rightarrow [0, 1]$, where $\Omega(a, s', z) = Pr(z|a, s')$. The reward function $R(s, a) : S \times A \rightarrow \mathbb{R}$ gives the agent a real-value reward after it takes action a in state s. We let $R_{\max} = \max_{s \in S, a \in A} |R(s, a)|$. The last element $\gamma \in (0, 1)$ is the discount factor. Thus, the *expected total reward* is given by $\mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, where s_t and a_t are the agent's state and action at time t.

A *belief state* (or *belief*) b is a discrete probability distribution over the state space, whose element $b(s)$ gives the probability that the system's state is s. A belief state space $\mathcal{B}$ is comprised of all possible beliefs. A search space B is a subset of $\mathcal{B}$ and can be represented as an AND/OR belief tree T_B rooted at the *initial belief* b_0. In the tree, nodes and edges correspond to beliefs and action-observation pairs, respectively. Suppose that a child node b', denoted by $\tau(b, a, z)$, is connected to its parent b by an edge (a, z). We can compute b' using the Bayesian formula $b'(s') = \frac{1}{Pr(z|b,a)} \Omega(a, s', z) \sum_{s \in S} T(s, a, s') b(s)$, where $Pr(z|b, a) = \sum_{s' \in S} \Omega(a, s', z) \sum_{s \in S} T(s, a, s') b(s)$ (Kaelbling et al., 1998).

A POMDP solution is a *policy* π that specifies the action $\pi(b)$ for every belief b. Our goal is to find an *optimal policy* π^* that maximizes the expected total reward. A policy π induces a *value function* $V^{\pi}(b)$ that specifies the expected total reward of executing π starting from any belief b. The *optimal value function* $V^*(b)$ is the value function associated with the optimal policy π^*. We define $Q^*(b, a)$ as $\sum_{s \in S} R(s, a) b(s) + \sum_{z \in Z} Pr(z|b, a) V^*(\tau(b, a, z))$, and therefore, have $V^*(b) = \max_{a \in A} Q^*(b, a)$. We denote V^L and V^U as the lower and upper bounds of V^*, respectively. Both V^L and V^U are assumed to be *uniformly improvable* (Smith, 2007) in this paper, meaning that applying a point-based update brings them everywhere closer to V^*. We also assume that we are provided with heuristics, f and g, that can provide initial values V_f^L to the lower bound and V_g^U to the upper bound. Similarly, we use Q^L, Q^U, Q_f^L and Q_g^U for the corresponding bounds of Q^*. In practice, the bounds for Q can be constructed from the bounds for V by one step lookahead (Hauskrecht, 2000).

We describe the mathematical definition of the covering number of a set of points as follows:

Definition 1. *(Hsu et al., 2007) Given a metric space X, a δ-cover of a set $B \subseteq X$ is a set of points $C \subseteq X$ such that for every point $b \in B$, there is a point $c \in C$ with $||b - c|| \leq \delta$. The δ-covering number of B, denoted by $\mathcal{C}_B(\delta)$, is the size of the smallest δ-cover of B.*

Intuitively, the covering number is equal to the minimum number of balls of radius δ needed to cover the set B. In this paper, we measure the distance between belief points in an L^1 metric space $\mathcal{B}$: for $b_1, b_2 \in \mathcal{B}$, $||b_1 - b_2|| = \sum_{s \in S} |b_1 - b_2|$. We refer a belief b's δ-region as a subspace in X that satisfies $||b' - b|| \leq \delta$ for all b' in the region.

We use the covering number to measure the size of belief spaces. The set of beliefs that are reachable from the initial belief b_0 under arbitrary sequences of actions and observations is denoted by $\mathcal{R}(b_0)$. The *optimally reachable belief space* $\mathcal{R}^*(b_0)$ refers to the set of beliefs that are reachable from b_0 under some optimal policy. We denote $\mathcal{C}(\delta)$ as the δ-covering number of $\mathcal{R}(b_0)$, $\mathcal{C}^*(\delta)$ as the δ-covering number of $\mathcal{R}^*(b_0)$, and $\mathcal{T}_{\mathcal{R}}$ as the tree rooted at b_0 consisting of beliefs in the *reachable belief space* $\mathcal{R}(b_0)$.

3. Related Work

Kakade and his colleagues applied the notion of the covering number into reinforcement learning (Kakade et al., 2003). Later, Hsu et al. (2007) extended use of covering number from MDPs to POMDP planning. Recent research work provided empirical evidence that estimated covering number of $\mathcal{R}(b_0)$ was better than the state space size in predicting the difficulty of POMDP planning and learning on small benchmark problems (Zhang et al., 2012).

In (Hsu et al., 2007), the key point of connecting the covering number to POMDP planning complexity is the following: for any two beliefs, if their distance is small, then their optimal values are also similar. Thus, when the value of a belief b is accurate enough, it can be used to estimate the value of beliefs that are close to b with only small error. By stopping the search when it is near to a region that has been searched before, the covering number can be exploited to bound the width of the search tree. Together with the idea of bounding the depth of the search tree by the discount factor, it was shown that an approximately optimal POMDP solution can be computed in time at most *quadratic poly-*

nomial in both $\mathcal{C}(\delta)$ and $\mathcal{C}^*(\delta)$.

