Article

The International Journal of Robotics Research
2018, Vol. 38(2-3) 162–181
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364918780322
journals.sagepub.com/home/ijr
SAGE

Importance sampling for online planning under uncertainty

Yuanfu Luo[1], Haoyu Bai[2*], David Hsu[1] and Wee Sun Lee[1]

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, 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 DESPOT's 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 demonstrate empirically that importance sampling significantly improves the performance of online POMDP planning for suitable tasks. We also present a general method for learning the importance sampling distribution.*

Keywords
Planning under uncertainty, POMDP, importance sampling

1. Introduction

Uncertainties in robot control and sensing, present significant barriers to reliable robot operation. The *partially observable Markov decision process* (POMDP) (Smallwood and Sondik, 1973) provides a principled general framework for robot decision making and planning under uncertainty. While POMDP planning is computationally intractable in the worst case (Papadimitriou and Tsisiklis, 1987), approximate POMDP planning algorithms have scaled up to a wide range of challenging tasks in robotics and beyond, e.g. autonomous driving (Bai et al., 2015), grasping (Hsiao et al., 2007), manipulation (Koval et al., 2016b), disaster rescue management (Wu et al., 2015), and intelligent tutoring systems (Folsom-Kovarik et al., 2013). Many of these recent advances leverage probabilistic sampling for computational efficiency. This work investigates effective sampling distributions for planning under uncertainty under the POMDP framework.

Intuitively, planning under uncertainty is difficult, because of the myriad future scenarios that could all affect the optimality of a plan. To overcome the resulting computational complexity, a very general idea is to sample a finite set of scenarios as an approximate representation of uncertainty and compute an optimal or near-optimal plan under these sampled scenarios. Theoretical analysis reveals that a small number of sampled scenarios guarantees near-optimal online planning, provided that there exists an optimal plan with a compact representation (Somani et al., 2013). In practice, the sampling distribution plays a critical role in probabilistic sampling algorithms and has significant impact on performance (Kalos and Whitlock, 1986). Consider an autonomous vehicle navigating among pedestrians. Hitting a pedestrian may have low probability, but very severe consequences. Failing to sample these *rare*, but *critical* events often results in suboptimal plans.

Importance sampling (Kalos and Whitlock, 1986) provides a well-established tool to address this challenge. We have developed IS-DESPOT (see Section 3), which applies importance sampling to *determinized sparse partially observable tree* (DESPOT) (Somani et al., 2013), a state-of-the-art sampling-based online POMDP algorithm. Like DESPOT, IS-DESPOT searches for a near-optimal plan at every time step under a set of sampled scenarios. However, it samples the scenarios according to their 'importance' rather than their natural probability of occurrence.

[1]Department of Computer Science, National University of Singapore, Singapore
[2]Movel AI Pte Ltd, Singapore

Corresponding author:
Yuanfu Luo, Department of Computer Science, National University of Singapore, 117417 Singapore.
Email: yuanfu@comp.nus.edu.sg
*This work was completed while the author was with the Department of Computer Science, National University of Singapore.

It then reweights the samples when computing the plan. While the basic idea of importance sampling is well known, the challenge here is to integrate it with DESPOT, which performs a lookahead search over a set of sampled scenarios. Each sample in DESPOT or IS-DESPOT is not a single event, but a sequence of interconnected events over time. Further, the theory of importance sampling is well developed for Monte Carlo integration with independent identically distributed samples. The samples used in IS-DESPOT are not independent. Nevertheless, we prove that IS-DESPOT retains the theoretical guarantee of DESPOT. We also present experimental results showing that importance sampling significantly improves the performance of online POMDP planning for suitable tasks (see Sections 5 and 6).

A crucial element of IS-DESPOT is the importance sampling distribution. Constructing it manually is not always easy and requires detailed domain knowledge. We take a first step towards automating the importance distribution construction and present a general method for learning it offline (see Section 4).

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 in state $s \in S$ 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

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

where η is a normalizing constant.

A POMDP *policy* prescribes the action at a belief. 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

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

where s_t is the state at time t, $a_{t+1} = \pi(b_t)$ is the action that the policy π chooses at time t, 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* (Figure 1(a)). Each node of a belief tree corresponds to a belief b. At each node, the tree branches on all actions in A and all observations in Z. If a node with belief b has a child node with belief b', then $b' = \tau(b,a,z)$. Conceptually, we may think of POMDP planning as a tree search in the *belief space*, the space of all possible beliefs that the agent may encounter. To find an optimal plan for a POMDP, we traverse the belief tree from the bottom up and compute an optimal action recursively at each node using Bellman's equation

$$\begin{aligned} V^*(b) = \max_{a \in A} \Bigg\{ & \sum_{s \in S} b(s) R(s,a) \\ & + \gamma \sum_{z \in Z} p(z|b,a) \, V^*\big(\tau(b,a,z)\big) \Bigg\} \end{aligned} \qquad (3)$$

POMDP planning is a special case of belief space planning. Belief planning is more general and does not require the planning model to satisfy the mathematical structure of POMDPs. For example, the reward function R may depend on the belief and not just on the state and the action.

2.2. Importance sampling

Suppose that we want to calculate the expectation

$$\mu = \mathbb{E}_p\big(f(s)\big) = \int_0^1 f(s) p(s) \, \mathrm{d}s$$

for a random variable s distributed according to p, but the function $f(s)$ is not efficiently integrable. One idea is to estimate μ with a set of samples $s_i, i = 1, 2, \ldots, n$

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} f(s_i), \quad s_i \sim p$$

The estimator $\hat{\mu}$ is unbiased, with variance $\mathrm{Var}(\hat{\mu}) = \sigma^2/n$, where $\sigma^2 = \int_0^1 (f(s) - \mu)^2 p(s) \, \mathrm{d}s$.

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

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

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 $\hat{\mu}_{\mathrm{UIS}}$ is also unbiased, with variance $\mathrm{Var}(\hat{\mu}_{\mathrm{UIS}}) = \sigma^2_{\mathrm{UIS}}/n$, where $\sigma^2_{\mathrm{UIS}} = \int_0^1 (\frac{f(s)p(s)}{q(s)} - \mu)^2 q(s) \, \mathrm{d}s$. Clearly the estimator's variance depends on the choice of q. The optimal importance distribution

$$q^*(s) = \frac{|f(s)| p(s)}{\mathbb{E}_p\big(|f(s)|\big)}$$

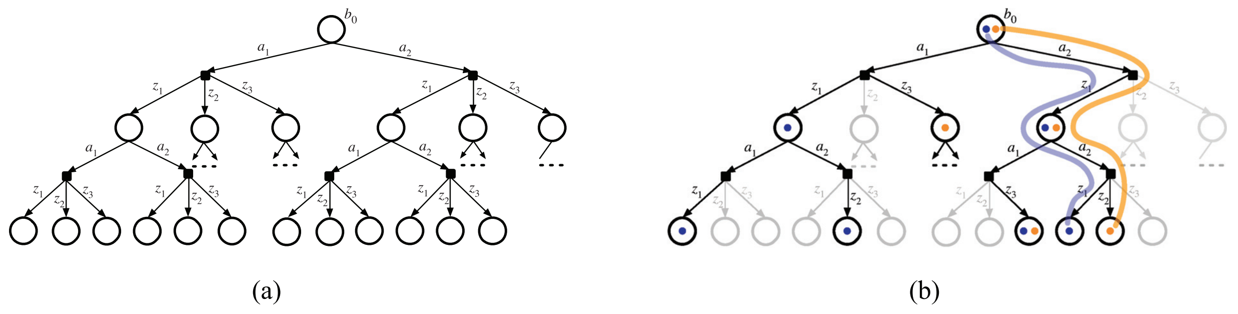


Fig. 1. Online POMDP planning performs lookahead search on a tree. (a) A standard belief tree of height $D = 2$. Each belief tree node represents a belief. At each node, the tree branches on every action and observation. (b) A DESPOT (black), obtained under two sampled scenarios marked by blue and orange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

gives the lowest variance (Owen, 2013).

