PLEASE: Palm Leaf Search for POMDPs with Large Observation Spaces

Zongzhang Zhang[a,b,c], **David Hsu**[c], **Wee Sun Lee**[c], **Zhan Wei Lim**[c], **Aijun Bai**[d]
[a]School of Computer Science and Technology, Soochow University, Suzhou 215006, P.R. China
[b]Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, P.R. China
[c]Department of Computer Science, National University of Singapore, Singapore 117417, Singapore
[d]School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230027, P.R. China
zzzhang@suda.edu.cn, {dyhsu, leews, limzhanw}@comp.nus.edu.sg, baj@mail.ustc.edu.cn

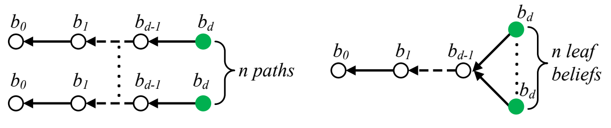
Abstract

Trial-based asynchronous value iteration algorithms for large Partially Observable Markov Decision Processes (POMDPs), such as HSVI2, FSVI and SARSOP, have made impressive progress in the past decade. In the forward exploration phase of these algorithms, only the outcome that has the highest potential impact is searched. This paper provides a novel approach, called Palm LEAf SEarch (PLEASE), which allows the selection of more than one outcome when their potential impacts are close to the highest one. Compared with existing trial-based algorithms, PLEASE can save considerable time to propagate the bound improvements of beliefs in deep levels of the search tree to the root belief because of fewer point-based value backups. Experiments show that PLEASE scales up SARSOP, one of the fastest algorithms, by orders of magnitude on some POMDP tasks with large observation spaces.

Introduction

Partially Observable Markov Decision Processes (POMDPs) provide a rich mathematical model for planning under uncertainty (Kaelbling, Littman, and Cassandra 1998). However, POMDP planning is notoriously hard due to the computational complexity of solving it exactly (Lusena, Goldsmith, and Mundhenk 2001). Consequentially, there have been much works on computing approximate POMDP solutions (Smith 2007; Ross et al. 2008; Bonet and Geffner 2009; Silver and Veness 2010; Shani, Pineau, and Kaplow 2013), including a number of *point-based POMDP algorithms* (Pineau, Gordon, and Thrun 2003; Smith and Simmons 2005; Pineau, Gordon, and Thrun 2006; Kurniawati, Hsu, and Lee 2008; Zhang, Hsu, and Lee 2014).

Most point-based POMDP algorithms use *value iteration*, which exploits the fact that the optimal value function must satisfy the *Bellman equation* (Bellman 1957). Value iteration algorithms start with an initial policy represented as a value function V and update values on V by iterating on the Bellman equation until the iteration converges. The idea behind point-based algorithms is to sample a representative set of beliefs from the entire belief space and to compute an approximately optimal value function by updating α-vectors (via *point-based value backups*) over the sampled beliefs.



(a) Trial-based search *(b) Palm leaf search*

Figure 1: Trial-based search vs. palm leaf search.

Several representative point-based algorithms do a *trial-based search* in their sampling strategies, where the current bounds of the optimal value function are used to select a *single path* from the initial belief b_0 to one leaf belief during each trial. To our knowledge, *few* existing (PO)MDP literature have discussed the implementation techniques and empirical results of selecting multiple actions and outcomes in trial-based search. Keller and Helmert (2013) suggested that this might be an opportunity of future research.

This paper provides a new method called *Palm LEAf SEarch* (PLEASE). We use Figure 1 to explain the motivation behind the method. Although the justification to some extent is simplistic, it helps us reveal two main characteristics of PLEASE in an easy way. Assume that, for a POMDP, n child beliefs of a belief at level $d-1$, denoted b_{d-1}, need to perform point-based value backups to propagate their bound improvements to the initial belief b_0 to guarantee to find a near optimal policy at b_0. For each trial, trial-based algorithms need to forward search one of b_{d-1}'s n child beliefs so as to propagate its improvement to b_0. Since each trial length is $d+1$, totally, there are $n \times (d+1)$ beliefs that need to perform backups. The PLEASE method allows the selection of *more than one* outcome during its forward exploration phase when their potential impacts are close to the highest one. Instead of repeatedly performing backups on b_i, where $i = 0, \cdots, d-1$, n times, PLEASE can perform them *once*. As a result, there are only $n + d$ beliefs that need to perform backups in PLEASE. When d or n becomes larger, $\frac{d(n-1)}{n+d}\left(=\frac{n(d+1)-(n+d)}{n+d}\right)$ increases. This suggests that PLEASE is more attractive when tackling a POMDP with *large* observation size (probably large n) and that needs to search *deeply* (large d) to find a near optimal solution.

In the challenging POMDP case with large observation space, it often happens that potential impacts of a few child beliefs are close to the highest one. A difficult problem is how to use the heuristics of beliefs to select a promising belief subset from them to make PLEASE perform best. We provide a novel heuristic threshold function, which depends on specific beliefs and is changeable over time, aiming to address the problem. Prior knowledge of POMDP model parameters is exploited in the function. Experimentally, PLEASE outperforms SARSOP by orders of magnitude on some difficult robotic tasks with large observation spaces.

Preliminaries

The POMDP framework is a rich mathematical model for single agent's sequential decision making to maximize its total reward in a partially observable and stochastic environment. It can be defined by a tuple $(S, A, Z, T, \Omega, R, \gamma)$, where S, A and Z are the finite and discrete state space, action space and observation space, respectively, and $\gamma \in (0, 1)$ is the discount factor. 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 $T(s, a, s') : S \times A \times S \to [0, 1]$, where $T(s, a, s') = Pr(s'|s, a)$. The agent then perceives an observation. The outcome of observing $z \in Z$ is given by an observation function $\Omega(a, s', z) : A \times S \times Z \to [0, 1]$, where $\Omega(a, s', z) = Pr(z|a, s')$. The reward function $R(s, a) : S \times A \to \mathbb{R}$ gives the agent a real-value reward after it takes action a in state s.