In the past decade, point-based value-iteration algorithms have made impressive progress in computing approximate solutions to large POMDPs (Shani et al., 2013). Their success is mainly due to the efficient sampling strategies. Point-Based Value Iteration (PBVI) (Pineau et al., 2003) prefers to sample beliefs that are far away from those sampled beliefs. Heuristic Search Value Iteration (HSVI2) (Smith & Simmons, 2005), SARSOP (Kurniawati et al., 2008) and GapMin (Poupart et al., 2011) sample beliefs by using both the lower and upper bound heuristics. The lower bound is usually obtained by using the blind policy and the upper bound is often initialized by the QMDP or Fast Informed Bound (FIB) method (Hauskrecht, 2000). These algorithms use the action-selection strategy that chooses the action with the highest upper bound. Compared with HSVI2 and SARSOP, GapMin strives to compute tight bounds by performing a prioritized breadth-first search, propagating upper bound improvements, and computing exact interpolations by Linear Programming (LP) (Poupart et al., 2011). However, the idea of using the insight from the covering number to control the width of the search tree has not been exploited in the three state-of-the-art algorithms. The performance guarantee for HSVI2, provided in (Smith, 2007), is $O(h \cdot |A|^h |Z|^h)$, where h is the height of the search tree. It is essentially the time required to search the whole depth-bounded search tree.

4. Complexity of Heuristic-Based POMDP Planning

We examine the complexity of POMDP planning when various heuristics are used. We show that these heuristics can be used to define various search spaces and that approximately optimal POMDP solutions can be found in time polynomial in the covering number of these search spaces.

We start with insights from practical POMDP algorithms such as HSVI2 and SARSOP. These algorithms are *action optimistic* – they select actions with the highest Q^U during the search process. Since $V^*(b) \leq \max_{a\in A} Q^U(b,a)$ and Q^U is uniformly improvable, we know that action optimistic algorithms have zero probability of visiting beliefs under the action branch a that satisfies $Q^U_g(b,a) < V^*(b)$. This allows us to define the search space reachable under action optimistic algorithms with heuristic g, $\mathcal{R}^U_g(b_0)$, as the set of beliefs b that can be reached from b_0 by taking action branches a that satisfy $Q^U_g(b,a) \geq V^*(b)$.

The size of $\mathcal{R}^U_g(b_0)$ gives a reasonable indication of the complexity of solving the POMDP exactly. However, we would like to approximate in two ways: by fixing the depth of the search tree and by interpolating the values of nearby beliefs to take advantage of the Lipschitz property. We define $\mathcal{C}^U_g(\delta)$ as the δ-covering number of $\mathcal{R}^U_g(b_0)$. Interestingly, approximately optimal POMDP solutions can be found in time polynomial in $\mathcal{C}^U_g(\delta)$. As $\mathcal{C}^U_g(\delta)$ depends on Q^U_g, the covering number of the space reachable under action optimistic algorithms is a reasonable measure of the quality of the heuristic. Note also that $\mathcal{R}^U_g(b_0)$ becomes $\mathcal{R}^*(b_0)$ if we use Q^* as our heuristic, hence the covering number converges to the covering number of $\mathcal{R}^*(b_0)$ as the quality of the heuristic improves.

Theorem 1. *Given any constant $\epsilon > 0$ and any initial belief $b_0 \in \mathcal{B}$. Let $\mathcal{C}^U_g(\delta)$ be the δ-covering number of $\mathcal{R}^U_g(b_0)$. Then, an approximation $V(b_0)$ of $V^*(b_0)$, with error $|V^*(b_0) - V(b_0)| \leq \epsilon$, can be found in time $O(h \cdot \mathcal{C}^U_g(\delta/2)^2)$, where $h = \log_\gamma \frac{(1-\gamma)\epsilon}{2R_{\max}}$, $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$ and $Q^U_g(b,a)$ is used for the initial upper bound.*

Before proving Theorem 1, we state two lemmas from (Hsu et al., 2007). The first lemma states that the optimal value function V^* satisfies the following Lipschitz condition:

Lemma 1. *(Hsu et al., 2007) For any two belief points b and b', if $||b - b'|| \leq \delta$, then $|V^*(b) - V^*(b')| \leq \frac{R_{\max}}{1-\gamma}\delta$.*

The second one is related to the packing number, a notion closely related to the covering number:

Definition 2. *Given a metric space X, a δ-packing of a set $B \subseteq X$ is a set of points P in B such that for any two points $p_1, p_2 \in P$, $||p_1 - p_2|| > \delta$. The δ-packing number of a set B, denoted $\mathcal{P}_B(\delta)$, is the size of the largest δ-packing of B.*

For any set B, the following relationship holds between its packing number and covering number.

Lemma 2. *(Hsu et al., 2007) $\mathcal{C}_B(\delta) \leq \mathcal{P}_B(\delta) \leq \mathcal{C}_B(\delta/2)$.*

Proof. (of Theorem 1) To prove the result, we give an algorithm that computes the required approximation. It performs a depth-first search on a depth-bounded belief tree and uses approximate memorization to avoid unnecessarily computing the values of very similar beliefs. Intuitively, to achieve a polynomial time algorithm, we bound the height of the tree by using the discount factor and bound the width of the tree by exploiting the covering number.

We perform the depth-first search recursively on $\mathcal{T}_\mathcal{R}$ that has root b_0 and height h, while maintaining a δ-packing at *every* level of $\mathcal{T}_\mathcal{R}$. By convention, the root node is at level 0 and the leaf nodes are at level h. Each leaf node b_l is initialized with estimated value $V(b_l) = 0$. At an internal node b, we first check if b is within a distance δ of a point b' in the current packing at level i. If it is we set the estimate $V(b) = V(b')$, abort the recursion at b, and backtrack. Otherwise, we add b to the packing, sort the actions using $Q^U_g(b,a)$ and explore the actions according to the sorted order, with the larger values searched earlier. The estimate $V(b)$ is initialized to the value returned by searching the first action and updated each time searching a new action, which returns an improved value, until $Q_g^U(b,a) < V(b) + \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$.