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 (Owen, 2013). 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 decrease in variance often outweighs the increase in bias, leading to reduced estimation error overall (Koller and Friedman, 2009).

2.3. Related work

Uncertainties may arise from imperfect robot control, noisy sensors, unknown or changing environments, etc. They pose major challenges for robot planning. To plan effectively under uncertainty, a robot must gather information that reduces uncertainty and exploit the information to achieve specified task objectives. Occasionally it is possible to compute optimal or near-optimal plans without explicitly representing uncertainties, e.g. the bug algorithms (Lumelsky and Stepanov, 1987). More often, explicit representation of uncertainties enables much more effective planning. Early work represented uncertainties as *sets* of possibilities (Goldberg, 1993; Lozano-Pérez et al., 1984). Recent work favors more powerful probabilistic models, such as stochastic optimal control (Bertsekas, 2005) or POMDPs, which capture uncertainties in probability distributions. Both stochastic optimal control and the POMDP, plan in the belief space, the space of probability distributions. While the two were born in different research communities, they are closely related. As uncertainty models, they differ mainly in the continuity assumption. Stochastic optimal control assumes continuous states, actions, and observations. The POMDP often, though not exclusively, assumes discrete states, actions, and observations. For the discussion here, we refer to both as *belief-space planning* and do not differentiate between them.

A special case of belief-space planning is *linear-quadratic Gaussian* (LQG) control, which assumes a linear system, a quadratic cost function, and Gaussian noise. LQG control admits a closed-form optimal solution (Bertsekas, 2005). However, belief-space planning is much more challenging in its general form. Some methods aim for locally optimal solutions (Hauser, 2011; van den Berg et al., 2011, 2012). POMDP planning usually aims for globally optimal solutions.

There are two general approaches to POMDP planning: *offline* and *online*. 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 (Kurniawati et al., 2008; Pineau et al., 2003; Smith and Simmons, 2004; Spaan and Vlassis, 2005), they are inherently limited in scalability, because the number of possible future events grows exponentially with the planning horizon. In contrast, the online approach (Ross et al., 2008) 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 (Gelly and Silver, 2007; He et al., 2011).

Our work focuses on online planning. Partially Observable Monte-Carlo Planning (POMCP) (Silver and Veness, 2010) and DESPOT (Somani et al., 2013) are among the fastest online POMDP algorithms available today. DESPOT has found applications in many different robotic tasks, including autonomous driving in a crowd (Bai et al., 2015), robot demining in Humanitarian Robotics and Automation Technology Challenge 2015 (Madhavan et al., 2015), and push manipulation (Koval et al., 2016a). One key idea underlying both POMCP and DESPOT is the use of Monte Carlo simulation to sample future events and evaluate the quality of candidate policies. A similar idea has been used in offline POMDP planning (Bai et al., 2010). It is, however, well known that standard Monte Carlo sampling may miss rare, but critical events, resulting in suboptimal policies. Importance sampling is one way to alleviate this difficulty and improve planning performance.

Importance sampling is a well-established probabilistic sampling technique and has applications in many fields, e.g. Monte Carlo integration (Kalos and Whitlock, 1986), ray tracing for computer graphic rendering (Veach, 1997), and option pricing in finance (Glasserman, 2003). In the more closely related field of reinforcement learning, importance sampling has been used to estimate the value of a policy in a partially observable setting (Shelton, 2001). The focus of the work there is to derive the best estimator given an importance distribution. Within robotics, many methods for sampling-based motion planning make use of importance sampling (Boor et al., 1999; Kurniawati and Hsu, 2004; van den Berg and Overmars, 2005; Wilmarth et al., 1999). This work introduces importance sampling to robot planning under uncertainty and develops tools for obtaining good importance sampling distributions.

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 (Figure 1(a)) and applies equation (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 the Bayesian filtering equation (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 poses a major challenge for online planning when a POMDP has a large action space, large observation space, or long planning horizon.

A DESPOT is a sparse approximation of the belief tree, under K sampled scenarios (Figure 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 $\mathcal{O}(|A|^DK)$, 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, leading to much faster online planning.

Clearly the sampled scenarios have a major effect on the optimality of the chosen action. The original DESPOT algorithm sampled 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 equation (4) or (5). We show 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 Sections 5 and 6).

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* $\boldsymbol{\phi}_b = (s_0, \varphi_1, \varphi_2, \dots)$ for a belief b consists of a state s_0 sampled according to b and a sequence of random numbers $\varphi_1, \varphi_2, \dots$ sampled independently and uniformly over the range $[0, 1]$. The deterministic simulative model is a function $\mathcal{G}: S \times A \times \mathrm{R} \mapsto S \times Z$, 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

$$\begin{aligned} p(s', z'|s, a) &= p(s'|s, a)p(z'|s', a) \\ &= T(s, a, s')\,O(s', a, z') \end{aligned}$$

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 $(a_1, a_2, \dots)$ under a scenario $\boldsymbol{\phi}_b = (s_0, \varphi_1, \varphi_2, \dots)$, we start at s_0 and apply the deterministic simulative model $\mathcal{G}$ at each time step. The resulting simulation sequence $\boldsymbol{\zeta} = (s_0, a_1, s_1, z_1, a_2, s_2, z_2, \dots)$ traverses a path $(a_1, z_1, a_2, z_2, \dots)$ in the belief tree, starting at its root (Figure 1(b)). The nodes and edges along this 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 $\mathcal{O}(|A|^DK)$.

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 $V_\pi(b)$ using unnormalized importance sampling equation (4), IS-DESPOT samples a subset $\mathcal{Z}' \subset \mathcal{Z}$ according to a given 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

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

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 $\boldsymbol{\zeta}_{0:t}$.

Now it appears that we just need to find a policy π that maximizes $\hat{V}_\pi(b)$. Observe, however, that $\hat{V}_\pi(b)$ is calculated with respect to a set of sampled scenarios while our goal is to find a policy optimal under all scenarios and not just the sampled ones. 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 a regularization constant. More details on the benefits of regularization are available in Ye et al. (2017).

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 equation (8) at b_0 and chooses the action $a = \pi(b_0)$ for execution.

To optimize equation (8), we substitute equation (7) into equation (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 $\mathcal{T}$; and 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 $\mathcal{T}$. At each node b, we may follow a default policy π_0 or explore one of the action branches by applying Bellman's equation recursively, similar to equation (3)

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

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 equation (10), π_0 denotes a given default policy; $s_{\boldsymbol{\zeta},\Delta(b)}$ denotes the state in $\boldsymbol{\zeta}$ at time step $\Delta(b)$; and $V_{\pi_0,s}$ is the value of π_0 starting from state s. The outer maximization in equation (10) chooses between following π_0 or exploring the action branches, 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 equation (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 an 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 expand $\mathcal{T}$ by adding new nodes at the end of the path. The algorithm then traces the path back to b_0 and applies equation (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. To achieve the best performance, there are various techniques for performing the tree search efficiently and constructing good upper and lower bounds. We refer the reader to Ye et al. (2017) for details.

In summary, the core idea of IS-DESPOT is to sample a set of scenarios according to the importance distribution, construct a DESPOT incrementally, and search the tree to find an optimal action under the reweighted scenarios. It is conceptually simple and easy to implement.