A *belief state* (or *belief*, *belief node*) b is a sufficient statistic for the history of actions and observations. A belief state space $\mathcal{B}$ is comprised of all possible belief states. When the agent takes action a at belief b and receives observation z, it will arrive at a new belief $b^{a,z}$ $\big(= \tau(b, a, z)\big)$:

$$b^{a,z}(s') = \frac{\Omega(a, s', z) \sum_{s \in S} T(s, a, s')b(s)}{Pr(z|b, a)}. \tag{1}$$

Here $Pr(z|b, a) = \sum_{s' \in S} \Omega(a, s', z) \sum_{s \in S} T(s, a, s')b(s)$. We denote b_0 as the initial belief state, and $T_{\mathcal{R}}$ as the belief tree rooted at b_0 consisting of beliefs reachable from b_0 by following an arbitrary policy. The *Average Discounted Return (ADR)* is given by $\mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|b_0]$, where s_t and a_t are the agent's underlying state and action at time t, respectively.

The goal of solving a POMDP planning problem is to find an *optimal policy* that maximizes the ADR. A POMDP *policy* $\pi : \mathcal{B} \to A$ maps a belief $b \in \mathcal{B}$ to a prescribed action $a \in A$. A policy π induces a *value function* $V^{\pi}(b)$ that specifies the ADR of executing π starting from b. The *optimal value function* V^*, the value function associated with the optimal policy π^*, can be approximated arbitrary well by a *piecewise-linear and convex* function:

$$V^*(b) \approx \max_{\alpha \in \Gamma^*}(\alpha \cdot b), \tag{2}$$

where Γ^* is the finite set of $|S|$-dimensional hyperplanes, called α-vectors, representing the optimal value function V^*. With each α-vector an action is associated, which can be used to obtain the best immediate policy at the current belief state. 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 $V^*(b) = \max_{a \in A} Q^*(b, a)$. The lower and upper bounds of V^* are denoted V^L and V^U, respectively. We assume that both V^L and V^U are *uniformly improvable* (Smith 2007) in this paper, meaning that applying a point-based value backup—α-vector's update for a selected belief—brings them everywhere closer to V^*. Similarly, we use Q^L, Q^U for the corresponding bounds of Q^*.

Trial-based Value Iteration

Asynchronous value iteration for POMDPs allows some belief states to be updated more than others, based on the assumption that the value function accuracy around the initial belief b_0 is more crucial for some beliefs in the reachable space $\mathcal{R}(b_0)$ (Shani, Brafman, and Shimony 2007). Trial-based search is a well-known form of asynchronous value iteration. Every trial-based value iteration algorithm executes a sequence of trials until a stopping criterion is satisfied. One popular stopping criterion is to stop when the gap between bounds at b_0 is small enough or the algorithm runs out of computational time. During each trial, a trajectory is created and only beliefs in the trajectory are performed backups. An important aspect for the efficiency of trial-based algorithms is a good heuristic that leads to important beliefs, on which backups are performed in some order.

RTDP-Bel (Geffner and Bonet 1998; Bonet and Geffner 2009) is the first trial-based algorithm for POMDPs. It discretizes beliefs and maintains Q-values for the discretized beliefs only. Trials are executed over the POMDP mapping the real belief into the closest discretized belief, which is then used for backups. HSVI2 (Smith and Simmons 2005), Focused RTDP (Smith 2007), FSVI (Shani, Brafman, and Shimony 2007), SARSOP (Kurniawati, Hsu, and Lee 2008) and GapMin (Poupart, Kim, and Kim 2011) all explore the reachable belief space $\mathcal{R}(b_0)$ from b_0 by repeated trials, and can be viewed as RTDP (Barto, Bradtke, and Singh 1995) variants in the POMDP field. Distinguishing from RTDP-Bel, who only uses the upper bound on V^*, these five practical algorithms use the lower bound as a heuristic in searching important beliefs. These algorithms can be specified in terms of five ingredients: heuristic function, backup function, action selection, outcome selection, and trial length. We briefly describe two of them as follows.

HSVI2

Algorithms 1 and 2 describe the details of HSVI2 (Smith and Simmons 2005). HSVI2 uses α-vectors as lower bounds and sawtooth representations as upper bounds. The initialized lower bound, denoted V_0^L, is obtained by using the blind policy (Hauskrecht 2000), and the initialized upper bound, denoted V_0^U, is obtained by the Fast Informed Bound (FIB) method (Hauskrecht 2000). Both the lower and upper bounds are used in the selection of action and observation (Lines 4 and 5 in Algorithm 2).

HSVI2 is always *action optimistic* – it chooses the action with the highest Q^U during the search process. The observation that HSVI2 selects is the one with the highest weighted excess uncertainty, where the excess function is defined as

Algorithm 1 $\pi = \text{HSVI2}(\epsilon)$.

```
1: Initialize the bounds $V^L$ and $V^U$;
2: while $V^U(b_0) - V^L(b_0) > \epsilon$ do
3:     EXPLORE($b = b_0, d_b = 0, \epsilon$);
4: end while
5: return the action corresponding to lower bound $V^L$;
```

Algorithm 2 EXPLORE(b, d_b, ϵ).

```
1: if excess($b, d_b, \epsilon$) $\leq 0$ then
2:     return ;
3: end if
4: $a^* = \arg\max_{a \in A} Q^U(b, a)$;
5: $z^* = \arg\max_{z \in Z}[Pr(z|b, a^*) \cdot \text{excess}(\tau(b, a^*, z), d_b + 1, \epsilon)]$;
6: EXPLORE($\tau(b, a^*, z^*), d_b + 1, \epsilon$);
7: add($V^L$, backup($b, V^L$));
8: add($\Upsilon^U$, backup($b, V^U$));
```

excess$(b, d_b, \epsilon) = V^U(b) - V^L(b) - \epsilon/\gamma^{d_b}$, and d_b is the depth of b in the reachable belief tree $T_{\mathcal{R}}$ rooted from b_0. Thus, HSVI2 prefers to visit belief states which have a bigger estimated gap between lower and upper bounds of V^* and have higher probabilities to be visited from b_0. The trial length is controlled by the stopping criterion that the gap between bounds of V^* at b is smaller than ϵ/γ^{d_b}. After finding a trajectory, HSVI2 performs backups on both bounds of V^* at beliefs (via backup(b, V^L) and backup(b, V^U) in Lines 7 and 8 of Algorithm 2) along the trajectory to b_0 reversely (or in round-trip order). In Line 8 of Algorithm 2, Υ^U is a finite set of belief/value points $(b, V^U(b))$. HSVI2 periodically prunes dominated elements in both the lower bound vector set and the upper bound point set.

