APPEARED IN
Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR), 2016

Importance Sampling for Online Planning under Uncertainty

Yuanfu Luo, Haoyu Bai, David Hsu, and Wee Sun Lee

National University of Singapore, Singapore 117417, Singapore

Abstract. The partially observable Markov decision process (POMDP) provides a principled general framework for robot planning under uncertainty. Leveraging the idea of Monte Carlo sampling, recent POMDP planning algorithms have scaled up to various challenging robotic tasks, including, e.g., real-time online planning for autonomous vehicles. To further improve online planning performance, this paper presents IS-DESPOT, which introduces *importance sampling* to DESPOT, a state-of-the-art sampling-based POMDP algorithm for planning under uncertainty. Importance sampling improves the planning performance when there are critical, but rare events, which are difficult to sample. We prove that IS-DESPOT retains the theoretical guarantee of DESPOT. We present a general method for learning the importance sampling distribution and demonstrate empirically that importance sampling significantly improves the performance of online POMDP planning for suitable tasks.

1 Introduction

Uncertainty in robot control and sensing presents significant barriers to reliable robot operation. The partially observable Markov decision process (POMDP) provides a principled general framework for robot decision making and planning under uncertainty [20]. While POMDP planning is computationally intractable in the worst case, approximate POMDP planning algorithms have scaled up to a wide variety of challenging tasks in robotics and beyond, e.g., autonomous driving [1], grasping [8], manipulation [13], disaster rescue management [25], and intelligent tutoring systems [4]. Many of these recent advances leverage probabilistic sampling for computational efficiency. This work investigates efficient sampling distributions for planning under uncertainty under the POMDP framework.

A general idea for planning under uncertainty is to sample a finite set of "scenarios" that capture uncertainty approximately and compute an optimal or near-optimal plan under these sampled scenarios on the average. Theoretical analysis reveals that a small sampled set guarantees near-optimal online planning, provided there exists an optimal plan with a compact representation [22]. In practice, the sampling distribution may have significant effect on the planning performance. Consider a de-mining robot navigating in a mine field. Hitting a mine may have low probability, but severe consequence. Failing to sample these rare, but critical events often results in sub-optimal plans.

Importance sampling [10] provides a well-established tool to address this challenge. We have developed IS-DESPOT, which applies importance sampling to DESPOT [22], a state-of-the-art sampling-based online POMDP algorithm (Section 3). The idea is to sample probabilistic events according to their "importance" instead of their natural probability of occurrence and then reweight the samples when computing the plan. We

prove that IS-DESPOT retains the theoretical guarantee of DESPOT. We also present a general method for learning the importance sampling distribution (Section 4). Finally, we present experimental results showing that importance sampling significantly improves the performance of online POMDP planning for suitable tasks (Section 5).

2 Background

2.1 POMDP Preliminaries

A POMDP models an agent acting in a partially observable stochastic environment. It is defined formally as a tuple (S, A, Z, T, O, R, b_0), where S, A and Z are the state space, the action space, and the observation space, respectively. The function $T(s, a, s') = p(s'|s, a)$ defines the probabilistic state transition from $s \in S$ to $s' \in S$, when the agent takes an action $a \in A$. It can model imperfect robot control and environment changes. The function $O(s, a, z) = p(z|s, a)$ defines a probabilistic observation model, which can capture robot sensor noise. The function $R(s, a)$ defines a real-valued reward for the agent when it takes action $a \in A$ in state $s \in S$.

Because of imperfect sensing, the agent's state is not known exactly. Instead, the agent maintains a *belief*, which is a probability distribution over S. The agent starts with an initial belief b_0. At time t, it infers a new belief, according to Bayes' rule, by incorporating information from the action a_t taken and the observation z_t received:

$$b_t(s') = \tau(b_{t-1}, a_t, z_t) = \eta O(s', a_t, z_t) \sum_{s \in S} T(s, a_t, s') b_{t-1}(s), \tag{1}$$

where η is a normalizing constant.

A POMDP *policy* maps a belief to an action. The goal of POMDP planning is to choose a policy π that maximizes its *value*, i.e., the expected total discounted reward, with initial belief b_0:

$$V_\pi(b_0) = \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_{t+1}) \;\middle|\; b_0, \pi \right) \tag{2}$$

where s_t is the state at time t, $a_{t+1} = \pi(b_t)$ is the action taken at time t to reach s_{t+1}, and $\gamma \in (0, 1)$ is a discount factor. The expectation is taken over the sequence of uncertain state transitions and observations over time.

A key idea in POMDP planning is the *belief tree* (Fig. 1*a*). Each node of a belief tree corresponds to a belief b. At each node, the tree branches on every action $a \in A$ and every observation $z \in Z$. If a node b has a child node b', then $b' = \tau(b, a, z)$. To find an optimal plan, one way is to traverse the tree from the bottom up and and compute an optimal action recursively at each node using the Bellman's equation:

$$V^*(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s) R(s, a) + \gamma \sum_{z \in Z} p(z|b, a) V^*\big(\tau(b, a, z)\big) \right\}. \tag{3}$$

2.2 Importance Sampling

We want to calculate the expectation $\mu = \mathbb{E}\big(f(s)\big) = \int f(s)p(s)\,\mathrm{d}s$ for a random variable s distributed according to p, but the function $f(s)$ is not integrable. One idea is to estimate μ by Monte Carlo sampling: $\hat{\mu} = (1/n)\sum_{i=1}^{n} f(s_i)$, where $s_i \sim p$ for all samples $s_i, i = 1, 2, \ldots, n$. The estimator $\hat{\mu}$ is unbiased, with variance $\mathrm{Var}(\hat{\mu}) = \sigma^2/n$, where $\sigma^2 = \int (f(s) - \mu)^2 p(s)\,\mathrm{d}s$.

Importance sampling reduces the variance of the estimator by carefully choosing an *importance distribution* q for sampling instead of using p directly:

$$\hat{\mu}_{\text{UIS}} = \frac{1}{n}\sum_{i=1}^{n} \frac{f(s_i)p(s_i)}{q(s_i)} = \frac{1}{n}\sum_{i=1}^{n} f(s_i)w(s_i), \quad s_i \sim q, \tag{4}$$

where $w(s_i) = p(s_i)/q(s_i)$ is defined as the *importance weight* of the sample s_i and $q(s) \neq 0$ whenever $f(s)p(s) \neq 0$. The estimator $\hat{\mu}_{\text{UIS}}$ is also unbiased, with variance $\mathrm{Var}(\hat{\mu}) = \sigma^2/n$, where $\sigma^2 = \int \big(\frac{f(s)p(s)}{q(s)} - \mu\big)^2 q(s)\,\mathrm{d}s$. Clearly the choice of q affects the estimator's variance. The optimal importance distribution $q^*(s) = |f(s)|\,p(s) \big/ \mathbb{E}_p\big(\,|f(s)|\,\big)$ gives the lowest variance [16].

When either p or q is unnormalized, an alternative estimator normalizes the importance weights:

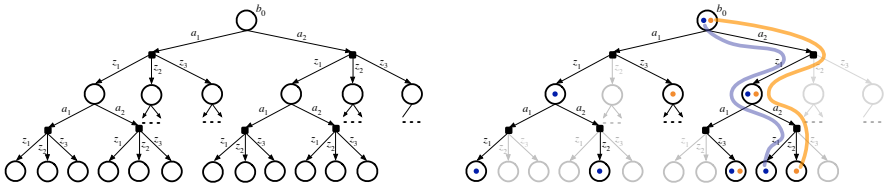
$$\hat{\mu}_{\text{NIS}} = \frac{\sum_{i=1}^{n} f(s_i)w(s_i)}{\sum_{i=1}^{n} w(s_i)}, \quad s_i \sim q, \tag{5}$$

which requires $q(s) \neq 0$ whenever $p(s) \neq 0$. This estimator is biased, but is asymptotically unbiased as the number of samples increases [16]. The performance of $\hat{\mu}_{\text{NIS}}$ versus that of $\hat{\mu}_{\text{UIS}}$ is problem-dependent. While $\hat{\mu}_{\text{NIS}}$ is biased, it often has lower variance than $\hat{\mu}_{\text{UIS}}$ in practice, and the reduction in variance often outweighs the bias [12].