3.4. Analysis

We now show that IS-DESPOT retains the strong theoretical guarantee of DESPOT. The two theorems below generalize the earlier results (Somani et al., 2013) 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 on the value of a policy.

Theorem 1. *Let b_0 be a given belief. Let $\Pi_{\mathcal{T}}$ be the set of all policy trees 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 policy $\pi \in \Pi_{b_0,D,K}$ satisfies*

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

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

$$W_{\max} = \left\{ \max_{\substack{s,s' \in S \\ a \in A, z \in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^D \tag{12}$$

is the maximum importance weight.

The estimation error bound in equation (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 equation (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 equation (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 $1-\tau$

$$V_{\hat{\pi}}(b_0) \geq \frac{1-\alpha}{1+\alpha}V_{\pi^*}(b_0) - \frac{R_{\max}W_{\max}}{(1+\alpha)(1-\gamma)} \times \left(\frac{\ln(8/\tau) + |\pi^*|\ln(KD|A||Z|)}{\alpha K} + (1-\alpha)\left(\sqrt{\frac{2\ln(2/\tau)}{K}} + \gamma^D\right) \right) \tag{13}$$

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.

3.5. Normalized importance sampling

We also applied normalized importance sampling to our algorithm; the algorithm remains basically the same, other than normalizing the importance weights. The analysis is also similar, but is more involved. Although the variance of a normalized importance sampling estimator and an unnormalized importance sampling estimator is incomparable in theory, the normalized estimator, in practice, often has a lower variance than that of the unnormalized estimator (Koller and Friedman, 2009). The decrease in variance often leads to better estimation of the policy value, despite the increase in bias, and helps to produce a better policy. Our experiments show that IS-DESPOT with normalized importance weights produces better performance in some cases (see Section 5).

4. Importance distributions

The importance distribution is a key element of importance sampling and, in particular, IS-DESPOT. We now derive the optimal importance distribution for IS-DESPOT. It provides insight for manually constructing importance distributions. It also forms the basis for a general method that learns importance distributions from data. To focus on the main idea, we restrict the discussion here to the importance distribution for sampling state transitions.

4.1. Optimal importance distributions

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), one crucial difference here is that IS-DESPOT estimates $f(s) = \mathbb{E}(v|s,\pi)$ by Monte Carlo simulation of π rather than evaluating $f(s)$ deterministically. Thus the importance distribution must take into consideration not only the mean $\mathbb{E}(v|s,\pi)$ but also the variance $\mathrm{Var}(v|s,\pi)$ resulting from Monte Carlo simulation; it gives 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 value, i.e. 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*

$$w_\pi(s) = \frac{\mathbb{E}_b\left(\sqrt{[\mathbb{E}(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)}\right)}{\sqrt{[\mathbb{E}(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)}} \tag{14}$$

Theorem 3 specifies the optimal importance distribution for a given policy π, but IS-DESPOT searches for an optimal policy and during the search, evaluates many policies in the 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 a random variable representing the total discounted reward of executing a policy, starting from some initial state. If two policies π and π' satisfy $\frac{[\mathbb{E}(v|s,\pi')]^2}{[\mathbb{E}(v|s,\pi)]^2} \leq 1+\epsilon$ and $\frac{\mathrm{Var}(v|s,\pi')}{\mathrm{Var}(v|s,\pi)} \leq 1+\epsilon$ for all $s \in S$ and some $\epsilon > 0$, then*

$$\mathrm{Var}\left(\hat{V}_{\pi',q^*_\pi}(b)\right) \leq (1+\epsilon)\left(\mathrm{Var}\left(\hat{V}_{\pi,q^*_\pi}(b)\right) + \frac{1}{K}V^*(b)^2\right) \tag{15}$$

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.

Theorem 4 quantifies the benefits of using an optimal importance distribution q^*_π for π to evaluate another similar policy π'. When ϵ is small, i.e. the two policies π and π' are close, the variance of estimating the value of π' under q^*_π, $\mathrm{Var}(\hat{V}_{\pi',q^*_\pi}(b))$, can be bounded in terms of $\mathrm{Var}(\hat{V}_{\pi,q^*_\pi}(b))$. Further, the variance becomes smaller as the number of samples, K, grows.

For a given DESPOT $\mathcal{T}$, the policies in the set $\Pi_\mathcal{T}$ are close to one another, because the branch-and-bound heuristic search helps to narrow down quickly the search space to a few near-optimal policies. Theorem 4 thus provides the basis for designing an importance distribution for a particular policy in $\Pi_\mathcal{T}$ and applying it to online DESPOT policy search.

4.2. Learning importance distributions

The common approach in the literature is to construct importance distributions manually. This is domain-specific and not always easy. Alternatively, we may learn importance distributions automatically from data. Our main idea is to perform a sufficiently large number of simulations *offline* to gather information on a domain. We then use the simulation data to learn an effective importance distribution for online planning using IS-DESPOT.

Theorems 3 and 4 provide the direction for the learning approach. They suggest that a good importance distribution for IS-DESPOT can be obtained, if the mean $\mathbb{E}(v|s,\pi)$ and the variance $\mathrm{Var}(v|s,\pi)$ are known for all $s \in S$ under a suitable policy $\pi \in \Pi_\mathcal{T}$. Two issues arise here. 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, to compute a policy. Theorem 4 helps to justify this choice. For the second issue, we use a discrete feature mapping over S. Specifically, we learn the function

$$\xi(s) = \sqrt{[\mathbb{E}(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)} \tag{16}$$

which is inversely proportional to the importance weight $w_\pi(s)$. We then set the importance distribution to $q(s) = \eta p(s)\xi(s)$, where η is a 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 according to the state-transition probability $p(s_t|s_{t-1}, a_t) = T(s_{t-1}, a_t, s_t)$, 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 onwards. 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 a feature vector $\mathcal{F}(s)$. Finally, We discretize the feature space into a finite set of bins of equal size and insert each collected data pair (s, v) into the bin containing $\mathcal{F}(s)$. We calculate the mean $\mathbb{E}(v)$ and the variance $\mathrm{Var}(v)$ for each bin. For any state $s \in S$, $\xi(s)$ is then approximated from the mean and the variance of the bin containing $\mathcal{F}(s)$.

We have two primary considerations in feature selection: the expected total discounted reward and its variance. For example, in navigation tasks, the distance between the agent and the goal is a useful feature, because the agent gets higher discounted reward as it approaches the goal. The distance between the agent and the nearest obstacle is another useful feature. If we assume imperfect robot control, this distance affects the probability of the robot hitting obstacles and in turn, the variance on the robot's expected total reward. See Section 5 for more examples. The number of features required is problem dependent. In our experiences, two to five features are usually enough.

This learning method is simple. It scales up to high-dimensional and continuous state spaces, provided that 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

Table 1. Performance comparison. The table shows the average total discounted reward (with 95% confidence interval) for 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
$\lvert S\rvert$	2	9,720	$\sim 10^{49}$	7,372,800	$\sim 10^{56}$
$\lvert A\rvert$	3	3	9	20	4
$\lvert Z\rvert$	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

Table 2. Importance distribution q for AsymmetricTiger. DESPOT samples according to p. IS-DESPOT samples according to q.

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

experimental results in the next section show that importance distributions captured by a small feature set effectively improve online planning for large POMDPs.