SARSOP

Contrary to HSVI2, SARSOP uses a termination condition that allows selective deep sampling. It exploits the insight that, on some POMDP problems, some belief nodes with high returns lie deep in the tree $T_{\mathcal{R}}$. An efficient algorithm must allow the sampling path to go deep enough in order to reach them to achieve satisfiable performance. SARSOP uses a simple learning technique to predict the optimal value $V^*(b)$ for some belief nodes whose gaps between bounds are smaller than ϵ/γ^{d_b}. It gives preference to lower bound improvements and continues down a sampling path beyond the belief node with a gap of ϵ/γ^{d_b}, if it predicts that doing so likely leads to an improvement in the lower bound at the root belief b_0 (Kurniawati, Hsu, and Lee 2008). In comparison to the α-vector pruning strategy in HSVI2, the pruning step in SARSOP is more aggressive to improve computational efficiency of backup operations. The called δ-dominance pruning strategy tries to prune an α-vector if it is dominated by other α-vectors over the optimally reachable belief space $\mathcal{R}^*(b_0)$, rather than the entire belief space $\mathcal{B}$. Using these more effective sampling and pruning strategies, Kurniawati, Hsu, and Lee (2008) showed that SARSOP yields better performance than HSVI2 on several benchmark problems.

Algorithm 3 EXPLORE(b, d_b, ϵ) in PLEASE.

```
1: if $b$'s gap termination condition == true then
2:     return ;
3: end if
4: $a^* = \arg\max_{a \in A} Q^U(b, a)$;
5: $z^* = \arg\max_{z \in Z}[Pr(z|b, a^*) \cdot \text{excess}(\tau(b, a^*, z), d_b + 1, \epsilon)]$;
6: for $z \in Z$ do
7:     if $Pr(z|b, a^*) \cdot \text{excess}(\tau(b, a^*, z), d_b + 1, \epsilon) \geq \zeta \cdot Pr(z^*|b, a^*) \cdot \text{excess}(\tau(b, a^*, z^*), d_b + 1, \epsilon)$ then
8:         EXPLORE($\tau(b, a^*, z), d_b + 1, \epsilon$);
9:     end if
10: end for
11: add($V^L$, backup($b, V^L$));
12: add($\Upsilon^U$, backup($b, V^U$));
```

Palm Leaf Search

In this section, we describe the PLEASE algorithm and its variant, PLEASE-Z. We use SARSOP as an example to explain how to modify it into the PLEASE method. Compared with Algorithm 2, Algorithm 3 adds a for loop in Lines 6$\sim$10 [1]. This allows Algorithm 3 to search towards more than one outcome in its forward exploration phase when their potential impacts are close to the highest one.

The remaining question is which outcomes should be selected using heuristics to make PLEASE do best? An intuitive answer is that the closer one outcome's weighted excess uncertainty to the highest one, the more it deserves to be visited. We control the palm leaf search at beliefs by setting ζ online (Line 7 in Algorithm 3). For a POMDP, the optimal value of ζ for the best performance is unknown, however, we know it depends on specific belief b and probably changes over time. So, it is not good to set ζ as a constant. We define it as a function of b and θ, called $\zeta(b, \theta)$, where θ is changeable over time and independent of b. This threshold function is an important part of our method.

Suppose that users have prior knowledge on time bound spent in tackling a POMDP and its observation space size. It is reasonable that these factors determine the users' preference in doing palm leaf search. For example, too aggressive palm leaf search in handling POMDPs with small observation spaces in limited time is often unwise. To make PLEASE work well in all POMDP cases, we give users an input constant C (≥ 0) in exploiting the knowledge. The constant C is defined as the *desired* ratio of #PLEASE - #SARSOP to #SARSOP, where #PLEASE is the total number of backups in the PLEASE approach, and #SARSOP is the total number of backups on the paths selected by SARSOP's action and outcome selection strategies and its belief's gap termination condition. In contrast to SARSOP, $C > 0$ reduces the number of backups to propagate the bound improvements of selected leaf nodes to the root node.

We let PLEASE control θ online so that it obtains an ideal value of θ over time to make the actual ratio of #PLEASE to

[1]The complete SARSOP algorithm is sophisticated. Algorithm 3 ignores some technique details for the convenience of discussion.

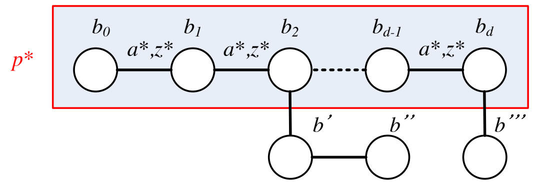


Figure 2: An example to explain dis(b, p^*), the distance from b to p^*.

#SARSOP close to $C+1$. Specifically, we define θ by using the following rule:

$$\theta = \begin{cases} \min\{\theta+\Delta, 1\} & \text{if } \frac{\#\text{PLEASE}}{\#\text{SARSOP}} \geq C+1, \\ \max\{\theta-\Delta, \theta_l\} & \text{otherwise.} \end{cases} \quad (3)$$

Here, θ is set to $\theta_0 = 1$ in the beginning of PLEASE, $\theta_l = 0.8$, and $\Delta = 0.01$ in this paper. PLEASE adjusts the value of θ in the beginning of each forward exploration phase from the initial belief b_0 in the PLEASE method.