2.3 Related Work

There are two general approaches to planning under uncertainty: offline and online. Under the POMDP framework, the offline approach computes beforehand a policy contingent on all possible future events. Once computed, the policy can be executed online very efficiently. While offline POMDP algorithms have made dramatic progress in the last decade [15,17,21,23], they are inherently limited in scalability, because the number of future events grows exponentially with the planning horizon. In contrast, the online approach [18] interleaves planning and plan execution. It avoids computing a policy for all future events beforehand. At each time step, it searches for a single best action for the current belief only, executes the action, and updates the belief. The process then repeats at the new belief. The online approach is much more scalable than the offline approach, but its performance is limited by the amount of online planning time available at each time step. The online and offline approaches are complementary and can be combined in various ways to further improve planning performance [5,7]

Our work focuses on online planning. POMCP [19] and DESPOT [22] are among the fastest online POMDP algorithms available today, and DESPOT has found applications in a range of robotics tasks, including autonomous driving in a crowd [1], robot de-mining in Humanitarian Robotics and Automation Technology Challenge 2015 [27],



(*a*) (*b*)

Fig. 1: Online POMDP planning performs lookahead search on a tree. (*a*) A standard belief tree of height $D = 2$. Each belief tree node represents a belief. At each node, the tree branches on every action and observation. (*b*) A DESPOT (black), obtained under 2 sampled scenarios marked with blue and orange dots, is overlaid on the standard belief tree. A DESPOT contains all actions branches, but only sampled observation branches.

and push manipulation [14]. One key idea underlying both POMCP and DESPOT is to use Monte Carlo simulation to sample future events and evaluate the quality of candidate policies. A similar idea has been used in offline POMDP planning [3]. It is, however, well known that standard Monte Carlo sampling may miss rare, but critical events, resulting in sub-optimal actions. Importance sampling is one way to alleviate this difficulty and improve planning performance.

We are not aware of prior use of importance sampling for robot planning under uncertainty. However, importance sampling is a well-established probabilistic sampling technique and has applications in many fields, e.g., Monte Carlo integration [10], ray tracing for computer graphic rendering [24], and option pricing in finance [6].

3 IS-DESPOT

3.1 Overview

Online POMDP planning interleaves planning and action execution. At each time step, the robot computes a near-optimal action a^* at the current belief b by searching a belief tree with the root node b and applies (3) at each tree node encountered during the search. The robot executes the action a^* and receives a new observation z. It updates the belief with a^* and z, through Bayesian filtering (1). The process then repeats. A belief tree of height D contains $\mathcal{O}(|A|^D|Z|^D)$ nodes. The exponential growth of the tree size is a major challenge for online planning when a POMDP has large action space, large observation space, or long planning horizon.

The *DEterminized Sparse Partially Observable Tree* (DESPOT) is a sparse approximation of the belief tree, under K sampled "scenarios" (Fig. 1*b*). The belief associated with each node of a DESPOT is approximated as a set of sampled states. A DESPOT contains all the action branches of a belief tree, but only the sampled observation branches. It has size $\mathcal{O}(|A|^D K)$, while a corresponding full belief tree has size $\mathcal{O}(|A|^D|Z|^D)$. Interestingly, K is often much smaller than $|Z|^D$ for a DESPOT to approximate a belief tree well, under suitable conditions. Now, to find a near-optimal action, the robot searches a DESPOT instead of a full belief tree. The reduced tree size leads to much faster online planning.

Clearly the sampled scenarios have major effect on the optimality of the chosen action. The original DESPOT algorithm samples scenarios with their natural probability of occurrence, according to the POMDP model. It may miss those scenarios that incur large reward or penalty, but happen with low probability. To address this issue, IS-DESPOT samples from an *importance distribution* and reweights the samples, using (4) or (5). We show in this section that IS-DESPOT retains the theoretical guarantee of DESPOT for all reasonable choices of the importance distribution. We also demonstrate empirically that IS-DESPOT significantly improves the planning performance for good choices of the importance distribution (see Section 5).

3.2 DESPOT with Importance Sampling

We define the DESPOT constructively by applying a *deterministic simulative model* to all possible action sequences under K sampled scenarios. Formally, a *scenario* $\boldsymbol{\phi}_b = (s_0, \varphi_1, \varphi_2, \ldots)$ for a belief b consists of a state s_0 sampled from b and a sequence of random numbers $\varphi_1, \varphi_2, \ldots$ sampled independently and uniformly over the range $[0, 1]$. The deterministic simulative model is a function $\mathcal{G}\colon S \times A \times \mathrm{R} \mapsto S \times Z$, such that if a random number φ is distributed uniformly over $[0, 1]$, then $(s', z') = \mathcal{G}(s, a, \varphi)$ is distributed according to $p(s', z'|s, a) = T(s, a, s')O(s', a, z')$. Intuitively, $\mathcal{G}$ performs one-step simulation of the POMDP model. It is deterministic simulation of a probabilistic model, because the outcome is fixed by the input random number φ. To simulate a sequence of actions $(a_1, a_2, \ldots)$ under a scenario $\boldsymbol{\phi}_b = (s_0, \varphi_1, \varphi_2, \ldots)$, we start at s_0 and apply the deterministic simulative model $\mathcal{G}$ at each time step. The resulting simulation sequence $\boldsymbol{\zeta} = (s_0, a_1, s_1, z_1, a_2, s_2, z_2, \ldots)$ traverses a path $(a_1, z_1, a_2, z_2, \ldots)$ in the belief tree, starting at its root (Fig. 1*b*). The nodes and edges along the path are added to the DESPOT. Further, each belief node contains a set of sampled states, commonly called a *particle set*, which approximates the corresponding belief. If $\boldsymbol{\zeta}$ passes through the node b at time step t, the state s_t is added to the particle set for b. Repeating this process for all possible action sequences under all K sampled scenarios completes the construction of the DESPOT. Clearly, the size of a DESPOT with height D is $\mathcal{O}(|A|^D K)$.

A DESPOT policy π can be represented as a policy tree derived from a DESPOT $\mathcal{T}$. The policy tree contains the same root as $\mathcal{T}$, but it contains at each internal node b only one action branch determined by $a = \pi(b)$. We define the size of such a policy, $|\pi|$, as the number of internal policy tree nodes. A singleton policy tree thus has size 0.

Given an initial belief b, the value of a policy π can be approximated by integrating over $\mathcal{Z}$, the space of all possible D-step simulation sequences under π:

$$V_\pi(b) \approx \int_{\boldsymbol{\zeta} \in \mathcal{Z}} V_{\boldsymbol{\zeta}}\, p(\boldsymbol{\zeta}|b, \pi)\, \mathrm{d}\boldsymbol{\zeta},$$

where $p(\boldsymbol{\zeta}|b, \pi) = b(s_0) \prod_{t=0}^{D-1} p(s_{t+1}, z_{t+1}|s_t, a_{t+1})$ is the probability of $\boldsymbol{\zeta}$ and $V_{\boldsymbol{\zeta}} = \sum_{t=0}^{D-1} \gamma^t R(s_t, a_{t+1})$ is the total discounted reward of $\boldsymbol{\zeta}$. To estimate $V_\pi(b)$ by unnormalized importance sampling (4), IS-DESPOT samples a subset $\mathcal{Z}' \subset \mathcal{Z}$ according to an importance distribution

$$q(\boldsymbol{\zeta}|b, \pi) = q(s_0) \prod_{t=0}^{D-1} q(s_{t+1}, z_{t+1}|s_t, a_{t+1}), \tag{6}$$

where $q(s_0)$ is the distribution for sampling the initial state and $q(s_{t+1}, z_{t+1}|s_t, a_{t+1})$ is the distribution for sampling the state transitions and observations. Then,

$$\hat{V}_\pi(b) = \frac{1}{|\mathcal{Z}'|} \sum_{\boldsymbol{\zeta} \in \mathcal{Z}'} w(\boldsymbol{\zeta}) V_{\boldsymbol{\zeta}} = \frac{1}{|\mathcal{Z}'|} \sum_{\boldsymbol{\zeta} \in \mathcal{Z}'} \sum_{t=0}^{D-1} w(\boldsymbol{\zeta}_{0:t}) \gamma^t R(s_t, a_{t+1}), \tag{7}$$

where $w(\boldsymbol{\zeta}) = p(\boldsymbol{\zeta}|b, \pi)/q(\boldsymbol{\zeta}|b, \pi)$ is the importance weight of $\boldsymbol{\zeta}$, $\boldsymbol{\zeta}_{0:t}$ is a subsequence of $\boldsymbol{\zeta}$ over the time steps $0, 1, \ldots, t$, and $w(\boldsymbol{\zeta}_{0:t})$ is the importance weight of $\boldsymbol{\zeta}_{0:t}$.