We now prove that $|V^*(b) - V(b)| \le \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$ after the update operations are completed at any b at level i of $\mathcal{T}_{\mathcal{R}}$. Following (Hsu et al., 2007), we prove that $|V(b) - V^*(b)| \le \frac{\gamma R_{\max}(1-\gamma^{h-i})}{(1-\gamma)^2}\delta + \gamma^{h-i}\frac{R_{\max}}{1-\gamma}$. This gives $|V^*(b) - V(b)| \le \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$ by setting $h = \log_\gamma \frac{(1-\gamma)\epsilon}{2R_{\max}}$ and $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$.

Let $\epsilon_i = |V(b) - V^*(b)|$ be the approximation error for a node b at level i of $\mathcal{T}_{\mathcal{R}}$, if search is not aborted at b. Let ϵ_i' be the error for b, if the search aborts at b and sets $V(b) = V(b')$ for some b' in the packing at level i. We have

$$\begin{aligned}
\epsilon_i' &= |V^*(b) - V(b')| \\
&\le |V^*(b) - V^*(b')| + |V^*(b') - V(b')| \\
&\le \frac{R_{\max}}{1-\gamma}\delta + \epsilon_i,
\end{aligned}$$

where the last inequality uses Lemma 1 and the definition of ϵ_i. At the leaves, we set the estimated value to 0, hence $\epsilon_h \le R_{\max}/(1-\gamma)$. The children of b, which are at level $i-1$, have error at most ϵ_{i-1}'. We do a proof by induction. Assume that, at level $i+1$, $|V(b) - V^*(b)| \le \frac{\gamma R_{\max}(1-\gamma^{h-i-1})}{(1-\gamma)^2}\delta + \gamma^{h-i-1}\frac{R_{\max}}{1-\gamma}$. For every action a that we search at level i, we have

$$\begin{aligned}
&|Q(b,a) - Q^*(b,a)| \\
\le\ & \gamma\left(\frac{\gamma R_{\max}(1-\gamma^{h-i-1})}{(1-\gamma)^2}\delta + (\delta + \gamma^{h-i-1})\frac{R_{\max}}{1-\gamma}\right) \\
=\ & \frac{\gamma R_{\max}(1-\gamma^{h-i})}{(1-\gamma)^2}\delta + \gamma^{h-i}\frac{R_{\max}}{1-\gamma} \\
=\ & \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}
\end{aligned}$$

when we set $h = \log_\gamma \frac{(1-\gamma)\epsilon}{2R_{\max}}$ and $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$.

Each action is backed up in sorted order until $Q_g^U(b,a) < V(b) + \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$, where the right hand side is the current upper bound on $V^*(b)$. Because the remaining actions a' have $Q_g^U(b,a') \le Q_g^U(b,a)$, we know for sure that $V^*(b) \le V(b) + \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$. At the same time, any action a that has been searched establishes that $V^*(b) \ge Q(b,a) - \frac{\epsilon}{2\gamma^i} - \frac{\epsilon(1-\gamma^{h-i})}{2}$. As $V(b)$ is the value of the largest $Q(b,a)$ found so far, we have $V^*(b) \ge V(b) - \frac{\epsilon}{2\gamma^i} - \frac{\epsilon(1-\gamma^{h-i})}{2}$. Taken together, this implies $|V^*(b) - V(b)| \le \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$.

Finally, we calculate the running time of the algorithm. We first note that $V^*(b) \le V(b) + \frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$, hence we will never search an action for which $Q_g^U(b,a) < V^*(b)$. This implies that the number of elements in the packing at each level is bounded by the packing number of $\mathcal{R}_g^U(b_0)$. For each node b in the packing of $\mathcal{R}_g^U(b_0)$ is expanded and it takes $O(|A|\log|A|)$ time to determine the search order of action branches. Then, it calculates the beliefs and the corresponding values for all its (expanded) children and performs a point-based update at b to compute $V(b)$. It takes $O(|S|^2)$ time to calculate the belief at a child node. After that, we perform a nearest neighbour search in $O(\mathcal{P}_g^U(\delta)|S|)$ time to check whether the child node lies within a distance δ of any point in the packing of $\mathcal{R}_g^U(b_0)$ at that level. Since b has at most $|A||Z|$ expanded children, the expansion operation takes $O(|A|\log|A| + |A||Z|(|S|^2 + |S|\mathcal{P}_g^U(\delta)))$ time. The point-based update then computes $V(b)$ as an average of its children's values, weighted by the probabilities specified by the observation function, and takes only $O(|A||Z|)$ time. Since there are h packing of size $\mathcal{P}_g^U(\delta)$ each and by Lemma 2, $\mathcal{P}_g^U(\delta) \le \mathcal{C}_g^U(\delta/2)$, the total running time of our algorithm is $O\Big(h \cdot \mathcal{C}_g^U(\delta/2)(|A|\log|A| + |A||Z|(|S|^2 + (|S| + 1)\mathcal{C}_g^U(\delta/2)))\Big)$. Assume that $|S|$, $|A|$, and $|Z|$ are constant to focus on the dependency on the covering number. So, we get the final result. □

In Theorem 1, only an initial upper bound is utilized. Algorithms such as HSVI2 and SARSOP utilize an initial lower bound as well (Smith & Simmons, 2005; Kurniawati et al., 2008). The use of a good initial lower bound may cut down the size of the search space substantially. We define a search space limited by lower bound $V_f^L(b)$ and upper bound $V_g^U(b)$ as follows: $\mathcal{R}_{f,L}^{g,U}(b_0)$ is the space reachable from b_0 under all action-observation sequences satisfying $Q_g^U(b,a) \ge V^*(b)$ and $V_g^U(b) - V_f^L(b) > \frac{\epsilon}{\gamma^{d_b}}$, where d_b is the depth (or level) of b in the belief tree $\mathcal{T}_{\mathcal{R}}$.