Now, we define the function $\zeta(b, \theta)$ as $\sqrt[\text{dis}(b,p^*)+1]{\theta}$ by using θ and b's heuristic information. As suggested in the introduction section, PLEASE may be more efficient when it gives more time to do palm leaf search in promising beliefs with deep levels. PLEASE achieves this goal by doing more aggressive palm leaf search around the current best path p^* in each forward exploration phase starting at b_0. Here, p^* represents the path generated by SARSOP's action and outcome selection strategies and its belief's gap termination condition (Figure 2). Specifically, we define the distance from b to p^*, denoted dis(b, p^*), as the number of beliefs that b needs to go through to arrive in p^*. For example, in Figure 2, dis$(b_i, p^*) = 0$ for $i = 0, 1, \cdots, b_d$, dis$(b', p^*) =$ dis$(b''', p^*) = 1$ and dis$(b'', p^*) = 2$. Thus, in this criterion, PLEASE uses more time to explore promising belief nodes in deep levels only if their distances to the current best path p^* are small.

Variant: PLEASE-Z

We describe a variant of PLEASE with a different definition of C, called PLEASE-Z. The constant C in PLEASE-Z is defined as the desired ratio of #PLEASE - #MAXAZ to #MAXAZ, where #MAXAZ is the total number of backups on sampled beliefs that can be represented as $\tau(\cdot, a^*, z^*)$. In other words, #MAXAZ stores the number of sampled beliefs induced from the one-step best action and observation selection. We use Figure 3 to distinguish PLEASE-Z from PLEASE. Assume that the belief tree is sampled by PLEASE or PLEASE-Z. Thus, #PLEASE=12. The number of beliefs in p^* (#SARSOP), denoted by the solid black nodes, is 4. However, the number of beliefs that can be represented as $\tau(\cdot, a^*, z^*)$, denoted by both solid black and gray nodes, is 8. The white nodes represent the sampled beliefs that cannot be represented as $\tau(\cdot, a^*, z^*)$. Generally, #MAXAZ≥#SARSOP, which results in the good C in PLEASE-Z appears to be (much) smaller than the good C in PLEASE empirically.

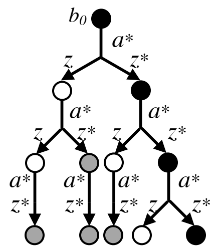


Figure 3: Example belief tree to distinguish PLEASE-Z from PLEASE.

Theoretical Analysis

Essentially, palm leaf search can be viewed as a kind of complete anytime best-first beam search (Zhang 1998) in POMDPs. Similar techniques of tree-trials have been used in Monte-Carlo tree search, such as df-UCT by Yoshizoe et al. (2011). At each step, the complexity of each PLEASE exploration step is at least the time and space complexity of each SARSOP step. However, its conservative theoretical time bound of finding an ϵ-optimal policy is verifiable to be not worse than the original trial-based algorithm by using HSVI2 style proof technique (Smith 2007).

In this section, we give proofs of the PLEASE method and its variant PLEASE-Z's convergence. We use HSVI2 style proof techniques to prove PLEASE's convergence. Based on it, a relatively tighter convergence bound can be found for the PLEASE-Z method.

PLEASE's Convergence

Proposition 1. *Let $\epsilon > 0$ and the threshold function $\zeta > 0$. Assume that the initialized upper and lower bounds, V_0^U and V_0^L, are uniformly improvable. Then PLEASE(ϵ) is guaranteed to terminate after performing at most $d_{\max}\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1}$ point-based value backups, where $d_{\max} = \lceil \log_\gamma \frac{\epsilon}{||V_0^U - V_0^L||_\infty} \rceil$.*

Proof. Recall that in PLEASE, sampled beliefs in each forward exploration process from b_0 is composed of a tree rooted at b_0, compared with a single path from b_0 in trial-based algorithms. The path length from b_0 to each leaf node in the tree is no longer than $d_{\max}$. At the beginning of executing PLEASE, there are totally $\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1}$ unfinished beliefs. Here, a finished belief satisfies excess$(b, d_b, \epsilon) < 0$ or its ancestor beliefs are finished. Thus, we only need to prove the following statement: when PLEASE explores towards beliefs under an observation branch, at least one unfinished belief under the branch will switch to the finished status after the exploration phase.

If PLEASE explores towards beliefs under an observation branch z from b after taking a, then excess$(\tau(b, a^*, z), d_b + 1, \epsilon) > 0$. Otherwise, the conditional statement in Line 7 of Algorithm 3 cannot be satisfied due to $\zeta > 0$. Let $b' = \tau(b, a^*, z)$. Now we consider the following two cases: (1) Its child beliefs $\tau(b', a^*, z)$ for all $z \in Z$ are finished, namely, excess$(\tau(b', a^*, z), d_b + 2, \epsilon) \leq 0$. In this case after

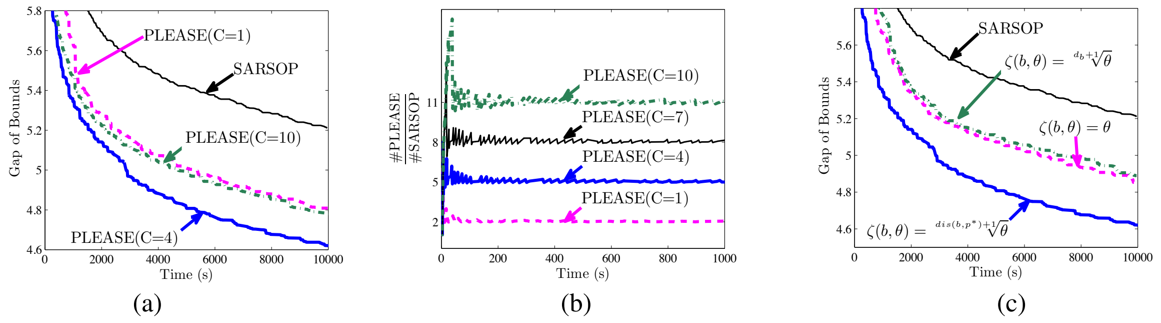


Figure 4: Results on Tag(55). See text for details.