5. Experiments in simulation

We evaluated IS-DESPOT in simulation on a suite of five tasks (Table 1). The first three tasks are known to contain rare but critical states, states that are not encountered frequently, but may lead to significant consequences. The other two, RockSample (Smith and Simmons, 2004) and Pocman (Silver and Veness, 2010), 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 1 s per step. For POMCP, we slightly modified the software provided by the authors to make it strictly follow the 1 s time limit for planning. 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 (Ye et al., 2017) in AsymmetricTiger, CollisionAvoidance, and Demining. Following earlier work (Ye et al., 2017), we set the number of sampled scenarios in RockSample and Pocman to 500 and 100, respectively, 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. To learn the importance sampling distribution for IS-DESPOT, we applied the approach described in Section 4.2 and ran 50,000 simulations offline to collect data for 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.

5.1. *AsymmetricTiger*

Tiger is a classic POMDP problem (Kaelbling et al., 1998). 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 gathering and information exploitation prevalent in robot planning under uncertainty. We use it to gain understanding of 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 significant consequences. Sampling s_R is crucial.

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

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 (Table 2). Unnormalized and normalized IS-DESPOT are similar in performance, as normalization has little effect on this small toy problem.

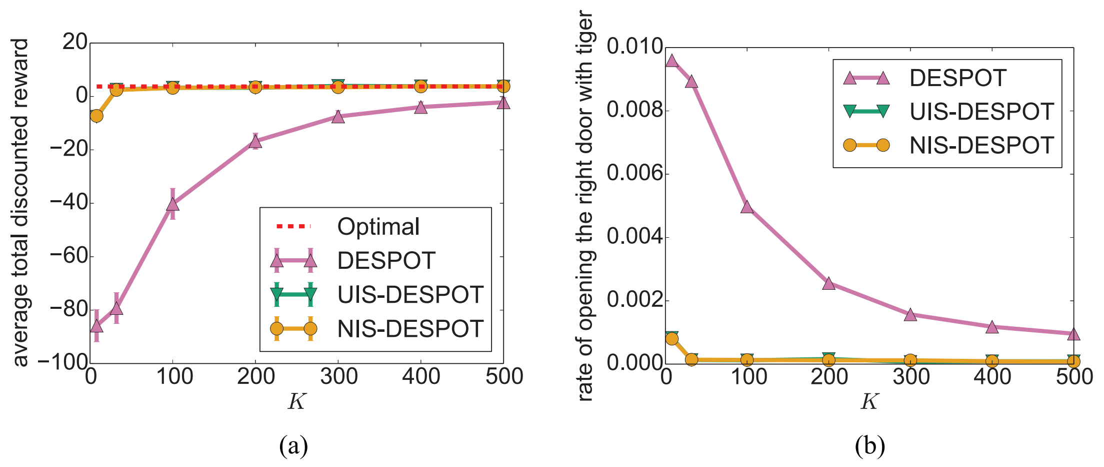


Fig. 2. AsymmetricTiger. (a) Average total discounted reward versus K, the number of sampled scenarios. (b) The rate of opening the right door with the tiger behind versus K.

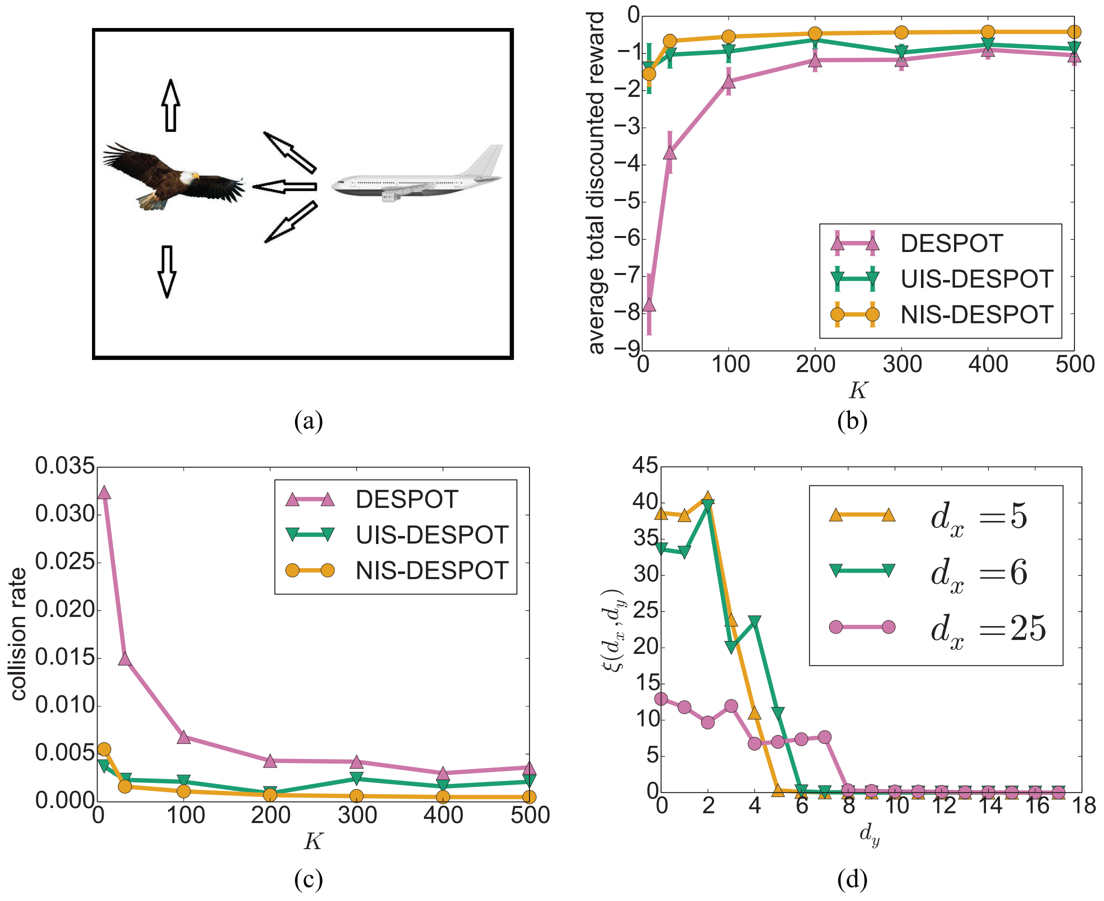


Fig. 3. CollisionAvoidance. (a) An aircraft changes direction to avoid a collision. (b) Average total discounted reward versus K, the number of sampled scenarios. (c) Collision rate versus K. (d) $\xi(d_x, d_y)$ learned from data.

We conducted additional experiments to understand how K, the number of sampled scenarios, affects performance (Figure 2). To calibrate the performance, we ran an offline POMDP algorithm, Successive Approximations of the Reachable Space under Optimal Policies (SARSOP), (Kurniawati et al., 2008) 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 $K = 32$, both versions of IS-DESPOT are near-optimal, while DESPOT does not reach a comparable performance level even at $K = 500$. All these confirm the benefit of importance sampling for small sample sizes.

One may ask how DESPOT and IS-DESPOT compare on the original Tiger task. The original Tiger task 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.

5.2. *CollisionAvoidance*

This is inspired by the aircraft collision avoidance task (Bai et al., 2012). The agent starts moving from a random position in the right-most column of a 18×30 grid map (Figure 3(a)). An obstacle randomly moves in the left-most column of this map. It moves up with probability 0.25, moves down with probability 0.25, and stays put with probability 0.50. The probabilities become 0, 0.25, and 0.75, respectively when the obstacle is in the top-most row, and become 0.25, 0, and 0.75, respectively when it is in the bottom-most row. At each time step, the agent may 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 observed position, affected by the Gaussian noise, is rounded to the nearest grid cell. The probability of the agent colliding with the obstacle is small, but the penalty of collision is high.

To learn 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 agent and the obstacle, 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. When the agent is closer to the obstacle, the probability of collision is higher, resulting in a worse mean and larger variance.