Theorem 2. *Given any constant $\epsilon > 0$ and any initial belief $b_0 \in \mathcal{B}$. Let $\mathcal{C}_{f,L}^{g,U}(\delta)$ be the δ-covering number of $\mathcal{R}_{f,L}^{g,U}(b_0)$. Then, an approximation $V(b_0)$ of $V^*(b_0)$, with error $|V^*(b_0) - V(b_0)| \le 2\epsilon$, can be found in time $O(h \cdot \mathcal{C}_{f,L}^{g,U}(\delta/2)^2)$, where $h = \log_\gamma \frac{\epsilon(1-\gamma)}{2R_{\max}}$, $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$, $V_f^L(b)$ is used as an initial lower bound and $V_g^U(b)$ is used as an initial upper bound.*

Proof. We prove this theorem by using a modified algorithm in the proof of Theorem 1. We perform the depth-first search recursively on a belief tree $\mathcal{T}_{\mathcal{R}}$ that has root b_0 and height h, while maintaining a δ-packing of $\mathcal{R}_{f,L}^{g,U}(b_0)$ at *every* level. If b is not in $\mathcal{R}_{f,L}^{g,U}(b_0)$, namely $V_0^U(b) - V_0^L(b) \le \frac{\epsilon}{\gamma^{d_b}}$, we set $V(b) = \frac{V_f^L(b) + V_g^U(b)}{2}$, abort the recursion at b, and backtrack. Else, if b is within a distance δ of a b' in the current packing at level i, we set $V(b) = V(b')$, abort the recursion at b, and backtrack. Otherwise, we add b to the packing, sort the actions using $Q^U_g(b,a)$ and explore the actions according to the sorted order, with the larger values searched earlier. The estimate $V(b)$ is initialized to the value returned by searching the first action and updated each time searching a new action, which returns an improved value, until $Q^U_g(b,a) < V(b) + \frac{3\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i-1})}{2}$.

We now calculate the values for h and δ required to achieve the given approximation bound 2ϵ at b_0. Let $\epsilon_i = |V^*(b) - V(b)|$ denote the approximation error for a node b at level i of $\mathcal{T_R}$, if the recursive search continues in the children of b. Let ϵ'_i denote the error for b, if the search aborts at b and sets $V(b) = V(b')$ for some b' in the packing at level i. Hence, $\epsilon'_i \le \frac{R_{\max}}{1-\gamma}\delta + \epsilon_i$. As in Theorem 1, we explore an action only if its initial upper bound is at least as large as the best upper bound among the searched actions. By the same argument as in Theorem 1's proof, after completing the update operations at b at level i, we have

$$\epsilon_i \le \gamma \max\left\{\frac{\epsilon}{\gamma^{i+1}}, \epsilon'_{i+1}\right\} \le \gamma\Big[\frac{\epsilon}{\gamma^{i+1}} + \epsilon'_{i+1}\Big]$$

and thus we can write the recurrence

$$\epsilon_i \le \gamma\Big[\frac{\epsilon}{\gamma^{i+1}} + \Big(\frac{R_{\max}}{1-\gamma}\delta + \epsilon_{i+1}\Big)\Big].$$

Clearly, we can set $V^U_g(b) \le \frac{R_{\max}}{1-\gamma}$ and $V^L_f(b) \ge -\frac{R_{\max}}{1-\gamma}$ for all $b \in \mathcal{B}$. So $\epsilon_h \le \max_{b\in\mathcal{B}} \frac{V^U_g(b) - V^L_f(b)}{2} \le \frac{R_{\max}}{1-\gamma}$.

Expanding the recurrence, we get

$$\begin{aligned} \epsilon_k &\le \frac{\epsilon}{\gamma^k} + \frac{\gamma R_{\max}(1-\gamma^{(h-1)-k})}{(1-\gamma)^2}\delta + \frac{\gamma^{h-k}R_{\max}}{1-\gamma} \\ &= \frac{3\epsilon}{2\gamma^k} + \frac{\epsilon(1-\gamma^{(h-1)-k})}{2}, \end{aligned}$$

which holds for all $0 \le k \le h-2$, by setting $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$ and $h = \log_\gamma \frac{\epsilon(1-\gamma)}{2R_{\max}}$. The final equality explains why we use $\frac{3\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i-1})}{2}$ to instead of $\frac{\epsilon}{2\gamma^i} + \frac{\epsilon(1-\gamma^{h-i})}{2}$ that was used in the proof of Theorem 1. So, we find that the error ϵ_0 at the root b_0 is given by $|V^*(b_0) - V(b_0)| \le \frac{3\epsilon}{2} + \frac{\epsilon(1-\gamma^{h-1})}{2} \le 2\epsilon$.

Using the same analysis method in the proof of Theorem 1, the running time of the algorithm here is $O\Big(h \cdot \mathcal{C}^{g,U}_{f,L}(\delta/2)(|A|\log|A| + |A||Z|(|S|^2 + (|S|+1)\mathcal{C}^{g,U}_{f,L}(\delta/2)))\Big)$. Thus, we get the final result. □

The theorem suggests one avenue of fighting the *curses of dimensionality and history* (Pineau et al., 2003; Silver & Veness, 2010): use domain knowledge to find good lower and upper bounds, V^L_f and V^U_g, so that $\mathcal{C}^{g,U}_{f,L}(\delta/2)$ is small.