performing a backup on b', b' will be switched to the finished status. (2) At least one of its child beliefs $\tau(b', a^*, z)$ is still unfinished. Then, EXPLORE($\tau(b', a^*, z), d_b + 2, \epsilon$) will switch $\tau(b', a^*, z)$ or one of its offsprings to the finished status. Thus, we get the final result. $\square$

PLEASE-Z's Convergence

Although PLEASE-Z seems less intuitive than PLEASE, we figure out that it has better theoretical guarantee. While PLEASE can only guarantee to find an ϵ-optimal solution after performing at most $u_{\max} = d_{\max}\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1} -$ #PLEASE point-based value backups, when it has performed point-based value backups #PLEASE times, PLEASE-Z can reduce $u_{\max}$ to $d_{\max}\Big(\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1} - (\#\text{PLEASE} - \#\text{MAXAZ})\Big) - \#\text{SARSOP}$ theoretically.

Proposition 2. *Let $\epsilon > 0$ and the threshold function $\zeta > 0$. Assume that the initialized upper and lower bounds, V_0^U and V_0^L, are uniformly improvable. Then PLEASE-Z(ϵ) is guaranteed to terminate after performing at most $d_{\max}\Big(\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1} - (\#\text{PLEASE} - \#\text{MAXAZ})\Big) -$ #SARSOP point-based value backups, where $d_{\max} = \lceil \log_\gamma \frac{\epsilon}{||V_0^U - V_0^L||_\infty} \rceil$, when it has performed point-based value backups #PLEASE times, including #MAXAZ times of backups on beliefs that can be represented as $\tau(\cdot, a^*, z^*)$ and #SARSOP times of backups on beliefs on the paths selected by SARSOP's action and outcome selection strategies and its belief's gap termination condition.*

Proof. Assume that the PLEASE-Z algorithm has called the EXPLORE($b = b_0, d_b = 0, \epsilon$) function #EXPLORE times after performing backups #PLEASE times. During executing each EXPLORE($b = b_0, d_b = 0, \epsilon$), one current best path p^* is always generated. Performing backups on beliefs on p^* can switch one unfinished belief to the finished status. We denote the number of beliefs on p^* during the i^{th} invocation of EXPLORE($b = b_0, d_b = 0, \epsilon$) as $L_i(p^*)$. Thus, $L_i(p^*) \le d_{\max}$ and

$$\#\text{SARSOP} = \sum_{i=1}^{\#\text{EXPLORE}} L_i(p^*) \le \#\text{EXPLORE} \cdot d_{\max}.$$

For each sampled belief that cannot be represented as $\tau(\cdot, a^*, z^*)$, there always exists such a unique trajectory that all successive beliefs are reached by applying the best action and observation selection strategies, namely, all successive beliefs along the trajectory can be represented as $\tau(\cdot, a^*, z^*)$. Performing backups on beliefs on each trajectory like this can switch at least one unfinished node to the finished node. The number of trajectories like this is #PLEASE − #MAXAZ after performing backups #PLEASE times. Thus, after performing backups #PLEASE times, at least

$$\#F = (\#\text{PLEASE} - \#\text{MAXAZ}) + \#\text{EXPLORE}$$

unfinished nodes switch to the finished status.

Since there are at most $\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1}$ unfinished nodes in the beginning of calling the PLEASE-Z algorithm, we figure out that at most

$$\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1} - \#F$$

unfinished nodes exist after performing backups #PLEASE times. So at most

$$\begin{aligned} & d_{\max}\Big(\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1} - (\#\text{PLEASE} - \#\text{MAXAZ})\Big) \\ & \qquad - d_{\max} \cdot \#\text{EXPLORE} \\ \le\ & d_{\max}\Big(\frac{(|A||Z|)^{d_{\max}+1}-1}{|A||Z|-1} - (\#\text{PLEASE} - \\ & \qquad \#\text{MAXAZ})\Big) - \#\text{SARSOP} \end{aligned}$$

backups are required to find an ϵ-optimal solution in the worst case. $\square$

Experiments

In this section, we mainly compare and analyze PLEASE and SARSOP's empirical performance on the two POMDP problems with large observation spaces: Tag(55) ($|S| = 3,080$, $|A| = 5$, $|Z| = 56$) and Two-Robot Tag ($|S| = 14,400$, $|A| = 25$, $|Z| = 625$) (Ong et al. 2010). Our experimental platform is a 16-core linux machine with 4GB memory, and each core is at 2.40GHz. PLEASE and SARSOP are implemented based on the APPL-0.95 software package[2].

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

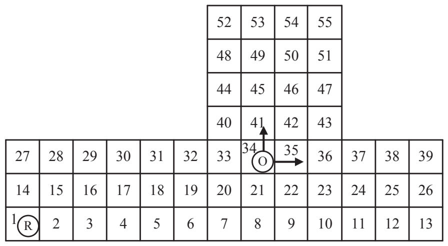


Figure 5: Tag(55) configuration. The "R" indicates the robot position, and the "O" indicates the opponent position.

Tag(55) Problem

The Tag problem was first described in the work of PBVI (Pineau, Gordon, and Thrun 2003). The environment has been used more recently in other POMDP planning work (Smith and Simmons 2005; Ong et al. 2010). The Tag(55) problem scales the configuration of the original Tag problem from 29 grids to 55 grids (Figure 5). In this environment, a mobile robot moves in a grid map with the goal of tagging an opponent that intentionally moves away. Both the robot and the opponent are located initially in independently selected random positions. The robot can choose either to move into one of four adjacent positions by actions {North, South, East, West} or to tag the opponent by a "Catch" action. The effect of the robot's action is deterministic. The robot can know its current position exactly, but the opponent's position is not observable for the robot unless they are in the same position. The robot tries to catch the opponent, i.e. arrive in the same position as that of the opponent, as quickly as possible to receive a good ADR since each move for the robot is expensive.

We use the problem to study the effects of varying the tuning parameter C and the heuristic function $\zeta(b,\theta)$.