To avoid over-fitting to the sampled scenarios, IS-DESPOT optimizes a regularized objective function:

$$\max_{\pi \in \Pi_\mathcal{T}} \left\{ \hat{V}_\pi(b) - \lambda |\pi| \right\}, \tag{8}$$

where $\Pi_\mathcal{T}$ is the set of all policy trees derived from a DESPOT $\mathcal{T}$ and $\lambda \geq 0$ is regularization constant. More details on the benefits of regularization are available in [26].

3.3 Online Planning

IS-DESPOT is an online POMDP planning algorithm. At each time step, IS-DESPOT searches a DESPOT $\mathcal{T}$ rooted at the current belief b_0. It obtains a policy π that optimizes (8) at b_0 and chooses the action $a = \pi(b_0)$ for execution.

To optimize (8), we substitute (7) into (8) and define the *regularized weighted discounted utility* (RWDU) of a policy π at each DESPOT node b:

$$\nu_\pi(b) = \frac{1}{|\mathcal{Z}'|} \sum_{\boldsymbol{\zeta} \in \mathcal{Z}'_b} \sum_{t=\Delta(b)}^{D-1} w(\boldsymbol{\zeta}_{0:t}) \gamma^t R(s_t, a_{t+1}) - \lambda |\pi_b|, \tag{9}$$

where $\mathcal{Z}'_b \subset \mathcal{Z}'$ contains all simulation sequences traversing paths in $\mathcal{T}$ through the node b; $\Delta(b)$ is the depth of b in $\mathcal{T}$; π_b is the subtree of π with the root b. Given a policy π, there is one-to-one correspondence between scenarios and simulation sequences. So, $|\mathcal{Z}'| = K$. We optimize $\nu_\pi(b_0)$ over $\Pi_\mathcal{T}$ by performing a tree search on $\mathcal{T}$ and applying Bellman's equation recursively at every internal node of $\mathcal{T}$, similar to (3):

$$\nu^*(b) = \max \left\{ \frac{\gamma^{\Delta(b)}}{K} \sum_{\boldsymbol{\zeta} \in \mathcal{Z}'_b} w(\boldsymbol{\zeta}_{0:\Delta(b)}) V_{\pi_0, s_{\boldsymbol{\zeta}, \Delta(b)}}, \max_{a \in A} \left\{ \rho(b, a) + \sum_{z \in Z_{b,a}} \nu^*(\tau(b, a, z)) \right\} \right\} \tag{10}$$

where

$$\rho(b, a) = \frac{1}{K} \sum_{\boldsymbol{\zeta} \in \mathcal{Z}'_b} \gamma^{\Delta(b)} w(\boldsymbol{\zeta}_{0:\Delta(b)}) R(s_{\boldsymbol{\zeta}, \Delta(b)}, a) - \lambda.$$

In (10), π_0 denotes a given default policy; $s_{\boldsymbol{\zeta}, \Delta(b)}$ denotes the state in $\boldsymbol{\zeta}$ at time step $\Delta(b)$; $V_{\pi_0, s}$ is the value of π_0 starting from a state s. At each node b, we may choose to follow a default policy π_0 or one of the action branches. The out maximization in (10) chooses between these two options, while the inner maximization chooses the specific action branch. The maximizer at b_0, the root of $\mathcal{T}$, gives the optimal action.

There are many tree search algorithms. One is to traverse $\mathcal{T}$ from the bottom up. At each leaf node b of $\mathcal{T}$, the algorithm sets $\nu^*(b)$ as the value of the default policy π_0

and then applies (10) at each internal node until reaching the root of $\mathcal{T}$. The bottom-up traversal is conceptually simple, but $\mathcal{T}$ must be constructed fully in advance. For very large POMDPs, the required number of scenarios, K, may be huge, and constructing the full DESPOT is not practical.

To scale up, an alternative is to perform anytime heuristic search. To guide the heuristic search, the algorithm maintains at each node b of $\mathcal{T}$ a lower bound and an upper bound on $\nu^*(b)$. It constructs and searches $\mathcal{T}$ incrementally, using K sampled scenarios. Initially, $\mathcal{T}$ contains only a single root node with belief b_0. The algorithm makes a series of explorations to expand $\mathcal{T}$ and reduces the gap between the upper and lower bounds at the root node b_0 of $\mathcal{T}$. Each exploration follows the heuristic and traverses a promising path from b_0 to add new nodes at the end of the path. The algorithm then traces the path back to b_0 and applies (10) to both the lower and upper bounds at each node along the way. The explorations continue, until the gap between the upper and lower bounds reaches a target level or the allocated online planning time runs out.

Online planning for IS-DESPOT is very similar to that for DESPOT. We refer the reader to [26] for details.

3.4 Analysis

We now show that IS-DESPOT retains the theoretical guarantee of DESPOT. The two theorems below generalize the earlier results [22] to the case of importance sampling. To simplify the presentation, this analysis assumes, without loss of generality, $R(s,a) \in [0, R_{\max}]$ for all states and actions. All proofs are available in the appendix.

Theorem 1 shows that with high probability, importance sampling produces an accurate estimate of the value of a policy.

Theorem 1. *Let b_0 be a given belief. Let $\Pi_{\mathcal{T}}$ be the set of all IS-DESPOT policies derived from a DESPOT $\mathcal{T}$ and $\Pi_{b_0,D,K} = \bigcup_{\mathcal{T}} \Pi_{\mathcal{T}}$ be the union over all DESPOTs with root node b_0, with height D, and constructed with all possible K importance-sampled scenarios. For any $\tau, \alpha \in (0,1)$, every IS-DESPOT policy $\pi \in \Pi_{b_0,D,K}$ satisfies*

$$V_\pi(b_0) \geq \frac{1-\alpha}{1+\alpha}\hat{V}_\pi(b_0) - \frac{R_{\max}W_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{\ln(4/\tau) + |\pi| \ln(KD|A||Z|)}{\alpha K} \tag{11}$$

with probability at least $1-\tau$, where

$$W_{\max} = \left\{ \max_{\substack{s,s' \in S \\ a \in A, z \in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^D$$

is the maximum importance weight.

The estimation error bound in (11) holds for all policies in $\Pi_{b_0,D,K}$ simultaneously. It also holds for any constant $\alpha \in (0,1)$, which is a parameter that can be tuned to tighten the bound. The additive error on the RHS of (11) depends on the size of policy π. It also grows logarithmically with $|A|$ and $|Z|$, indicating that IS-DESPOT scales up well for POMDP with very large action and observation spaces.

Theorem 2 shows that we can find a near-optimal policy $\hat{\pi}$ by maximizing the RHS of (11).

Theorem 2. *Let π^* be an optimal policy at a belief b_0. Let $\Pi_{\mathcal{T}}$ be the set of policies derived from a DESPOT $\mathcal{T}$ that has height D and is constructed with K importance-sampled scenarios for b_0. For any $\tau, \alpha \in (0,1)$, if*

$$\hat{\pi} = \arg\max_{\pi \in \Pi_{\mathcal{T}}} \left\{ \frac{1-\alpha}{1+\alpha} \hat{V}_\pi(b_0) - \frac{R_{\max} W_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{|\pi| \ln(KD|A||Z|)}{\alpha K} \right\},$$

then with probability at least $1-\tau$,

$$V_{\hat{\pi}}(b_0) \geq \frac{1-\alpha}{1+\alpha} V_{\pi^*}(b_0) - \frac{R_{\max} W_{\max}}{(1+\alpha)(1-\gamma)} \times \left(\frac{\ln(8/\tau) + |\pi^*| \ln(KD|A||Z|)}{\alpha K} + (1-\alpha) \left(\sqrt{\frac{2\ln(2/\tau)}{K}} + \gamma^D \right) \right). \tag{12}$$

The estimation errors in both theorems depend on the choice of the importance distribution. By setting the importance sampling distribution to the natural probability of occurrence, we recover exactly the same results for the original DESPOT algorithm [22].