Figure 3(d) plots $\xi(d_x, d_y)$ for a few chosen horizontal distances d_x. 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 a collision would require the agent and obstacle to move towards each other for several steps in the remaining $d_x = 5$ steps. This is unlikely to happen, as the DESPOT policy actively tries to avoid collisions. Furthermore, collision is not possible at all when $d_y \geq 10$ because it requires at least 6 steps for the agent and the obstacle to meet. With this learned importance distribution, IS-DESPOT ignores states unlikely to result in collisions and focuses on sampling the rest. This significantly improves the planning performance, as shown in Figures 3(b) and (c). Unnormalized IS-DESPOT (UIS-DESPOT) performs better than DESPOT with small K. Normalized IS-DESPOT (NIS-DESPOT) improves the performance further. Both perform well with small K, while DESPOT cannot reach the same level of performance even with a substantially larger K.

5.3. *Demining*

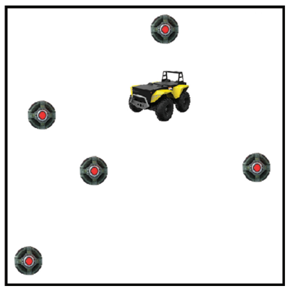
This is adapted from the robotic demining task in the Humanitarian Robotics and Automation Technology Challenge (HRATC) 2015 (Madhavan et al., 2015). It requires an agent to sweep a field, represented as a 10×10 grid, to detect and report landmines. Each grid cell contains a mine with probability 0.05. At each time step, the agent may move one step along one of the four orthogonal directions and then observe whether the adjacent cells of the new location contain mines with 90% accuracy. The agent receives a reward of $+10$ if it reports a mine correctly and receives -10 otherwise. If the agent steps over a mine, there is a high penalty $-1,000$ and the task terminates. DESPOT was a core component of the system that won HRATC 2015. We show that IS-DESPOT handles rare but critical states in the task better than DESPOT, resulting in improved performance overall.

To learn the importance distribution for IS-DESPOT, we map a state s to a feature vector (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 changes in the mean and variance of policy values for different states, because n determines the maximum total discounted reward that the agent can get, and d affects the probability of stepping over mines.

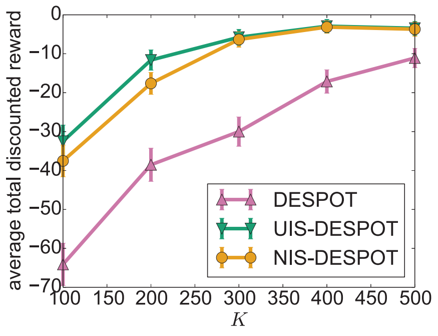
Figure 4(b) shows the performance comparison between IS-DESPOT and DESPOT for increasing number of scenarios. IS-DESPOT converges faster than DESPOT and significantly outperforms DESPOT even for a relatively large number of scenarios ($K = 500$). Figure 4(c) confirms that the performance improvement results from the decreased number of mine 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.

5.4. *RockSample*

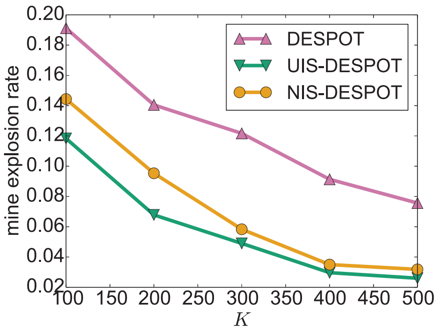
RockSample is a standard POMDP benchmark problem (Smith and Simmons, 2004) (Fig.5). In the problem *RockSample*(n, k), the agent moves on an $n \times n$ grid map that has k rocks. Each of the rocks can either be 'good' or 'bad'. The agent knows each rock's position but does not know its state (good or bad). At each time step, the agent may move to an adjacent grid cell deterministically, make a long-range noisy observation on the state of any rock, or sample a rock if there is one at the agent's current location. The observation accuracy decreases exponentially with respect to the



(a)



(b)



(c)

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

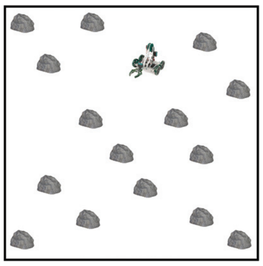


Fig. 5. RockSample. A robot senses rocks to identify 'good' ones and samples them. Upon completion, it exits the east boundary.

distance between the agent and the rock. The agent receives a reward of +10 if a sampled rock is good and receives −10 otherwise. A rock turns bad after being sampled. When the agent exits from the east boundary of the map, it gets a reward of +10, and the task terminates.

To learn the importance distribution, we map a state s to a feature vector (n, d_+, d_-), where n is the number of remaining good rocks, and d_+ and d_- are the Manhattan distances to the nearest good rock and the nearest bad rock, respectively.

This task does not contain rare but critical states. IS-DESPOT achieves comparable performance to DESPOT (Table 1)

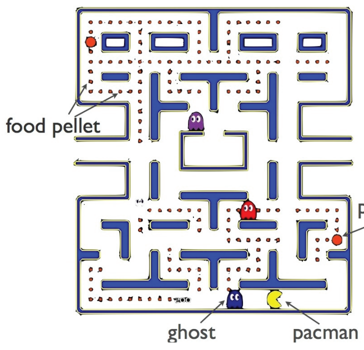
5.5. Pocman

Pocman (Silver and Veness, 2010) is a partially observable variant of the popular Pacman game (Figure 6(a)). An agent and four ghosts move in a 17 × 19 maze populated with 'food pellets' 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 a food pellet with probability 0.5. Eating a food pellet 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 agent 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; if the agent possesses the power pill, the ghost runs away, but slips with probability 0.25. When $d > 5$, the ghost moves uniformly at random to feasible adjacent cells. Unlike 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 $d \leq 2$.

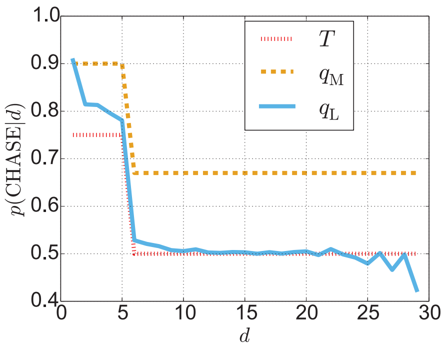
To learn the importance distribution, we map a state s to a feature vector $(n, d_{\min})$,where $d_{\min}$ is the Manhattan distance to the nearest ghost and n is the number of remaining steps for which the agent retains a power pill.

Table 1 shows that UIS-DESPOT does not provide improvement over DESPOT. However, NIS-DESPOT does, because normalization of importance weight reduces the high variance in this complex task.

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 penalties. An effective importance distribution must sample such states with increased probability. One crucial aspect of ghost behavior is the decision to chase the agent, governed by 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. This manually constructed importance distribution q_{M}, when used in normalized IS-DESPOT, achieved an average total discounted reward of 317.02 ± 4.21, which is weaker than that of the learned importance distribution q_{L}. To understand why, let us compare q_{M} with q_{L} (Figure 6(b)). Although q_{L} does result in the ghost chasing the agent more often than q_{M} does, it does not increase $p(\text{CHASE}|d)$ uniformly. The increase depends on the distance d. This suggests that even when our intuition is correct, getting the details of the importance distribution right is not always straightforward. Learning the importance distribution from data provides substantial benefits in such cases.



(a)



(b)

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

6. Experiments with an autonomous vehicle

We further evaluated IS-DESPOT on a single-seater autonomous vehicle.