Figure 4(a) shows the evolution of the gaps between upper and lower bounds of the optimal value at b_0 in SARSOP and PLEASE for varying C on Tag(55). "Gap between bounds" on the y-axis means $V^U(b_0)-V^L(b_0)$. For every algorithm, we ran it 10,000 seconds. PLEASE(C=4) achieves the gap 5.21 in only 1,527 seconds and the smallest gap 4.62 in 9,946 seconds, while SARSOP achieves the gap 5.21 in 10,000 seconds. PLEASE with $C = 1$ is less efficient than the corresponding ones with higher values of C. It suggests that the target ratio, $C = 1$, is still too small to result in significant time-saving. When setting C to be a value bigger than 4, PLEASE appears not to be as good as PLEASE(C=4). It implies that it becomes more difficult to select promising beliefs from very deep levels of the search tree (since the nodes increases exponentially) as θ in Equation 3 decreases.

Figure 4(b) shows the evolution of the actual ratio of #PLEASE to #SARSOP. As expected, $\frac{\#\text{PLEASE}}{\#\text{SARSOP}}$ quickly approaches to $C+1$, the desired ratio, as time goes on.

We tested two other heuristic ways of defining $\zeta(b,\theta)$, i.e., setting it to be θ and $\sqrt[d_b+1]{\theta}$, and empirical results with fixed C(=4) show that the two ways are less efficient (Figure 4(c)) in terms of the evolution of the gaps between bounds at b_0. The issue in setting $\zeta(b,\theta) = \theta$ is that b's heuristic is disused. The issue in setting $\zeta(b,\theta) = \sqrt[d_b+1]{\theta}$ is that the depth information is not well exploited to reduce the time of propagating bound improvements in deep levels to the root belief node.

Setting θ_l an arbitrary value in $(0, 0.9]$ should not affect the results we report here. It is because θ is always in $[0.95, 1.00]$ for PLEASE(C=4), and ranges from 0.90 to 1.00 for PLEASE(C=10) on Tag(55). It implies that the potential impacts of lots of outcomes are very close to the highest one.

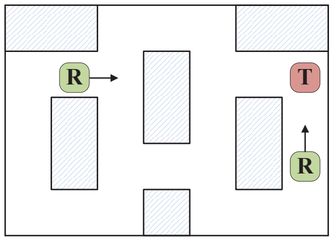


Figure 6: The configuration of the Two-Robot Tag problem. Stripped regions indicate obstacles.

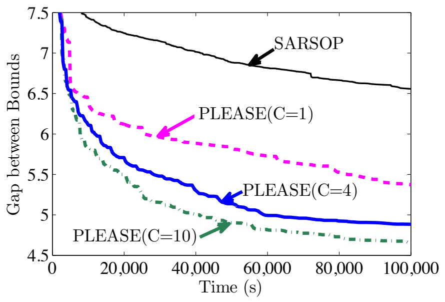


Figure 7: Results of SARSOP and PLEASE varying C on the Two-Robot Tag problem.

Two-Robot Tag Problem

This problem is a variation of Tag, and was introduced in (Ong et al. 2010). In this task, two robots (labeled "R") and a target (labeled "T") operate in a 7×5 grid environment, in which 11 grids are filled in the obstacles (Figure 6). Two robots attempt to catch the target that is always moving away from the closer robot, as quickly as possible to get a high ADR. Centralized planning and execution are assumed in the problem. The two robots maintain a communication link and share their knowledge of action and observation in decision making. This task is usually used to test the abilities of POMDP algorithms in handling more than one robot.

We use the larger robotic task to further study the effects of varying the input constant C. Figure 7 shows the evolution of the gaps between bounds of V^* at b_0 in SARSOP and PLEASE varying C on the Two-Robot Tag problem in 100,000 seconds. SARSOP needs 100,000 seconds to achieve the gap 6.56, while PLEASE(C=10) only needs 4,174 seconds and achieves the gap 4.65 in 100,000

Table 1: Performance comparison of SARSOP, PLEASE with $C = 3.22 \log_{10} |Z|$ and PLEASE-Z with $C = 0.13 \log_{10} |Z|$ on six other test problems.

Algorithm	Gap	$V^L(b_0)$	$V^U(b_0)$	Time
Tag(85) ($\lvert S\rvert = 7,310, \lvert A\rvert = 5, \lvert Z\rvert = 86$)				
SARSOP	7.02	-12.45	-5.43	9,995
PLEASE(C=6.23)	7.02	-12.18	-5.16	583
PLEASE-Z(C=0.25)	7.01	-12.14	-5.13	339
Tag(102) ($\lvert S\rvert = 10,506, \lvert A\rvert = 5, \lvert Z\rvert = 103$)				
SARSOP	7.68	-13.56	-5.88	9,993
PLEASE(C=6.48)	7.66	-13.15	-5.49	365
PLEASE-Z(C=0.26)	7.57	-13.27	-5.70	533
Hallway ($\lvert S\rvert = 60, \lvert A\rvert = 5, \lvert Z\rvert = 21$)				
SARSOP	0.18	1.01	1.19	9,995
PLEASE(C=4.26)	0.18	1.00	1.18	115
PLEASE-Z(C=0.17)	0.18	1.00	1.18	89
Hallway2 ($\lvert S\rvert = 92, \lvert A\rvert = 5, \lvert Z\rvert = 17$)				
SARSOP	0.46	0.42	0.88	9,990
PLEASE(C=3.96)	0.46	0.40	0.86	426
PLEASE-Z(C=0.16)	0.46	0.40	0.86	427
FieldVisionRockSample_5_5 ($\lvert S\rvert = 801, \lvert A\rvert = 5, \lvert Z\rvert = 32$)				
SARSOP	0.47	23.27	23.74	9,764
PLEASE(C=4.85)	0.47	23.28	23.75	3,585
PLEASE-Z(C=0.20)	0.45	23.29	23.74	3,755
HomeCare ($\lvert S\rvert = 5,408, \lvert A\rvert = 9, \lvert Z\rvert = 928$)				
SARSOP	2.98	16.77	19.75	99,686
PLEASE(C=9.56)	2.96	16.77	19.73	3,706
PLEASE-Z(C=0.39)	2.94	16.72	19.66	3,129

seconds. Setting $C = 10$ is a better choice than setting $C = 4$ on the larger problem. It implies that more aggressive palm leaf search is helpful to get better performance on problems with larger observation space sizes. Later, we will use the insight to construct a C's generator and test its performance on more benchmarks.