3.5 Normalized Importance Sampling

As we discussed in Section 2.2, normalized importance sampling often reduces the variance of an estimator in practice. The algorithm remains basically the same, other than normalizing the importance weights. The analysis is also similar, but is more involved. Our experiments show that IS-DESPOT with normalized importance weights produces better performance in some cases (see Section 5).

4 Importance Distributions

We now derive the optimal importance distribution for IS-DESPOT and then present a general method for learning it. We focus on the importance distribution for sampling state transitions and leave that for sampling observations as future work.

4.1 Optimal Importance Distribution

The value of a policy π at a belief b, $V_\pi(b)$, is the expected total discounted reward of executing π, starting at a state s distributed according to b. It can be obtained by integrating over all possible starting states:

$$V_\pi(b) = \int_{s \in S} \mathbb{E}(v|s,\pi) b(s) \, \mathrm{d}s,$$

where v is a random variable representing the total discounted reward of executing π starting from s and $\mathbb{E}(v|s,\pi)$ is its expectation. Compared with standard importance sampling (Section 2.2), IS-DESPOT estimates $f(s) = \mathbb{E}(v|s,\pi)$ by Monte Carlo simulation of π rather than evaluates $f(s)$ deterministically. Thus the importance distribution

must account for not only the mean $\mathbb{E}(v|s,\pi)$ but also the variance $\mathrm{Var}(v|s,\pi)$ resulting from Monte Carlo Sampling, and give a state s increased importance when either is large. The theorem below formalizes this idea.

Theorem 3. *Given a policy π, let v be a random variable representing the total discounted reward of executing π, starting from state s, and let $V_\pi(b)$ be the expected total discounted reward of π at a belief b. To estimate $V_\pi(b)$ using unnormalized importance sampling, the optimal importance distribution is $q^*_\pi(s) = b(s)/w_\pi(s)$, where*

$$w_\pi(s) = \frac{\mathbb{E}_b\Big(\sqrt{[\mathbb{E}(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)}\Big)}{\sqrt{[\mathbb{E}(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)}}. \tag{13}$$

Theorem 3 specifies the optimal importance distribution for a given policy π, but IS-DESPOT must evaluate many policies within a set $\Pi_{\mathcal{T}}$, for some DESPOT $\mathcal{T}$. Our next result suggests that we can use the optimal importance distribution for a policy π to estimate the value of another "similar" policy π'. This allows us to use a single importance distribution for IS-DESPOT.

Theorem 4. *Let $\hat{V}_{\pi,q}(b)$ be the estimated value of a policy π at a belief b, obtained by K independent samples with importance distribution q. Let v be the total discounted reward of executing a policy π, starting from a state $s \in S$. If two policies π and π' satisfy $\frac{\mathrm{Var}(v|s,\pi')}{\mathrm{Var}(v|s,\pi)} \leq 1+\epsilon$ and $\frac{[\mathbb{E}(v|s,\pi')]^2}{[\mathbb{E}(v|s,\pi)]^2} \leq 1+\epsilon$ for all $s \in S$ and some $\epsilon > 0$, then*

$$\mathrm{Var}\big(\hat{V}_{\pi',q^*_\pi}(b)\big) \leq (1+\epsilon)\Big(\mathrm{Var}(\hat{V}_{\pi,q^*_\pi}(b)) + \frac{1}{K}V^*(b)^2\Big), \tag{14}$$

where q^*_π is an optimal importance distribution for estimating the value of π and $V^*(b)$ is the value of an optimal policy at b.

4.2 Learning Importance Distributions

According to Theorems 3 and 4, we can construct the importance distribution for IS-DESPOT, if we know the mean $\mathbb{E}(v|s,\pi)$ and the variance $\mathrm{Var}(v|s,\pi)$ for all $s \in S$ under a suitable policy $\pi \in \Pi_{\mathcal{T}}$. Two issues remain. First, we need to identify π. Second, we need to represent $\mathbb{E}(v|s,\pi)$ and $\mathrm{Var}(v|s,\pi)$ compactly, as the state space S may be very large or even continuous. For the first issue, we use the policy generated by the DESPOT algorithm, without importance sampling. Theorem 4 helps to justify this choice. For the second issue, we use offline learning, based on discrete feature mapping. Specifically, we learn the function

$$\xi(s) = \sqrt{[\mathbb{E}(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)}, \tag{15}$$

which is proportional to the inverse importance weight $1/w_\pi(s)$. We then set the importance distribution to $q(s) = \eta p(s)\,\xi(s)$, where η is the normalization constant.

To learn $\xi(s)$, we first generate data by running DESPOT many times offline in simulation without importance sampling. Each run starts at a state s_0 sampled from the initial belief b_0 and generates a sequence $\{s_0, a_1, r_1, s_1, a_2, r_2, s_2, \ldots, a_D, r_D, s_D\}$,

where a_t is the action that DESPOT chooses, s_t is a state sampled from the model $p(s_t|s_{t-1}, a_t) = T(s_{t-1}, a_t, s_t)$, and $r_t = R(s_{t-1}, a_t)$ is the reward at time t for $t = 1, 2, \ldots$. Let $v_t = \sum_{i=t}^{D} \gamma^{(i-t)} r_i$ be the total discounted reward from time t. We collect all pairs (s_{t-1}, v_t) for $t = 1, 2, \ldots$ from each run. Next, we manually construct a set of features over the state space S so that each state $s \in S$ maps to its feature value $\mathcal{F}(s)$. Finally, We discretize the feature values into a finite set of bins of equal size and insert each collected data pair (s, v) into the bin that contains $\mathcal{F}(s)$. We calculate the mean $\mathbb{E}(v)$ and the variance $\mathrm{Var}(v)$ for each bin. For any state $s \in S$, $\xi(s)$ can then be approximated from the mean and the variance of the bin that contains $\mathcal{F}(s)$.

This learning method scales up to high-dimensional and continuous state spaces, provided a small feature set can be constructed. It introduces approximation error, as a result of discretization. However, the importance distribution is a heuristic that guides Monte Carlo sampling, and IS-DESPOT is overall robust against approximation error in the importance distribution. The experimental results in the next section show that the importance distribution captured by a small feature set effectively improves online planning for large POMDPs.

5 Experiments

We evaluated IS-DESPOT on a suite of five tasks (Table 1). The first three tasks are known to contain *critical* states, states which are not encountered frequently, but may lead to significant consequences. The other two, RockSample [21] and Pocman [19], are established benchmark tests for evaluating the scalability of POMDP algorithms.

We compared two versions of IS-DESPOT, unnormalized and normalized, with both DESPOT and POMCP. See Table 1 for the results. For all algorithms, we set the maximum online planning time to one second per step. For DESPOT and IS-DESPOT, we set the number of sampled scenarios to 500 and used an uninformative upper bound and fixed-action lower bound policy without domain-specific knowledge [26] in AsymmetricTiger, CollisionAvoidance, and Demining. Following earlier work [26], we set the number of sampled scenarios to 500 and 100, respectively, in RockSample and Pocman, and used domain-specific heuristics. For fair comparison, we tuned the exploration constant of POMCP to the best possible and used a rollout policy with the same domain knowledge as that for DESPOT and IS-DESPOT in each task.

Overall, Table 1 shows that both unnormalized and normalized IS-DESPOT substantially outperform DESPOT and POMCP in most tasks, including, in particular, the two large-scale tasks, Demining and Pocman. Normalized IS-DESPOT performs better than unnormalized IS-DESPOT in some tasks, though not all.