6.1. System overview

Autonomous driving has been gaining popularity in recent years. Our single-seater autonomous vehicle, called *Scooter*, is designed to drive in densely populated urban areas, such as town plazas, shopping malls, and hospital complexes (Figure 7). It drives at moderate speed, but in close proximity to pedestrians. A primary challenge for Scooter is to handle the complex, dynamic environment with many walking pedestrians and sudden changes in pedestrian behavior, in order to drive safely, efficiently and smoothly.

Figure 8 gives an overview of the vehicle hardware platform. The sensor package consists of two LIDARs, an inertia measurement unit (IMU), and wheel encoders. The top-mounted SICK LMS151 LIDAR and the bottom-mounted SICK TiM551 LIDAR, both scan at 50 Hz over 270°, with a maximum range of 50 m and 10 m, respectively. They are used for localization and pedestrian detection. The onboard computer is fitted with an Intel Core i7-4770R processor running at 3.90 GHz and 16 GB RAM. Scooter drives at a maximum of 8 km/h, roughly the speed of fast human walking.

Our autonomous driving system replans online at each time step in near real time. We adopt a two-level hierarchical approach developed earlier (Bai et al., 2015). At the high level, we use the hybrid A* algorithm (Stanley, 2006) to search for a path. At the low level, we use a POMDP algorithm to control the vehicle speed along the planned path. To drive reliably near pedestrians, one key issue is to infer pedestrian intentions, which dictate their walking behaviors. Our intention POMDP model hedges against the uncertainty in pedestrian intention estimates and reasons about the long-term effects of the immediate vehicle control.

Fig. 7. Our single-seater autonomous vehicle, *Scooter*, driving amidst pedestrians.

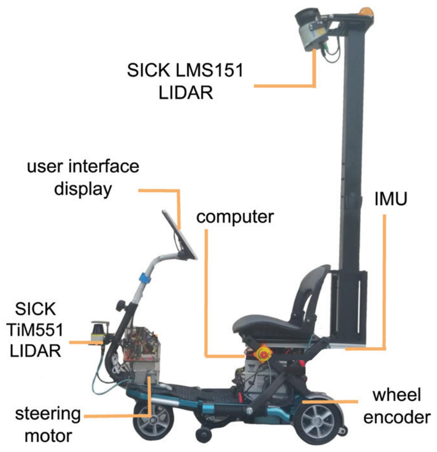


Fig. 8. The vehicle platform.

The autonomous driving system is implemented on top of the *robot operating system* (ROS). Scooter localizes itself in a given map through adaptive Monte Carlo localization (Thrun et al., 2005), which integrates the LIDAR, the IMU, and the wheel encoder data.

6.2. Intention POMDP and importance sampling

In our intention POMDP model, a state consists of two components: the vehicle state and the pedestrian state. The vehicle state contains its position, orientation, and speed. The pedestrian state contains the position, speed and goal location of each pedestrian. We model a pedestrian's intention as his/her goal location and assume that a pedestrian walks towards his/her goal directly along a shortest path with Gaussian noise. The pedestrian goals are hidden variables and must be inferred from observations over time. An *observation* consists of the position and speed of the vehicle and the positions of all pedestrians. The vehicle may choose from three discrete actions, ACCELERATE, MAINTAIN, and DECELERATE, which modulate the vehicle speed. Both the ACCELERATE and DECELERATE actions may fail with small probability. When they fail, the vehicle maintains its current speed.

Our earlier work (Bai et al., 2015) used DESPOT to solve the POMDP model. Collision or near-collision incidents are clearly rare, but critical events. If DESPOT fails to sample such events, the chosen actions may be suboptimal and lead to an emergency stop or collision; it is reasonable to expect that IS-DESPOT provides improved performance here.

To apply IS-DESPOT, we need to construct an effective importance distribution. We analyzed the near-collision incidents in our experiments with DESPOT and found three main causes. First, DESPOT fails to sample events in which a pedestrian suddenly changes his/her intention and walks towards an alternative goal. Second, DESPOT fails to sample events in which DECELERATE fails for several steps in a row. Third, DESPOT fails to sample events in which pedestrians walk directly into the vehicle, when they are distracted by their mobile phones or some sudden event. All three types of events are rare, but critical. We manually constructed an importance distribution to boost the probability of sampling these events. We did not apply the learning method, because of the difficulty in obtaining sufficient driving data on the vehicle.

6.3. Results

We conducted extensive experiments to evaluate IS-DESPOT's online planning performance on a plaza in the University Town of our campus. The area is roughly 70 m× 60 m in size, with many people passing by (Figure 7). In the experiments, we set the number of sampled scenarios to 100. We used a simple reactive controller as the default policy for the IS-DESPOT search. The reactive controller chooses ACCELERATE, MAINTAIN, or DECELERATE based on the distance D from the nearest pedestrian to the vehicle. It compares D with two distance thresholds D_{near} and D_{far}. Then it chooses DECELERATE if $D \leq D_{\text{near}}$, chooses MAINTAIN if $D_{\text{near}} < D < D_{\text{far}}$, and chooses ACCELERATE if $D \geq D_{\text{far}}$. The autonomous driving system performed online path planning at about 2 Hz and POMDP planning for speed control at 3 Hz. The maximum vehicle speed was set to 1.2 m/s. We used exactly the same settings for DESPOT when comparing performance.

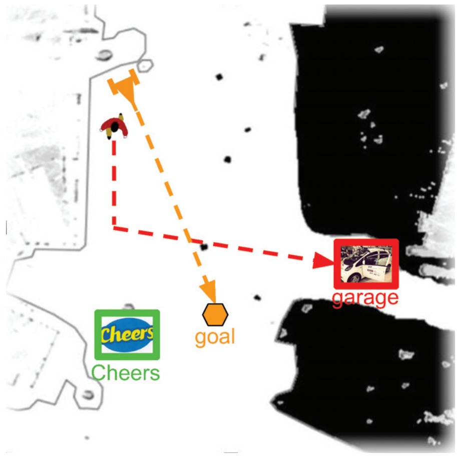


Fig. 9. An enacted scene on the University Town plaza. Initially the pedestrian appears to walk towards Cheers, but abruptly changes direction and moves towards the garage after a few steps.

Overall, the autonomous driving system performed well. Scooter did not have any collision or near-collision incidents. It did not cause alarm to pedestrians nearby. It also had few sudden acceleration changes, which may cause passenger discomfort. The online video[1] shows some sample runs.

On the real robot, a comparison of DESPOT and IS-DESPOT is much more difficult, as it is impossible to repeat the exact same dynamic scene. To illustrate their performance differences, we enacted a scene in which a pedestrian suddenly changes his intention (Figure 9). The pedestrian has three possible intentions: going to Cheers, going to the garage, or standing still. Initially the pedestrian appears to walk towards Cheers, but abruptly changes direction and moves towards the garage after a few steps.

Scooter starts with a uniform belief over the three pedestrian intentions (Figure 10, top row, frame 1). As the pedestrian walks towards Cheers for multiple steps, the belief on the pedestrian going to Cheers increases significantly (top row, frame 2). The probability of going to the garage is very low. So is the probability of staying put. DESPOT samples events according to their natural probability of occurrence, i.e. the belief. Hence it obtains almost no samples for going to the garage. Without being aware of this possibility, it then chooses to accelerate in order to reach the goal faster and get a higher reward. Suddenly, the pedestrian changes his direction (top row, frame 3), which causes a quick increase in belief on the pedestrian going to the garage. Noticing that increase, DESPOT decides to decelerate in order to avert a possible collision. However, Scooter cannot slow down fast enough, because of the high speed resulting from the ACCELERATE action earlier. Therefore, emergency stop is triggered (top row, frame 4).