More Benchmarks

Now, we study PLEASE's efficiency on a large number of classical POMDP benchmarks with medium or large observation spaces, and PLEASE's performance degradation if applied to problems with small observation spaces.

We first compare PLEASE with SARSOP on six benchmark problems (Littman, Cassandra, and Kaelbling 1995; Ross and Chaib-Draa 2007; Hsu, Lee, and Rong 2008; Kurniawati, Hsu, and Lee 2008) in terms of the gap between bounds at b_0, $V^L(b_0)$, $V^U(b_0)$ and running time, as shown in Table 1. Figure 8 is the configuration of the Tag(N) domain, where $N = 13 \times i + 4 \times i$, used in Table 1. For each test problem, we report the gap that SARSOP achieved when 10,000 or 100,000 seconds reached, and the time that PLEASE needed to achieve the same gap. We use $C = m \log_{10} |Z|$, inspired by the fact that C should be larger when $|Z|$ increases, as a simple formula to set the input constant automatically. Here, $m = 3.22$ is fitted by using the least squares method, which uses $C = 4$, $|Z| = 56$ from the Tag(55) problem and $C = 10$, $|Z| = 625$ from the Two-Robot Tag problem as the training set. On most of these problems this table shows that PLEASE with $C = 3.22 \log_{10} |Z|$ is substantially faster than SARSOP by orders of magnitude.

We next compare PLEASE and SARSOP's performance

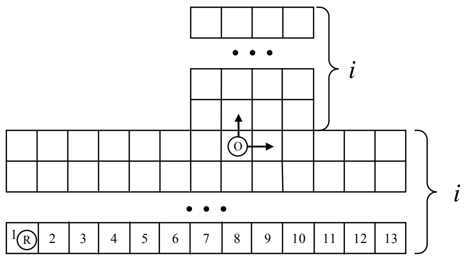


Figure 8: Tag(N) configuration, where $N = 13 \times i + 4 \times i$.

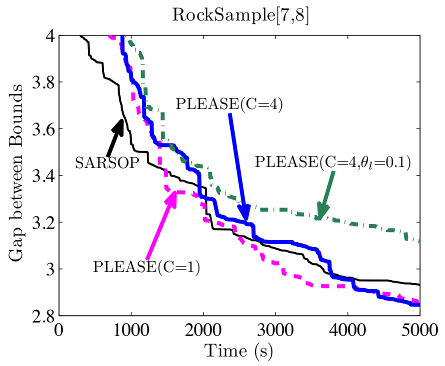


Figure 9: Results on RockSample[7,8].

on RockSample[7,8] ($|S| = 12,545$, $|A| = 13$, $|Z| = 2$) (Smith and Simmons 2004), a classical benchmark with extremely small observation size (Figure 9). PLEASE(C=1) appears to be better than SARSOP after 1,500 seconds in terms of the gap between bounds at b_0. It implies that palm leaf search is still helpful in the case of very small observation size, when searching more deeply as time goes on. As C increases (e.g., $C = 4$), PLEASE's performance becomes slightly worse. It might be because C is overestimated and not many promising beliefs in terms of potential impacts exist in the problem. When decreasing θ_l from 0.8 to 0.1, PLEASE with C=4's performance degenerates further since sampling unpromising beliefs wastes its computational time.

Further Discussion

Some existing works focus on developing efficient planning algorithms for POMDPs with large observation spaces (Atrash and Pineau 2006; Hoey and Poupart 2005). In 2006, Atrash and Pineau proposed to automatically find good low-dimensional observation spaces by clustering and principal component analysis. A way of finding a lossless partitioning of the observation space is discussed in (Hoey and Poupart 2005). Compared with these previous approaches, our PLEASE method explores promising observation branches without propagating back the bound improvements to the root frequently by allowing to select multiple outcomes adaptively during its forward exploration phase. PLEASE can be easily combined with several existing efficient POMDP planning algorithms. We did not compare PLEASE with these previous methods because their implementation

Table 2: Performance comparison of HSVI2 and SARSOP.

Algorithm	Gap	$V^L(b_0)$	$V^U(b_0)$	Time (s)
Tag(55)				
HSVI2	5.09	-10.14	-5.05	9,998
SARSOP	5.21	-9.89	-4.68	9,836
Two-Robot Tag				
HSVI2	8.85	-13.85	-5.00	9,997
SARSOP	7.43	-12.71	-5.28	9,998
HomeCare				
HSVI2	10.02	10.10	20.12	10,000
SARSOP	3.43	16.55	19.98	9,987

and experimental results on our test problems are not available for us yet.

From more detailed data in our experiments, we can obtain the three other observations as below.

First, the performance of HSVI2 on our test problems is worse than SARSOP in most cases, and GapMin variants are not efficient in tackling large problems. Table 2 compares HSVI2 and SARSOP's performance on the Tag(55), Two-Robot Tag and HomeCare problems. We ran both HSVI2 and SARSOP on each problem in 10,000 seconds, respectively. HSVI2's performance is slightly better on the Tag(55) problem, but much worse on the Two-Robot Tag and HomeCare problems. GapMin variants failed to load the three problems in 10,000 seconds.

Second, compared with SARSOP, PLEASE needs much fewer α-vectors to get the same gaps on test problems. For example, when the gap is 3.43 on HomeCare, the number of α-vectors in PLEASE(C=4) is 7,947, while 30,633 in SARSOP. However, the numbers of expanded beliefs in PLEASE(C=4) and SARSOP are similar, 5,575 and 5,700 respectively. These data provide a good clue to explain PLEASE's efficiency. To achieve the same gap, PLEASE(C=4) sometimes needs to expand beliefs with the similar numbers. But PLEASE(C=4) needs fewer backups in finishing this. Since one backup generates an α-vector, more α-vectors are generated in SARSOP. This results in additional time cost in the pruning process of SARSOP. Although the pruning strategy in SARSOP can remove some unnecessary α-vectors, it still can not prune the α-vectors that are not δ-dominated by others to find a near optimal solution. As a result, more vectors have to be stored as lower bound in SARSOP, and the time-consuming of each backup increases as time goes on.