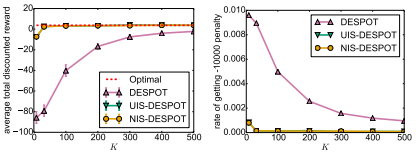
AsymmetricTiger Tiger is a classic POMDP [9]. A tiger hides behind one of two doors, denoted by states s_L and s_R, with equal probabilities. An agent must decide which door to open, receiving a reward of $+10$ for opening the door without the tiger and receiving -100 otherwise. At each step, the agent may choose to open a door or to listen. Listening incurs a cost of -1 and provides the correct information with probability 0.85. The task resets once the door is opened. The only uncertainty here is the tiger's location. Tiger is a toy problem, but it captures the key trade-off between information

Table 1: Performance comparison. The table shows the average total discounted rewards (95% confidence interval) of four algorithms on five tasks. UIS-DESPOT and NIS-DESPOT refer to unnormalized and normalized IS-DESPOT, respectively.

	AsymmetricTiger	*CollisionAvoidance*	*Demining*	*RockSample(15,15)*	*Pocman*
$\|S\|$	2	9,720	$\sim 10^{49}$	7,372,800	$\sim 10^{56}$
$\|A\|$	3	3	9	20	4
$\|Z\|$	2	18	16	3	1,024
POMCP	-5.60 ± 1.51	-4.19 ± 0.07	-17.11 ± 2.74	15.32 ± 0.28	294.16 ± 4.06
DESPOT	-2.20 ± 1.78	-1.05 ± 0.27	-11.09 ± 2.45	18.37 ± 0.28	317.90 ± 4.17
UIS-DESPOT	3.70 ± 0.49	-0.87 ± 0.19	-3.45 ± 1.76	18.30 ± 0.32	315.13 ± 4.92
NIS-DESPOT	3.75 ± 0.47	-0.44 ± 0.12	-3.69 ± 1.80	18.68 ± 0.28	326.92 ± 3.89

state	$\mathbb{E}(v\|s)$	$\mathrm{Var}(v\|s)$	$p(s)$	$q(s)$
s_L	4.36	7.34	0.99	0.40
s_R	-65.84	566978.9	0.01	0.60

(*a*)



(*b*) (*c*)

Fig. 2: AsymmetricTiger. (*a*) Importance distribution q. DESPOT samples according to p. IS-DESPOT samples according to q. (*b*) Average total discounted reward versus K, the number of sampled scenarios. (*c*) The rate of opening the door with the tiger behind versus K.

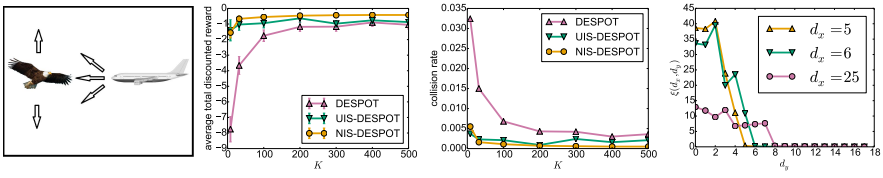
gathering and information exploitation prevalent in robot planning under uncertainty. We use it to gain understanding on some important properties of IS-DESPOT.

We first modify the original Tiger POMDP. Instead of hiding behind the two doors with equal probabilities, the tiger hides behind the right door with much smaller probability 0.01. However, if the agent opens the right door with the tiger hiding there, the penalty increases to $-10,000$. Thus the state s_R occurs rarely, but has a significant consequence. Sampling s_R is crucial.

Since this simple task only has two states, we learn the weight for each state and use them to construct the importance distribution for IS-DESPOT (Fig. 2*a*).

Table 1 shows clearly that both versions of IS-DESPOT significantly outperform DESPOT and POMCP. This is not surprising. DESPOT and POMCP sample the two states s_L and s_R according to their natural probabilities of occurrence and fail to account for their *significance*, in this case, the high penalty of s_R. As a result, they rarely sample s_R. In contrast, the importance distribution enables IS-DSPOT to sample s_R much more frequently (Fig. 2*a*). Unnormalized and normalized IS-DESPOT are similar in performance, as normalization has little effect on this small toy problem.

We conducted additional experiments to understand how K, the number of sampled scenarios, affects performance (Fig. 2*b*, *c*). To calibrate the performance, we ran an offline POMDP algorithm SARSOP [15] to compute an optimal policy. When K is small, the performance gap between DESPOT and IS-DESPOT is significant. As K increases, the gap narrows. When K is 32, both versions of IS-DESPOT are near-optimal, while



(*a*) (*b*) (*c*) (*d*)

Fig. 3: CollisionAvoidance. (*a*) An aircraft changes the direction to avoid collision. (*b*) Average total discounted reward versus K, the number of scenarios. (*c*) Collision rate versus K. (*d*) $\xi(d_x, d_y)$ learned from data.

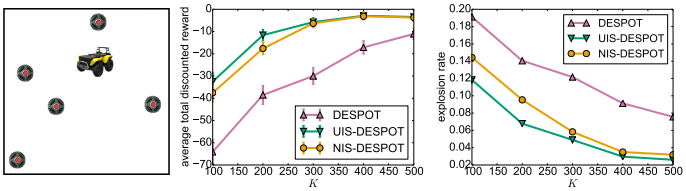
DESPOT does not reach a comparable performance level even at $K = 500$. All these confirm the value of importance sampling for small sample size.

One may ask how DESPOT and IS-DESPOT compare on the original Tiger task. The original Tiger has two symmetric states. The optimal importance distribution is flat. Thus DESPOT and IS-DESPOT have the same sampling distribution, and IS-DESPOT has no performance advantage over DESPOT. Importance sampling is beneficial only if the planning task involves significant events of low probability.

CollisionAvoidance This is an adaptation of the aircraft collision avoidance task [2]. The agent starts moving from a random position in the right-most column of a 18×30 grid map (Fig. 3*a*). An obstacle randomly moves in the left-most column of this map: moves up with probability 0.25, down with 0.25, and stay put with 0.50. The probabilities become 0, 0.25, and 0.75 respectively when the obstacle is at the top-most row, and become 0.25, 0, and 0.75 respectively when it is at the bottom-most row. Each time, the agent can choose to move upper-left, lower-left or left, with a cost of -1, -1, and 0 respectively. If the agent collides with the obstacle, it receives a penalty of $-1,000$. The task finishes when the agent reaches the left-most column. The agent knows its own position exactly, but observes the obstacle's position with a Gaussian noise $\mathcal{N}(0, 1)$. The probability of the agent colliding with the obstacle is small, but the penalty of collision is high.

To construct the importance distribution for IS-DESPOT, we map a state s to a feature vector (d_x, d_y) and learn $\xi(d_x, d_y)$. The features d_x and d_y are the horizontal and vertical distances between the obstacle and the agent respectively. The horizontal distance d_x is the number of steps remaining for the agent to move, and $(d_x, d_y) = (0, 0)$ represents a collision. These features capture the changes in the mean and variance of policy values for different states, because when the agent is closer to the obstacle, the probability of collision is higher, which leads to worse mean and larger variance.

Fig. 3*d* plots $\xi(d_x, d_y)$ for a few chosen horizontal distance d_x: $\xi(d_x = 5, d_y)$, $\xi(d_x = 6, d_y)$ and $\xi(d_x = 25, d_y)$. Overall, $\xi(d_x, d_y)$ is higher when the obstacle is closer to the agent. At $d_x = 5$, $\xi(d_x, d_y)$ is close to zero for all $d_y >= 4$, and $\xi(d_x, d_y) = 0$ when $d_y >= 10$. This indicates that the DESPOT policy rarely causes collisions when the vertical distance is larger than 4, because the collision would require the agent and obstacle move towards each other for several steps in the remaining $d_x = 5$ steps, which is unlikely to happen since the DESPOT policy is actively avoiding the collision. Furthermore, collision is not possible at all when $d_y >= 10$ because it



(*a*) (*b*) (*c*)

Fig. 4: Demining. (*a*) A robot moves in a 10×10 map to detect and report landmines. (*b*) Average total discounted reward versus K, the number of scenarios. (*c*) The rate of hitting mines versus K.

requires at least 6 steps for the agent and the obstacle to meet. With this importance distribution, IS-DESPOT can ignore the states that do not cause collision and focus on sampling states that likely lead to collisions. This could significantly improve the convergence rate, as shown in Fig. 3*b* and Fig. 3*c* : IS-DESPOT performs better than DESPOT with small K. NIS-DESPOT improves the performance further. It also performs well with small K, while DESPOT cannot reach the same level of performance even with a large K.