Algorithm 1 $\pi = \text{PGVI}(\epsilon, \delta)$.

```
Initialize the bounds $V^L$ and $V^U$;
packing = $\emptyset$, finished = $\emptyset$;
while $V^U(b_0) - V^L(b_0) > 2\epsilon$ do
    EXPLORE($b = b_0, d_b = 0, \epsilon, \delta$);
end while
return the action corresponding to $V^L$;
```

Algorithm 2 EXPLORE$(b, d_b, \epsilon, \delta)$.

```
if excess$(b, d_b, \epsilon) \le 0$ then
    insert $b$ into finished$(d_b)$;
    return ;
end if
$a^* = \arg\max_{a\in A} Q^U(b,a)$;
$z^* = \arg\max_{z\in Z^{\text{UF}}}[Pr(z|b,a^*) \cdot \text{excess}(\tau(b,a^*,z), d_b+1, \epsilon) \cdot \text{dis}(\tau(b,a^*,z), \text{packing}(d_b+1), \delta)]$;
if $z^*$ == NULL or excess$(\tau(b,a^*,z^*), d_b+1, \epsilon) \le 0$ then
    insert $b$ into finished$(d_b)$;
else
    $p'_{\min} = \arg\min_{p\in\text{packing}(d_b+1)} ||\tau(b,a^*,z^*) - p||$;
    if $||\tau(b,a^*,z^*) - p'_{\min}|| > \delta$ then
        insert $\tau(b,a^*,z^*)$ into packing$(d_b+1)$;
    end if
    if $||\tau(b,a^*,z^*) - p'_{\min}|| > \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$ then
        EXPLORE($\tau(b,a^*,z^*), d_b+1, \epsilon, \delta$);
    else if $p'_{\min} \notin$ finished$(d_b+1)$ then
        EXPLORE($p'_{\min}, d_b+1, \epsilon, \delta$);
    else
        insert $\tau(b,a^*,z^*)$ into finished$(d_b+1)$;
    end if
end if
Perform a point-based update of lower and upper bounds at belief $b$;
```

5. Packing-Guided Value Iteration

The algorithms in the proofs of Theorems 1 and 2 are designed to prove performance bounds, hence use depth-first search as the search strategy. Practical algorithms such as HSVI2 and SARSOP do a trial-based search, where the current bounds are used to select a path from b_0 to one leaf belief. The advantage of this is that the algorithm can be greedy with respect to the current best bounds, and this appears to be useful in practice. However, these algorithms do not really utilize the other insight of the analysis – that packing can be helpful in getting good performance.

In this section, we describe PGVI. It gets power by using the idea of building a separate packing of sampled beliefs at each level of the search tree in the proofs of our theorems. Such an idea can alleviate the curse of history in *both* theoretical and practical aspects. First, it controls the number of sampled beliefs at each level of search tree so that it does not grow exponentially in the depth of the tree (subject to $\mathcal{C}^{g,U}_{f,L}(\delta)$ being manageably small). Second, it can be used to spread the sampling areas in the search tree reduced by heuristics. PGVI achieves this by preferring to sample beliefs at level i which are far away from the beliefs in the packing of sampled beliefs at level i and that no belief in its δ-region has performed point-based updates recently.

5.1. PGVI Overview

PGVI is outlined in Algorithms 1 and 2. In these algorithms, we used the *packing* container to store a set of δ-packing and the *finished* container to store finished belief nodes at each level of the search tree in PGVI. Since PGVI is an extension of SARSOP, they have common points in selecting actions (Line 5), recursively invoking the EXPLORE function (Lines 15 and 17), and performing point-based updates (Line 22), as shown in Algorithm 2. We now emphasize some key differences between SARSOP and PGVI (see Algorithm 2). The first difference is the definition of finished belief nodes. Let $\text{excess}(b, d_b, \epsilon) = V^U(b) - V^L(b) - \frac{3\epsilon}{2\gamma^{d_b}} - \frac{\epsilon(1-\gamma^{h-d_b-1})}{2}$, where $h = \log_\gamma \frac{\epsilon(1-\gamma)}{2R_{\max}}$. A belief node b in the packing of sampled beliefs at level d_b, packing(d_b), is finished if $\text{excess}(b, d_b, \epsilon) \leq 0$. Let $p_{\min}(b) = \arg\min_{p \in \text{packing}(d_b)} ||b - p||$. A node b that is not in the packing(d_b) is finished if $\text{excess}(b, d_b, \epsilon) \leq 0$, or $||b - p_{\min}(b)|| \leq \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$ and $p_{\min}(b)$ is finished. The second difference is the observation selection strategy in Line 6. We define $Z^{\text{UF}} = \{z \in Z | \tau(b, a^*, z) \notin \text{finished}(d_b + 1)\}$. PGVI calculates the distance between $\tau(b, a^*, z)$ and the δ-packing of sampled beliefs at level $d_b + 1$, denoted $\text{dis}(\tau(b, a^*, z), \text{packing}(d_b + 1), \delta)$, to use it to spread the sampling in $\mathcal{R}^{g,U}_{f,L}(b_0)$. The new observation selection plays a key role in PGVI's efficient performance. The third difference is the criterion of forward exploration in Lines 7 $\sim$ 21. Line 7 is used to check whether all successors of b are finished, where b is a belief in the packing(d_b). If yes, Line 8 is executed to change b's status into finished. The belief $p'_{\min}$ in Line 10 is the belief in the packing$(d_b + 1)$ of sampled beliefs with minimal distance to the belief $\tau(b, a^*, z^*)$. Algorithm 2 inserts $\tau(b, a^*, z^*)$ into the packing at level $d_b + 1$ only if the distance is greater than δ (see Lines 11 $\sim$ 13). If the distance between $\tau(b, a^*, z^*)$ and $p'_{\min}$ is greater than $\frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$, it recursively invokes the EXPLORE function at $\tau(b, a^*, z^*)$. Otherwise, it checks whether $p'_{\min}$ is finished. If yes, it changes the status of $\tau(b, a^*, z^*)$ into finished. If not, it recursively invokes the EXPLORE function at $p'_{\min}$. Overall, Lines 14 $\sim$ 20 are used to control the width of the tree and the frequency of point-based updates to ensure PGVI's polynomial time performance, as stated later. If PGVI encounters a belief b at level d_b that is close to some belief b' in the packing at level d_b, it only explores forward from b' and terminates the forward search from b.