Third, when the gaps are the same, SARSOP and PLEASE's ADRs appear to be similar with each other. Specifically, the ADR is -9.73 ± 0.12 on the Tag(55) problem when the gap is 5.21, the ADR is -11.58 ± 0.12 on the Two-Robot Tag problem when the gap is 7.43, and the ADR is 17.03 ± 0.14 on the HomeCare problem when the gap is 3.43. This implies that relatively fewer vectors as lower bound in PLEASE do not sacrifice its policy quality.

Results of PLEASE-Z

Up to now, our experimental discussion only focuses on the PLEASE algorithm. Note that we also implemented PLEASE-Z and obtained its empirical results on all test problems. Empirically, PLEASE-Z(C=0.16) achieves the smallest gap 4.60 in 10,000 seconds on the Tag(55) problem and PLEASE-Z(C=0.40) achieves the smallest gap 4.54 in 100,000 seconds on the Two-Robot Tag problem (Figure 10). Table 1 shows PLEASE-Z's empirical results on the six problems. Similarly to PLEASE, PLEASE-Z in Table 1 sets $C = 0.13 \log_{10} |Z|$, and the value 0.13 is fitted by using the least squares method with the training set: $C = 0.16$, $|Z| = 56$ from the Tag(55) problem and $C = 0.40$, $|Z| = 625$ from the Two-Robot Tag problem. As expected, PLEASE-Z requires a relatively smaller C to achieve good performance. Overall, PLEASE-Z's empirical performance is slightly better than PLEASE on test problems.

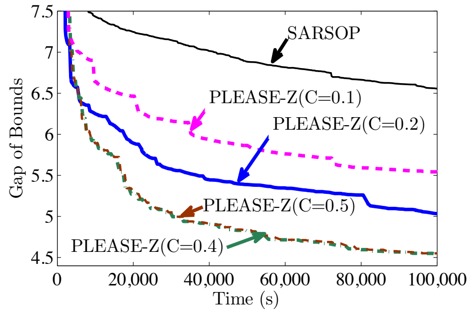


Figure 10: Results of SARSOP and PLEASE-Z varying C on the Two-Robot Tag problem.

Conclusion and Future Work

We present a novel approach, called PLEASE, which allows the selection of more than one outcome when their potential impacts are close to the highest one. PLEASE uses a heuristic way to find promising beliefs in deep levels during the palm leaf search process. Experimentally, PLEASE(-Z) is faster than SARSOP by orders of magnitude on challenging POMDP problems with large observation spaces and robust for problems with extremely small observation spaces.

One weak point of the current PLEASE method is the lack of theoretical analysis of its heuristic for observation selection, although it is intuitive and works quite well empirically. Another interesting topic is how to use more heuristic information of beliefs (e.g., the depth information of beliefs) in defining the threshold function $\zeta(b, \theta)$ and how to design a better function for choosing C automatically by considering $|S|$ and $|A|$ in the model parameters, to get better performance. We would like to know whether PLEASE can be further accelerated by allowing to select multiple actions at each belief state during the forward exploration phase. Besides offline algorithms we discussed before, online algorithms, such as POMCP (Silver and Veness 2010), FHHOP (Zhang and Chen 2012) and DESPOT (Somani et al. 2013), appear to be very promising in handling large POMDPs. Comparing them with the PLEASE method and considering how to use the palm leaf search idea into them should be a valuable topic for future exploration.

Acknowledgements

This work was supported in part by MoE AcRF grant 2010-T2-2-071, Collaborative Innovation Center of Novel Software Technology and Industrialization, and National Natural Science Foundation of China (61272005, 61303108, 61373094, 61472262). We thank the Juekun Li and anonymous reviewers for helpful comments.

References

Atrash, A., and Pineau, J. 2006. Efficient planning and tracking in POMDPs with large observation spaces. In *AAAI-06 Workshop on Empirical and Statistical Approaches for Spoken Dialogue Systems*.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1):81–138.

Bellman, R. 1957. *Dynamic programming*. Princeton University Press, Princeton, NJ, USA.

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

Geffner, H., and Bonet, B. 1998. Solving large POMDPs using real time dynamic programming. In *Proceedings of AAAI Fall Symposium on POMDPs*.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. 13:33–94.

Hoey, J., and Poupart, P. 2005. Solving POMDPs with continuous or large discrete observation spaces. In *IJCAI*, 1332–1338.

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

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

Keller, T., and Helmert, M. 2013. Trial-based heuristic tree search for finite horizon MDPs. In *ICAPS*, 135–143.

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

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *ICML*, 362–370.

Lusena, C.; Goldsmith, J.; and Mundhenk, M. 2001. Nonapproximability results for partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 14(1):83–103.

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

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

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

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

Ross, S., and Chaib-Draa, B. 2007. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *IJCAI*, 2592–2598.

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

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

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

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

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *UAI*, 520–527.

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

Smith, T. 2007. *Probabilistic planning for robotic exploration*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Somani, A.; Ye, N.; Hsu, D.; and Lee, W. 2013. DESPOT: Online POMDP planning with regularization. In *NIPS*, 1772–1780.

Yoshizoe, K.; Kishimoto, A.; Kaneko, T.; Yoshimoto, H.; and Ishikawa, Y. 2011. Scalable distributed Monte-Carlo tree search. In *SoCS*, 180–187.

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

Zhang, Z.; Hsu, D.; and Lee, W. S. 2014. Covering number for efficient heuristic-based POMDP planning. In *ICML*, 28–36.

Zhang, W. 1998. Complete anytime beam search. In *AAAI*, 425–430.