Demining This is an adapted version of the robotic de-mining task in Humanitarian Robotics and Automation Technology Challenge (HRATC) 2015 [27]. It requires an agent to sweep a 10×10 grid field to detect and report landmines. Each grid cell has a mine with probability 0.05. At each time step, the agent can move in four directions, and then observe the four adjacent cells with 0.9 accuracy. DESPOT is a core component of the system winning HRATC 2015. We show that IS-DESPOT handles critical states in the task better than DESPOT, resulting in better performance overall. If the agent reports the mine correctly, it receives reward $+10$, and -10 otherwise. Stepping over the mine causes high penalty $-1,000$ and task termination, which is a critical state.

To construct the importance distribution for IS-DESPOT, we map a state s to a feature vector (n, d) and learn $\xi(n, d)$, where n is the number of unreported mines and d is the Manhattan distance between the agent and the nearest mine. These two features capture the changes in the mean and variance of the rollout values for different states, because n determines the maximum total discounted rewards that the agent can get, and d affects the probability of stepping over mines.

Fig. 4*b* shows the performance comparison between IS-DESPOT and DESPOT for different number of scenarios K. IS-DESPOT converges faster than DESPOT, and significantly outperforms DESPOT even for relatively large number of scenarios ($K = 500$). Fig. 4*c* shows that the performance improvement results from a decrease in the number of explosions. For this task, the size of the state space is about 10^{49}, and IS-DESPOT clearly outperforms DESPOT and POMCP (Table 1), affirming the scalability of IS-DESPOT.

RockSample RockSample is a standard POMDP benchmark problem [21]. In the problem $RockSample(n, k)$, the agent moves on an $n \times n$ grid map which has k rocks. Each of the rocks can either be good or bad. The agent knows each rock's position but does not know its state (good or bad). At each time step, the agent can choose to observe the states, move, or sample. The agent can sample a rock it steps on, and get a reward of

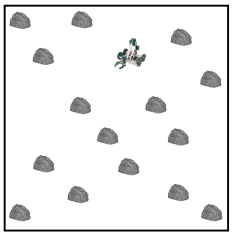


Fig. 5: RockSample. A robot senses rocks to identify "good" ones and samples them. Upon completion, it exits the east boundary.

$+10$ if the rock is good, -10 otherwise. A rock turns bad after being sampled. Moving or sensing has no cost. The agent can sense the state of the rock with accuracy decreasing exponentially with respect to the distance to the rock. When the agent exits from the east boundary of the map, it gets a reward of $+10$ and the task terminates.

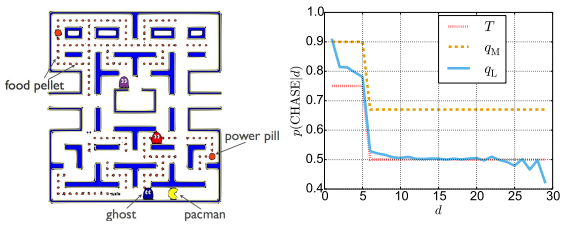
To construct the importance distribution for IS-DESPOT, we map a state s to a feature vector (n, d_+, d_-), where n is the number of remaining good rocks, d_+ and d_- are the Manhattan distances to the nearest good rock and the nearest bad rock respectively. This task does not contain critical states, nevertheless, IS-DESPOT maintains the same level of performance as DESPOT (Table 1).

Pocman Pocman [19] is a partially observable variant of the popular Pacman game (Fig. 6*a*). An agent and four ghosts move in a 17×19 maze populated with food pallets. The agent can move from a cell in the maze to an adjacent one if there is no wall in between. Each move incurs a cost of -1. A cell contains the food pallet with probability 0.5. Eating a food pallet gives the agent a reward of $+10$. Getting caught by ghosts incurs a penalty of -100. There are four power pills. After eating a power pill, the agents retains it for the next 15 steps and acquires the power to kill ghosts. Killing a ghost gives a reward of $+25$. Let d be the Manhattan distance between the agent and a ghost. When $d \leq 5$, the ghost chases the agent with probability 0.75 if the agent does not possess the power pill; the ghost runs away, but slips with probability 0.25 if the agent possesses the power pill. When $d > 5$, the ghost moves uniformly at random to feasible adjacent cells. In contrast with that in the original Pacman game, the Pocman agent does not know the exact locations of ghosts, but sees approaching ghosts when they are in a direct line of sight and hears them when $d \leq 2$.

To construct the importance distribution for IS-DESPOT, we map a state s to a feature vector $(n, d_{\min})$,where $d_{\min}$ is the Manhattan distance to the nearest ghost and n is the number of remaining steps for which the agent retains a power pill.

Table 1 shows that unnormalized IS-DESPOT does not bring in improvement over DESPOT. However, normalized IS-DESPOT does, because normalization of importance weight reduces variance in this case.

To understand the benefits of learning the importance distribution, we tried to construct an importance distribution q_{M} manually. The states, in which a ghost catches the agent, are critical. They do not occur very often, but incur high penalty. An effective importance distribution must sample such states with increased probability. One crucial aspect of the ghost behaviour is the decision of chasing the agent, governed by



(*a*) (*b*)

Fig. 6: Pocman. (*a*) The Pacman/Pocman game. (*b*) Compare the natural state-transition distribution T, the manually constructed importance distribution q_{M}, and the learned importance distribution q_{L}.

the distribution $p(\text{CHASE} \mid d)$. To obtain q_{M}, we shift $p(\text{CHASE} \mid d)$ upward by a constant amount, reasoning that having the ghost chase the agent more often increases the sampling of critical states in which a ghost catches an agent. Unfortunately q_{M}, when used in normalized IS-DESPOT, achieved an average total discounted reward of 317.02 ± 4.21, weaker than that of the learned importance distribution q_{L}. To understand why, let us compare q_{M} with q_{L} (Fig. 6*b*). Although q_{L} does have the ghost chase the agent more often as q_{M} does, it does not increase $p(\text{CHASE}|d)$ uniformly. The increase depends on the distance d. It is not realistic to handcraft such an importance distribution without the help of learning from data.

6 Conclusion

This paper introduces importance sampling to sampling-based online POMDP planning. Specifically, IS-DESPOT retains the theoretical guarantee of the original DESPOT algorithm, and it outperforms two state-of-the-art online POMDP algorithms on a test suite of five distinct tasks.

There are multiple directions to extend this work. First, our current method for learning the importance distribution focuses on critical states, but observations are equally important for planning under uncertainty. Extending IS-DESPOT to handle critical observations is straightforward. Second, we want to fully automate the importance distribution construction through feature learning. Finally, the idea of importance sampling is general and can be applied to other MDP and POMDP planning algorithms (e.g., [3,11,19]).

Acknowledgments. This work is supported in part by NUS AcRF Tier 1 grant R-252-000-587-112, A*STAR grant R-252-506-001-305, and US Air Force Research Laboratory under agreement number FA2386-15-1-4010.

References

1. Bai, H., Cai, S., Ye, N., Hsu, D., Lee, W.S.: Intention-aware online POMDP planning for autonomous driving in a crowd. In: Proc. IEEE Int. Conf. on Robotics & Automation (2015)
2. Bai, H., Hsu, D., Kochenderfer, M.J., Lee, W.S.: Unmanned aircraft collision avoidance using continuous-state POMDPs. Robotics: Science and Systems VII 1 (2012)