5.2. Packing-Guided Search

Besides using the set of δ-packing to control the width of the search tree, PGVI also uses it to guide search based on the two following principles. First, for beliefs b with at least δ distance to the δ-packing of sampled beliefs at level d_b of the tree, it prefers to sample beliefs that are far away from beliefs in packing(d_b). This is implemented by setting $\text{dis}(b, \text{packing}(d_b), \delta) = \min_{p \in \text{packing}(d_b)} ||b - p||$ when $\min_{p \in \text{packing}(d_b)} ||b - p|| > \delta$. Second, for beliefs with at most δ distance to the corresponding δ-packing, it biases sampling beliefs b that $p_{\min}(b)$ has not performed a point-based update recently. We record the time index of the last update at $p_{\min}(b)$ and denote it as $N(p_{\min}(b))$. Let N be the total number of point-based updates that have been performed by PGVI. We set $\text{dis}(b, \text{packing}(d_b), \delta) = \omega\delta$, where $\omega = (N + 1 - N(p_{\min}(b)))/(N + 1)$, when $\min_{p \in \text{packing}(d_b)} ||b - p|| \leq \delta$. Together, we define

$$\text{dis}(\tau(b, a^*, z), \text{packing}(d_b + 1), \delta)$$

$$= \begin{cases} \omega\delta \text{ if } \min_{p \in \text{packing}(d_b+1)} ||\tau(b, a^*, z) - p|| \leq \delta \\ \min_{p \in \text{packing}(d_b+1)} ||\tau(b, a^*, z) - p|| \quad \text{otherwise} \end{cases}$$

in Line 6 of Algorithm 2. The $\text{dis}(b, \text{packing}(d_b), \delta)$ is always greater than 0 in our setting. The key innovation here is to use the packing container to spread the sampling areas by modifying the observation selection strategy in HSVI2 and SARSOP to enable much better exploration.

5.3. Convergence

Theorem 3 shows that PGVI(ϵ, δ) terminates after performing at most $h^2\mathcal{C}^{g,U}_{f,L}(\delta/2)|A||Z|$ point-based updates. Its proof is a combination of the proof in HSVI2 style algorithm (Smith, 2007) and the proof in Theorem 2.

Theorem 3. *Given any constant $\epsilon > 0$ and any initial belief $b_0 \in \mathcal{B}$, PGVI(ϵ, δ) guarantees $V^*(b_0) - V^\pi(b_0) \leq 2\epsilon$ after at most $h^2\mathcal{C}^{g,U}_{f,L}(\delta/2)|A||Z|$ point-based updates, where $h = \log_\gamma \frac{\epsilon(1-\gamma)}{2R_{\max}}$, $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$, $V^L_f(b)$ and $V^U_g(b)$ are used as initial lower and upper bounds respectively.*

Proof. In the proof of Theorem 2, we have shown that if a point-based update is performed at b when all its children are finished, then b will also be finished. This shows the correctness of Lines 7 and 8 in the code.

Now we argue that each trial will switch one unfinished belief node into finished belief node if b_0 is still unfinished. From Algorithm 2 we can see a belief b is inserted into the finished(d_b) at the end of each trial. In Line 2 it inserts b in the packing(d_b) into finished nodes only when $\text{excess}(b, d_b, \epsilon) \leq 0$. In Line 8 it inserts b in the $\text{packing}(d_b)$ into finished nodes only when all of its children become finished. In Line 19 it inserts b not in the $\text{packing}(d_b)$ into finished nodes only when $p_{\min}(b)$ is finished. So, all nodes b inserted into the $\text{finished}(d_b)$ satisfy the condition of finished nodes.

PGVI only searches in $\mathcal{R}^{g,U}_{f,L}(b_0)$. From Lines 1 $\sim$ 3 and 7 $\sim$ 8 we can see that it does not search and perform point-based updates on nodes that satisfy $\text{excess}(b, d_b, \epsilon) \leq 0$. Since $\max_{a \in A} Q^U(b, a) \geq V^*(b)$, from Line 5 we can see that PGVI does not search towards action branches a that satisfy $Q^U(b, a) < V^*(b)$.

Finally, Algorithm 2 only searches the children of belief nodes b in the $\text{packing}(d_b)$. Thus, there are at most $\mathcal{C}^{g,U}_{f,L}(\delta/2)|A||Z|$ unfinished beliefs at each level of the search tree in PGVI. Totally, there are at most $h \cdot \mathcal{C}^{g,U}_{f,L}(\delta/2)|A||Z|$ unfinished beliefs. Each trial performs at most h point-based updates at beliefs. So, PGVI converges after performing at most $h^2 \cdot \mathcal{C}^{g,U}_{f,L}(\delta/2)|A||Z|$ updates. □

This theorem implies that PGVI converges with a number of updates that is quadratic polynomial rather than exponential in the planning horizon h, given there is no h's exponential term hidden in $\mathcal{C}^{g,U}_{f,L}(\delta)$.