In almost the same scene, Scooter performs much better under IS-DESPOT (Figure 10). At frame 2 (bottom

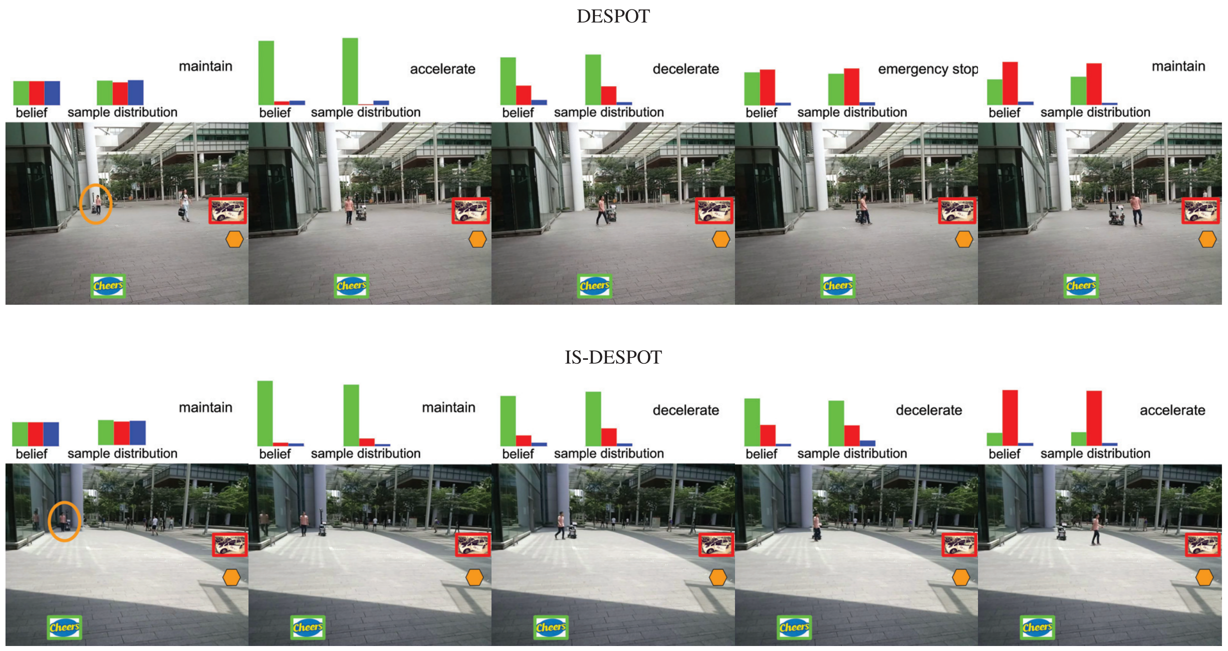


Fig. 10. Scooter encounters a pedestrian who suddenly changes his intention. Under DESPOT, Scooter triggers the emergency stop. Under IS-DESPOT, Scooter reaches its goal (orange hexagon) successfully. Above every video frame, there are two histograms. Each histogram shows the probabilities of the pedestrians' three intentions: going to Cheers (green), going to the garage (red), and standing still (blue). The histogram on the left shows the current belief. The histogram on the right shows the distribution of samples chosen by DESPOT or IS-DESPOT.

row), the belief on the pedestrian going to the garage is low. However, the importance distribution boosts the probability of sampling the event, because of the severe consequences of a collision. IS-DESPOT hence obtains many more such samples than DESPOT. As a result, IS-DESPOT chooses to maintain the current speed rather than accelerate, which helps to avoid the near-collision incident later. When the pedestrian changes his intention and walks towards the garage, IS-DESPOT chooses two successive DECELERATE actions (bottom row, frames 3 and 4). Therefore, Scooter slows down enough to let the pedestrian pass through and then accelerates toward its goal (bottom row, frame 5). Overall, Scooter reaches its goal successfully without any incidents.

A comparison of the sample distributions for DESPOT and for IS-DESPOT at frame 2 (Figure 10) suggests that DESPOT fails to sample the event that the pedestrian goes to the garage adequately. This is the direct cause of its aggressive acceleration, which eventually results in the near-collision incident.

To confirm and quantify the performance difference between DESPOT and IS-DESPOT, we performed additional simulation experiments in a similar setting. In simulation, we can run the two algorithms many times in exactly the same dynamic scenes. The vehicle drives for 10 m along a straight line in a 7 m $\times$ 17 m area (Figure 11(a)). Six pedestrians are initialized with random positions and goals. We compared the two algorithms according to three measures: the average collision rate as a measures of safety, the average travel time for the vehicle to reach the goal as a measure of efficiency, and the average total acceleration over time as a measure of driving smoothness. We performed a large number of simulations, 15,000 trials each for both IS-DESPOT and DESPOT, in order to obtain reliable data, because the low collision rate makes it difficult to obtain an accurate estimate.

The results show that the average collision rate for IS-DESPOT is nearly four times lower than for DESPOT (Figure 11(b)). Further, the collision rate does not decrease much as the collision penalty increases. Together this suggests that DESPOT likely fails to sample some critical events, and the performance gain of IS-DESPOT comes from improved sampling. DESPOT and IS-DESPOT are comparable in terms of average travel time and total acceleration (Figures 11(c) and (d)). Overall IS-DESPOT improves driving safety without much sacrifice on efficiency or smoothness.

7. Discussion

The importance distribution has a significant impact on the performance of IS-DESPOT. Sometimes we can construct it manually based on domain-specific knowledge (Section 6). It is, however, not always easy to do so effectively, as we have seen in the Pocman task (Section 5). To automate importance distribution construction, we propose to learn it from data. Our learning method is general and domain-independent. However, it still has several limitations. First,