3. Bai, H., Hsu, D., Lee, W.S., Ngo, V.A.: Monte Carlo value iteration for continuous-state POMDPs. In: Algorithmic Foundations of Robotics IX (2010)
4. Folsom-Kovarik, J., Sukthankar, G., Schatz, S.: Tractable POMDP representations for intelligent tutoring systems. ACM Trans. on Intelligent Systems & Technology 4(2) (2013)
5. Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: Proc. Int. Conf. on Machine Learning (2007)
6. Glasserman, P.: Monte Carlo methods in financial engineering, vol. 53. Springer Science & Business Media (2003)
7. He, R., Brunskill, E., Roy, N.: Efficient planning under uncertainty with macro-actions. J. Artificial Intelligence Research 40(1) (2011)
8. Hsiao, K., Kaelbling, L.P., Lozano-Perez, T.: Grasping POMDPs. In: Proc. IEEE Int. Conf. on Robotics & Automation (2007)
9. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1) (1998)
10. Kalos, M., Whitlock, P.: Monte Carlo Methods, vol. 1. John Wiley & Sons, New York (1986)
11. Kearns, M., Mansour, Y., Ng, A.Y.: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Machine Learning 49(2-3) (2002)
12. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
13. Koval, M., Pollard, N., Srinivasa, S.: Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. Int. J. Robotics Research 35(1–3) (2016)
14. Koval, M., Hsu, D., Pollard, N., Srinivasa, S.: Configuration lattices for planar contact manipulation under uncertainty. In: Algorithmic Foundations of Robotics XII—Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR) (2016)
15. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proc. Robotics: Science & Systems (2008)
16. Owen, A.B.: Monte Carlo theory, methods and examples (2013)
17. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proc. Int. Jnt. Conf. on Artificial Intelligence (2003)
18. Ross, S., Pineau, J., Paquet, S., Chaib-Draa, B.: Online planning algorithms for POMDPs. J. Artificial Intelligence Research 32 (2008)
19. Silver, D., Veness, J.: Monte-Carlo planning in large POMDPs. In: Advances in Neural Information Processing Systems (NIPS) (2010)
20. Smallwood, R., Sondik, E.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research 21 (1973)
21. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proc. Uncertainty in Artificial Intelligence (2004)
22. Somani, A., Ye, N., Hsu, D., Lee, W.S.: DESPOT: Online POMDP planning with regularization. In: Advances in Neural Information Processing Systems (NIPS) (2013)
23. Spaan, M., Vlassis, N.: Perseus: Randomized point-based value iteration for POMDPs. J. Artificial Intelligence Research 24 (2005)
24. Veach, E.: Robust Monte Carlo methods for light transport simulation. Ph.D. thesis, Stanford University (1997)
25. Wu, K., Lee, W.S., Hsu, D.: POMDP to the rescue: Boosting performance for RoboCup rescue. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (2015)
26. Ye, N., Somani, A., Hsu, D., Lee, W.S.: DESPOT: Online POMDP planning with regularization. J. Artificial Intelligence Research (to appear)
27. Humanitarian robotics and automation technology challenge (HRATC) 2015, http://www.isr.uc.pt/HRATC2015

A Proofs

A.1 Theorems 1 and 2

To prove Theorem 1, we rely on an earlier result [22]:

Theorem 5. *For any $\tau, \alpha \in (0,1)$, every policy tree $\pi \in \Pi_{b_0,D,K}$ satisfies*

$$V_\pi(b_0) \geq \frac{1-\alpha}{1+\alpha}\hat{V}_\pi(b_0) - \frac{R_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{\ln(4/\tau) + |\pi| \ln(KD|A||Z|)}{\alpha K},$$

with probability at least $1-\tau$, where $\hat{V}_\pi(b_0)$ is the estimated value of π under any set of K randomly sampled scenarios for belief b_0.

The theorem above is similar to Theorem 1, but provides the performance guarantee for the original DESPOT algorithm, which does not employ importance sampling. There are two key differences. When computing $\hat{V}_\pi(b_0)$ without importance sampling, we use the natural state-transition and observation probability distribution $p(s_{t+1}, z_{t+1}|s_t, a_{t+1})$ to generate the next state and observation in a simulation sequence $\boldsymbol{\zeta}$; with importance sampling, we use the importance distribution $q(s_{t+1}, z_{t+1}|s_t, a_{t+1})$. Further, without importance sampling, we use the reward $R(s_t, a_{t+1})$ from the POMDP model; with importance sampling, we use the weighted reward $w(\boldsymbol{\zeta}_{0:t})R(s_t, a_{t+1})$, where $\boldsymbol{\zeta}$ is a simulation sequence that leads to s_t and $w(\boldsymbol{\zeta}_{0:t})$ is the weight of the sub-sequence of $\boldsymbol{\zeta}$ over the time steps $0, 1, ..., t$. To prove Theorem 1, our general idea is to construct a new POMDP with modified dynamics and reward to account for importance sampling and then apply Theorem 5.

Proof. (*Theorem 1*) We construct a new POMDP with the state-transition and observation probability distribution $p'(s_{t+1}, z_{t+1}|s_t, a_{t+1}) = q(s_{t+1}, z_{t+1}|s_t, a_{t+1})$ and reward function $R'(s_t, a_{t+1}) = w(\boldsymbol{\zeta}_{0:t})R(s_t, a_{t+1})$. Running DESPOT algorithm on this new POMDP model and running IS-DESPOT on the original model generate exactly the same simulation sequences with same total rewards. It then follows from Theorem 5 that

$$V'_\pi(b_0) \geq \frac{1-\alpha}{1+\alpha}\hat{V}_\pi(b_0) - \frac{R'_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{\ln(4/\tau) + |\pi| \ln(KD|A||Z|)}{\alpha K}, \tag{16}$$

where $V'_\pi(b_0)$ is the value of π at b_0 under the new model and $R'_{\max} = \max_{s\in S, a\in A} R'(s,a)$. Note that

$$\max_{s\in S, a\in A} R'(s,a) = \max_{\substack{\boldsymbol{\zeta}_{0:t} \\ s_t\in S, a_{t+1}\in A}} w(\boldsymbol{\zeta}_{0:t})R(s_t, a_{t+1}) \leq \left\{ \max_{\substack{s,s'\in S \\ a\in A, z\in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^t R_{\max}.$$

Since $\mathbb{E}_{s,z\sim q(s,z|s',a)} \frac{p(s,z|s',a)}{q(s,z|s',a)} = 1$ for all $s' \in S$ and $a \in A$, we have

$$\max_{\substack{s,s'\in S \\ a\in A, z\in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \geq 1.$$

Hence

$$R'_{\max} \leq \left\{ \max_{\substack{s,s' \in S \\ a \in A, z \in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^{t} R_{\max} \leq \left\{ \max_{\substack{s,s' \in S \\ a \in A, z \in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^{D} R_{\max} = W_{\max} R_{\max}, \tag{17}$$

as $t \leq D$. Combining (16) and (17), we have

$$V'_{\pi}(b_0) \geq \frac{1-\alpha}{1+\alpha} \hat{V}_{\pi}(b_0) - \frac{R_{\max} W_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{\ln(4/\tau) + |\pi| \ln(KD|A||Z|)}{\alpha K}. \tag{18}$$

It remains to show that π has the same value under the new and the original POMDP models. By definition,

$$\begin{aligned} V'_{\pi}(b_0) &= \mathbb{E}_{\boldsymbol{\zeta} \sim q(\boldsymbol{\zeta}|b_0,\pi)} \left(\sum_{t=0}^{\infty} \gamma^t w(\boldsymbol{\zeta}_{0:t}) R(s_t, a_{t+1}) \right) \\ &= \int \sum_{t=0}^{\infty} \gamma^t w(\boldsymbol{\zeta}_{0:t}) R(s_t, a_{t+1}) q(\boldsymbol{\zeta}|b_0,\pi) \, \mathrm{d}\boldsymbol{\zeta} \\ &= \sum_{t=0}^{\infty} \int \frac{p(\boldsymbol{\zeta}_{0:t}|b_0,\pi)}{q(\boldsymbol{\zeta}_{0:t}|b_0,\pi)} \gamma^t R(s_t, a_{t+1}) q(\boldsymbol{\zeta}|b_0,\pi) \, \mathrm{d}\boldsymbol{\zeta}. \end{aligned}$$

Since the reward $R(s_t, a_{t+1})$ does not depend on the subsequence $\boldsymbol{\zeta}_{t+1:\infty}$, we marginalize it out and get

$$\begin{aligned} V'_{\pi}(b_0) &= \sum_{t=0}^{\infty} \int \frac{p(\boldsymbol{\zeta}_{0:t}|b_0,\pi)}{q(\boldsymbol{\zeta}_{0:t}|b_0,\pi)} \gamma^t R(s_t, a_{t+1}) q(\boldsymbol{\zeta}_{0:t}|b_0,\pi) \, \mathrm{d}\boldsymbol{\zeta}_{0:t} \\ &= \sum_{t=0}^{\infty} \int p(\boldsymbol{\zeta}_{0:t}|b_0,\pi) \gamma^t R(s_t, a_{t+1}) \, \mathrm{d}\boldsymbol{\zeta}_{0:t} \\ &= \int \sum_{t=0}^{\infty} \gamma^t R(s_t, a_{t+1}) p(\boldsymbol{\zeta}|b_0,\pi) \, \mathrm{d}\boldsymbol{\zeta} \\ &= V_{\pi}(b_0). \end{aligned} \tag{19}$$

The third line above again follows from marginalization over the subsequence $\boldsymbol{\zeta}_{t+1:\infty}$.