6. Experiments

In this section, we compare PGVI with some existing point-based algorithms in their performance on 65 out of the 68 small benchmark problems from Cassandra's POMDP website[1] and 4 larger robotic problems (Ross et al., 2008; Hsu et al., 2008; Kurniawati et al., 2008; 2011). We discarded 3 of the 68 problems (1d.noisy, 4×4.95 and bulkhead.A) due to parsing issues. Our experimental platform is a CPU at 2.40GHz, with 3GB memory. We used the APPL-0.95 software package[2] to implement the PGVI algorithm, but did not use the MOMDP representation (Ong et al., 2010). We used α-vectors as lower bounds and sawtooth representations as upper bounds (Smith, 2007). Although the convergence proof of PGVI suggests using $\delta = \frac{(1-\gamma)^2\epsilon}{2\gamma R_{\max}}$, the value is not useful for achieving the best performance in practice because δ often becomes very small for problems with large $R_{\max}$ and γ. We set $\delta = (t_{\max} - t)\delta_0/t_{\max}$, where $\delta_0 = 0.5$, $t_{\max}$ represents the upper bound of running time, and t represents the elapsed time in running PGVI, to make PGVI do the best in the available time. Given that the value of δ changes with time, we use the simpler value of $\text{excess}(b, d_b, \epsilon) = V^U(b) - V^L(b) - \epsilon/\gamma^{d_b}$ to terminate trials. Theorem 3 still holds when using the simple one. In PGVI and SARSOP, ϵ is set to $0.5 \times [V^U(b_0) - V^L(b_0)]$ in the beginning of each trial.

6.1. 65 Small Benchmark Problems

We compared PGVI with HSVI2, SARSOP, and GapMin on the suite of 65 benchmark problems:

- PGVI found a near optimal solution (gap smaller than one unit at the third significant digit) in less than 1,000 seconds for 35 problems. In comparison, HSVI found 33, SARSOP found 32, and GapMin found 46 (Poupart et al., 2011).
- Among the 33 problems in Tables 1 and 2 of (Poupart et al., 2011) with 1,000 seconds limit, PGVI achieved the highest lower bound on 31 problems, while GapMin (LP) only on 1 problem. PGVI achieved the smallest gap on 12 problems, while GapMin (LP) on 19 problems. PGVI achieved smaller gap than HSVI2 on 27 problems and than SARSOP on all 33 problems.
- Among the 8 problems in Table 3 of (Poupart et al., 2011) with 50,000 seconds limit, PGVI achieved the highest lower bound on 6 of the problems and the smallest gap on 2 of the problems (cit, pentagon).

To summarize, the performance of PGVI is comparable to HSVI2, SARSOP, and GapMin on these small problems.

For the 35 problems on which PGVI performed well, we recorded the number of α-vectors ($|\Gamma|$), the number of beliefs expanded ($|B^s|$), the estimated δ-packing number of B^s ($\hat{\mathcal{P}}^{g,U}_{f,L}(\delta)$), with $\delta = 10^{-6}$, and the corresponding computation time ($Time$). The linear correlation coefficient between $\hat{\mathcal{P}}^{g,U}_{f,L}(\delta)$ and $Time$ is as high as 0.987, while the correlation coefficient between $|\Gamma|$ and $Time$ is only -0.045. This suggests that $\hat{\mathcal{P}}^{g,U}_{f,L}(\delta)$ is a good indicator of the running time of PGVI.

6.2. Larger Benchmark Problems

We now report PGVI and SARSOP's experimental results on 4 more challenging robotic tasks: FieldVisionRockSample[5,5] (Ross et al., 2008), Tracking (Hsu et al., 2008), Homecare (Kurniawati et al., 2008) and 3D Navigation (Kurniawati et al., 2011) (see Table 1). For lack of space, we present only the comparison with SARSOP in details. While GapMin variants have good performance on small benchmarks, the software package[3] provided by the authors is unable to scale up on these larger problems. In general, SARSOP outperforms HSVI2 (Kurniawati et al., 2008).

For each problem, we ran SARSOP and recorded the gap it achieved and other performance measurements when 10,000 seconds reached. Then, for all test problems except 3D Navigation, we recorded the time that PGVI needed to achieve the same gap and other performance measurements at that time point. For the 3D Navigation problem we chose a time point to better distinguish PGVI from SARSOP.

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

[2] http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/

[3] https://cs.uwaterloo.ca/ ppoupart/

Table 1. Performance comparison.

Algorithm	Reward	Gap	$V^L(b_0)$	$V^U(b_0)$	$\|\Gamma\|$	$\|B^s\|$	$\hat{\mathcal{P}}^{g,U}_{f,L}(\delta)$	T_0 (s)	$Time$ (s)
FieldVisionRockSample[5,5] ($\|S\| = 801, \|A\| = 5, \|Z\| = 32$)									
SARSOP	23.31± 0.14	0.47	23.27	23.74	24,289	10,187	6,204	0.2	9,764
PGVI	23.32± 0.14	0.47	23.29	23.76	14,207	6,473	5,107	0.2	2,570
Tracking ($\|S\| = 9,248, \|A\| = 9, \|Z\| = 1,264$)									
SARSOP	14.79± 0.14	0.69	14.41	15.10	24,174	2,240	1,762	893	9,998
PGVI	14.76± 0.15	0.69	14.43	15.12	7,734	1,601	1,365	893	2,294
Homecare ($\|S\| = 5,408, \|A\| = 9, \|Z\| = 928$)									
SARSOP	16.97± 0.14	3.43	16.55	19.98	27,134	5,242	2,624	410	9,987
PGVI	17.06± 0.14	3.43	16.56	19.99	7,583	3,452	2,338	410	1,819
3D Navigation ($\|S\| = 16,969, \|A\| = 5, \|Z\| = 14$)									
SARSOP	−63± 0	754,977	−100	754,877	2	38,541	31,717	12	9,934
PGVI	202,665±1,859	744,045	16,339	760,383	289	15,687	13,085	12	1,719