Finally, putting together (18) and (19), we get the desired result. □

To prove Theorem 2, we use Theorem 1 and follow the same steps as the proof of Theorem 2 in [22]. We omit the details here.

A.2 Theorem 3

Proof. Given a policy π, we want to estimate $V_\pi(b)$, the value of π at the belief b, with the importance distribution $q(s)$. The variance of the estimator $\frac{b(s)}{q(s)}v$ is

$$\begin{aligned}
\mathrm{Var}\left(\frac{b(s)}{q(s)}v\middle|\pi\right) &= \mathbb{E}\left(\frac{b(s)^2}{q(s)^2}v^2\middle|\pi\right) - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)^2}v^2 p(v|s,\pi)q(s)\,\mathrm{d}v\,\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)}\left(\int v^2 p(v|s,\pi)\,\mathrm{d}v\right)\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)}\left(\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)\right)\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2.
\end{aligned} \tag{20}$$

Substituting $q^*_\pi(s) = \frac{b(s)\sqrt{[\mathbb{E}(v|s,\pi)]^2+\mathrm{Var}(v|s,\pi)}}{\mathbb{E}_b\left(\sqrt{[\mathbb{E}(v|s,\pi)]^2+\mathrm{Var}(v|s,\pi)}\right)}$ into (20), we get

$$\begin{aligned}
\mathrm{Var}\left(\frac{b(s)}{q^*_\pi(s)}v\middle|\pi\right) &= \mathbb{E}_b\left(\sqrt{\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)}\right)\int \sqrt{\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)}\; b(s)\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q^*_\pi(s)}v\middle|\pi\right)\right]^2 \\
&= \left[\mathbb{E}_b\left(\sqrt{\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)}\right)\right]^2 - \left[\mathbb{E}\left(\frac{b(s)}{q^*_\pi(s)}v\middle|\pi\right)\right]^2
\end{aligned}$$

Unnormalized importance sampling is unbiased. Thus, for any arbitrary importance distribution $q(s)$,

$$\mathbb{E}\left(\frac{b(s)}{q^*_\pi(s)}v\middle|\pi\right) = \mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right),$$

and

$$\begin{aligned}
\mathrm{Var}\left(\frac{b(s)}{q^*_\pi(s)}v\middle|\pi\right) &= \left[\mathbb{E}_b\left(\sqrt{\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)}\right)\right]^2 - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \left[\mathbb{E}_q\left(\frac{b(s)}{q(s)}\sqrt{\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)}\right)\right]^2 - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&\le \mathbb{E}_q\left(\frac{b(s)^2}{q(s)^2}\left(\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)\right)\right) - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)}\left(\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)\right)\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \mathrm{Var}\left(\frac{b(s)}{q(s)}v\middle|\pi\right).
\end{aligned}$$

The third line above follows because $[\mathbb{E}(X)]^2 \le \mathbb{E}(X^2)$ for any random variable X by the Cauchy-Schwarz inequality, and the last line follows from (20). Therefore, the variance of the estimator with importance distribution $q^*_\pi(s)$ is no greater than that with any other importance distribution, and $q^*_\pi(s)$ is optimal. □

A.3 Theorem 4

Proof. By the definition of $\hat{V}_{\pi',q_\pi^*}(b)$,

$$\hat{V}_{\pi',q_\pi^*}(b) = \frac{1}{K}\sum_{i=1}^{K}\frac{b(s_i)}{q_\pi^*(s_i)}v_i, \quad v_i \sim p(v|s_i,\pi'), \quad s_i \sim q_\pi^*(s),$$

where $p(v|s_i,\pi')$ is the probability distribution over the value v of following policy π' from the starting state s_i. Since the states are sampled independently,

$$\mathrm{Var}\left(\hat{V}_{\pi',q_\pi^*}(b)\right) = \frac{1}{K}\mathrm{Var}_{q_\pi^*}\left(\frac{b(s)}{q_\pi^*(s)}v\middle|\pi'\right)$$

Similar to (20), we get

$$\mathrm{Var}\left(\frac{b(s)}{q_\pi^*(s)}v\middle|\pi'\right) = \int\frac{b(s)^2}{q_\pi^*(s)}\left(\left[\mathbb{E}(v|s,\pi')\right]^2 + \mathrm{Var}(v|s,\pi')\right)\mathrm{d}s - V_{\pi'}(b)^2$$

where $V_{\pi'}(b) = \mathbb{E}_{q_\pi^*}\left(\frac{b(s)}{q_\pi^*(s)}v\middle|\pi'\right)$ is the value of policy π' at b. Hence

$$\begin{aligned}\mathrm{Var}\left(\hat{V}_{\pi',q_\pi^*}(b)\right) &= \frac{1}{K}\left\{\int\frac{b(s)^2}{q_\pi^*(s)}\left(\left[\mathbb{E}(v|s,\pi')\right]^2 + \mathrm{Var}(v|s,\pi')\right)\mathrm{d}s - V_{\pi'}(b)^2\right\}\\ &\le \frac{1}{K}\left\{(1+\epsilon)\int\frac{b(s)^2}{q_\pi^*(s)}\left(\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)\right)\mathrm{d}s - V_{\pi'}(b)^2\right\}\end{aligned}$$

The second line above follows from the given conditions: $\frac{\mathrm{Var}(v|s,\pi')}{\mathrm{Var}(v|s,\pi)} \le 1+\epsilon$ and $\frac{\left[\mathbb{E}(v|s,\pi')\right]^2}{\left[\mathbb{E}(v|s,\pi)\right]^2} \le 1+\epsilon$ for all $s \in S$ and some $\epsilon > 0$. Let $V_\pi(b) = \mathbb{E}\left(\frac{b(s)}{q_\pi^*(s)}v\middle|\pi\right)$.

$$\mathrm{Var}\left(\frac{b(s)}{q_\pi^*(s)}v\middle|\pi'\right) \le \frac{1}{K}\left\{(1+\epsilon)\left(\int\frac{b(s)^2}{q_\pi^*(s)}\left(\left[\mathbb{E}\left(v|s,\pi\right)\right]^2 + \mathrm{Var}(v|s,\pi)\right)\mathrm{d}s - V_\pi(b)^2 + V_\pi(b)^2\right) - V_{\pi'}(b)^2\right\}$$

Similar to (20), we have $\int\frac{b(s)^2}{q_\pi^*(s)}\left(\left[\mathbb{E}[v|s,\pi]^2 + \mathrm{Var}(v|s,\pi)\right)\right)\mathrm{d}s - V_\pi(b)^2 = \mathrm{Var}\left(\frac{b(s)}{q_\pi^*(s)}v|\pi\right)$. Hence

$$\begin{aligned}\mathrm{Var}\left(\frac{b(s)}{q_\pi^*(s)}v\middle|\pi'\right) &\le \frac{1+\epsilon}{K}\mathrm{Var}_{q_\pi^*}\left(\frac{b(s)}{q_\pi^*(s)}v|\pi\right) + \frac{1}{K}\left((1+\epsilon)V_\pi(b)^2 - V_{\pi'}(b)^2\right)\\ &= (1+\epsilon)\mathrm{Var}\left(\hat{V}_{\pi,q_\pi^*}(b)\right) + \frac{1}{K}\left((1+\epsilon)V_\pi(b)^2 - V_{\pi'}(b)^2\right)\\ &\le (1+\epsilon)\mathrm{Var}\left(\hat{V}_{\pi,q_\pi^*}(b)\right) + \frac{1+\epsilon}{K}V^*(b)^2\\ &= (1+\epsilon)\left(\mathrm{Var}(\hat{V}_{\pi,q_\pi^*}(b)) + \frac{1}{K}V^*(b)^2\right).\end{aligned}$$

□