In Table 1, Column 2 lists the estimated expected total rewards for the computed policies and the 95% confidence intervals. Each pair of reward and confidence interval was received over 100,000 simulation runs respectively. Each simulation was terminated after 100 steps. Columns 3∼10 list the gap between $V^L(b_0)$ and $V^U(b_0)$, lower bound $V^L(b_0)$, upper bound $V^U(b_0)$, $|\Gamma|$, $|B^s|$, $\hat{\mathcal{P}}^{g,U}_{f,L}(\delta)$ with $\delta = 10^{-1}$, the lower and upper bounds initialization time T_0, and the total computation time (including T_0), respectively. PGVI found the same gaps (see Column 3) but required significantly fewer α-vectors, expanded beliefs and computation time than SARSOP (see Columns 6, 7 and 10). Compared with SARSOP, PGVI had higher $\hat{\mathcal{P}}^{g,U}_{f,L}(\delta)$ / $|B^s|$ empirically (see Columns 7 and 8), which captured well the fact that PGVI prefers to sample beliefs far away from the sampled set. Since SARSOP and PGVI used the blind policy and FIB methods to initialize bounds, their initialization time T_0 were the same as each other on each problem (see Column 9). On these larger POMDP problems, PGVI substantially outperformed SARSOP by 3.80 $\sim$ 5.78 times in terms of $Time$ (see Column 10), and by 3.80 $\sim$ 6.80 times in terms of $Time - T_0$ (see Columns 9 and 10).

Figure 1 compares the evolution of the bound gap over time between PGVI and SARSOP. We have two observations from it. First, PGVI achieved a smaller gap than SARSOP over time on each problem. This implies that PGVI is more efficient. Second, PGVI was risky in consuming more space in order to store the packing set. As shown on the 3D Navigation task, PGVI ran out of memory around 6,000 seconds. Here, we leave the space reduction of the packing set stored in PGVI as a future topic.

7. Conclusion

In this paper, we presented two theoretical results that respectively connect the complexity of approximate POMDP planning to covering numbers of the search spaces reduced by the upper bound, and both the lower and upper bounds of the optimal value function. We designed the novel PGVI algorithm by using the idea of building a separate packing at each level of the search tree. The set of packing is used to alleviate the curse of history by controlling the width of the search tree and spreading the sampling areas in the search space reachable under heuristics. Theoretically, PGVI guarantees to find an ϵ-optimal solution after performing at most $h^2\mathcal{C}^{g,U}_{f,L}(\delta/2)|A||Z|$ point-based updates. Empirically, PGVI outperformed SARSOP by 3.80 $\sim$ 6.80 times on 4 challenging robotic problems; it also showed a very efficient performance compared with other state-of-the-art point-based algorithms on 65 small benchmark problems from Cassandra's POMDP website.

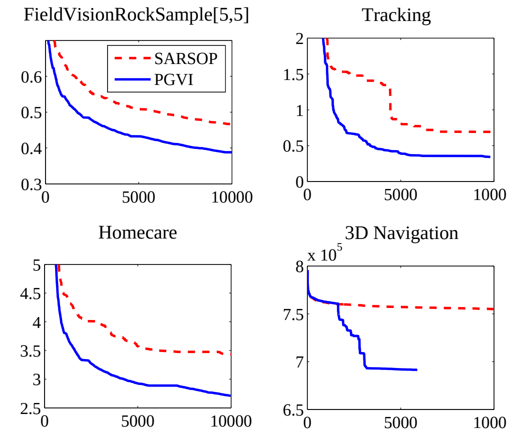

Figure 1. Evolution of the gap between upper and lower bounds (y-axis) over running time (x-axis) in PGVI and SARSOP.

Acknowledgements

Z. Zhang is supported in part by MoE ARF grant MOE2010-T2-2-071. D. Hsu is supported in part by the National Research Foundation Singapore through the Singapore MIT Alliance for Research and Technology's IRG research program (Subaward Agreement No. 41). W.S. Lee is supported in part by US Air Force Research Laboratory under agreement FA2386-12-1-4031.

References

Bonet, B. and Geffner, H. Solving POMDPs: RTDP-Bel vs. point-based algorithms. In *IJCAI*, pp. 1641–1646, 2009.

Grześ, M., Poupart, P., and Hoey, J. Isomorph-free branch and bound search for finite state controllers. In *IJCAI*, pp. 2282–2290, 2013.

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

Hsiao, K., Kaelbling, L.P., and Lozano-Perez, T. Grasping POMDPs. In *ICRA*, pp. 4685–4692, 2007.

Hsu, D., Lee, W.S., and Rong, N. What makes some POMDP problems easy to approximate. In *NIPS*, pp. 689–696, 2007.

Hsu, D., Lee, W.S., and Rong, N. A point-based POMDP planner for target tracking. In *ICRA*, pp. 2644–2650, 2008.

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

Kakade, S., Kearns, M., and Langford, J. Exploration in metric state spaces. In *ICML*, pp. 306–312, 2003.

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

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

Madani, O., Hanks, S., and Condon, A. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *AAAI*, pp. 541–548, 1999.

Ong, S.C., Png, S.W., Hsu, D., and Lee, W.S. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research*, 29 (8):1053–1068, 2010.

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

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

Poupart, P., Kim, K.E., and Kim, D. Closing the gap: Improved bounds on optimal POMDP solutions. In *ICAPS*, pp. 194–201, 2011.

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

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

Shani, G., Pineau, J., and Kaplow, R. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

Silver, D. and Veness, J. Monte-Carlo planning in large POMDPs. In *NIPS*, pp. 2164–2172, 2010.

Smith, T. *Probabilistic planning for robotic exploration*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2007.

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

Zhang, Z. and Chen, X. FHHOP: A factored hybrid heuristic online planning algorithm for large POMDPs. In *UAI*, pp. 934–943, 2012.

Zhang, Z., Littman, M.L., and Chen, X. Covering number as a complexity measure for POMDP planning and learning. In *AAAI*, pp. 1853–1859, 2012.