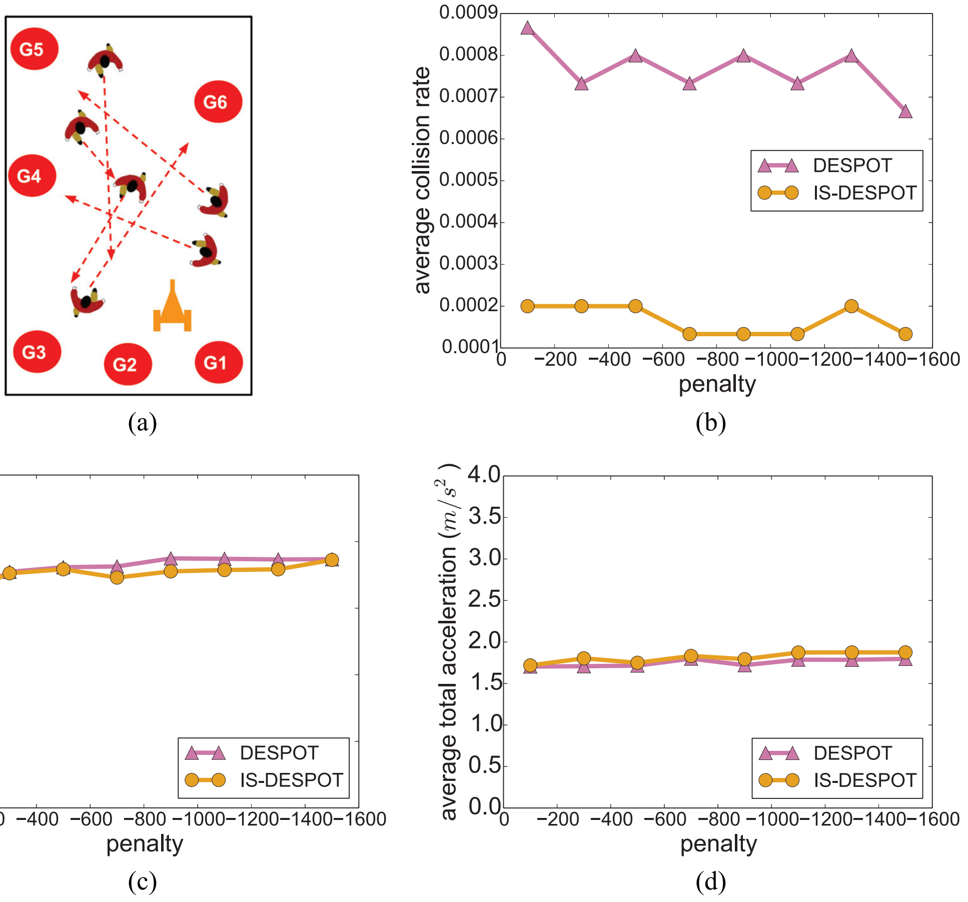


Fig. 11. Performance comparison of DESPOT and IS-DESPOT in autonomous driving amidst pedestrians. (a) An example simulation environment. (b) Average collision rate versus collision penalty. (c) Average travel time versus collision penalty. (d) Average total acceleration versus collision penalty.

it requires a realistic simulator and significant offline computational time for simulation in order to generate data. Second, when a task involves a very large state space, we rely on discrete feature mapping over the state space during learning. Currently, feature selection is manual and relies on domain-specific knowledge. Selecting features is simpler than specifying the exact importance distribution. Our method has eased the difficulty of importance distribution construction through learning, but it is not yet fully automated.

We can also potentially further improve the learning of the importance distribution through an iterative scheme. Concretely, we start with an initial importance distribution q0. We then run IS-DESPOT with q0 offline to gather data and learn the importance distribution q1. We iteratively learn the importance distribution qk using the data gathered from IS-DESPOT with qk–1. The rationale behind the iterative learning scheme is that, at each iteration, IS-DESPOT policy is improved by adopting the importance distribution generated from its previous iteration, and the improved policy further helps to learn a better importance distribution for the next iteration.

While IS-DESPOT is presented as a POMDP algorithm, it is in fact more general and applies to belief-space planning with or without the specific mathematical structure of the POMDP. To plan near-optimal actions, it requires only a black-box simulation model for sampling state-transitions and observations; it does not require the full probability distribution models. This is a major advantage in practice. In fact, constructing accurate probability distribution models for complex tasks such as Pocman or autonomous driving among pedestrians is extremely difficult. Further, the formal performance guarantee of IS-DESPOT carries through to the more general setting of belief-space planning without modification.

8. Conclusion

This paper introduces importance sampling to sampling-based online planning under uncertainty. 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 several distinct robotic 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. Third, we want to adopt the iterative learning scheme for learning the importance distribution and investigate its convergence. Finally, the idea of importance sampling is general and can be applied to other MDP and POMDP planning algorithms (Bai et al., 2010; Kearns et al., 2002; Silver and Veness, 2010).

Acknowledgements

We thank the anonymous reviewers for their insightful comments and suggestions which helped considerably in improving the paper.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported in part by NUS AcRF Tier 1 grant R-252-000-587-112 and US Air Force Research Laboratory under agreement number FA2386-15-1-4010.

ORCID iD

Yuanfu Luo, https://orcid.org/0000-0002-9804-6204

Note

1. http://motion.comp.nus.edu.sg/2018/04/23/online-pomdp-planning/

References

Bai H, Cai S, Ye N, Hsu D and Lee WS (2015) Intention-aware online POMDP planning for autonomous driving in a crowd. In: *2015 IEEE international conference on robotics & automation (ICRA)*, Seattle USA, 26–30 May 2015, Piscataway: IEEE Press, pp. 454–460.

Bai H, Hsu D, Kochenderfer MJ and Lee WS (2012) Unmanned aircraft collision avoidance using continuous-state POMDPs. In: *Robotics: science and systems VII*, Sydney, Australia, 9–13 July 2012. Cambridge: MIT Press, pp. 1–8.

Bai H, Hsu D, Lee WS and Ngo VA (2010) Monte Carlo value iteration for continuous-state POMDPs. In: *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics IX (WAFR)*. New York: Springer, pp. 175–191.

Bertsekas D (2005) *Dynamic Programming and Optimal Control*. MA: Athena Scientific.

Boor V, Overmars M and van der Stappen F (1999) The Gaussian sampling strategy for probabilistic roadmap planners. In: *1999 IEEE international conference on robotics & automation (ICRA)*, Michigan, 10–15 May 1999. Piscataway: IEEE Press, pp. 1018–1023.

Folsom-Kovarik J, Sukthankar G and Schatz S (2013) Tractable POMDP representations for intelligent tutoring systems. *ACM Transactions on Intelligent Systems & Technology* 4(2), 29:1–29:22.

Gelly S and Silver D (2007) Combining online and offline knowledge in UCT. In: *24th annual international conference on machine learning (ICML)*, Corvallis, USA, 20–24 June 2007. New York: ACM Press, pp. 273–280.

Glasserman P (2003) *Monte Carlo Methods in Financial Engineering*. New York: Springer.

Goldberg K (1993) Orienting polygonal parts without sensors. *Algorithmica* 10(2): 201–225.

Hauser K (2010) Randomized belief-space replanning in partially-observable continuous spaces. In: *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics IX (WAFR)*, Singapore, 13–15 December 2010. New York: Springer, pp. 193–209.

He R, Brunskill E and Roy N (2011) Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research* 40(1), 523–570.

Hsiao K, Kaelbling LP and Lozano-Perez T (2007) Grasping POMDPs. In: *2007 IEEE international conference on robotics & automation (ICRA)*, Roma, Italy, 10–14 April 2007. Piscataway: IEEE Press, pp. 523–570.

Kaelbling LP, Littman ML and Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1): 99–134.

Kalos M and Whitlock P (1986) *Monte Carlo Methods*. New York: John Wiley & Sons.

Kearns M, Mansour Y and Ng AY (2002) A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning* 49(2–3): 193–208.

Koller D and Friedman N (2009) *Probabilistic Graphical Models: Principles and Techniques*. Cambridge: MIT Press.

Koval M, Hsu D, Pollard N and Srinivasa S (2016a) Configuration lattices for planar contact manipulation under uncertainty. In: *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics XII (WAFR)*, San Francisco, 18–20 December 2016. New York: Springer.

Koval M, Pollard N and Srinivasa S (2016b) Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. *International Journal of Robotics Research* 35(1–3): 244–264.

Kurniawati H and Hsu D (2004) Workspace importance sampling for probabilistic roadmap planning. In: *IEEE/RSJ international conference on intelligent robots & systems*, Sendai, Japan, 28 September-2 October 2004. Piscataway: IEEE Press, pp. 1618–1623.

Kurniawati H, Hsu D and Lee WS (2008) SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Robotics: science & systems* Zurich, Switzerland, 25–28 June 2008, Cambridge: MIT Press, pp. 65–72.

Lozano-Pérez T, Mason M and Taylor R (1984) Automatic synthesis of fine-motion strategies for robot. *International Journal of Robotics Research* 3(1): 3–24.

Lumelsky V and Stepanov A (1987) Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica* 2(1): 403–430.

Madhavan R, Marques L, Prestes E, Maffei R, Jorge V, Gil B et al. (2015) 2015 Humanitarian Robotics and Automation Technology Challenge. *IEEE Robotics and Automation Magazine* 22(3): 182–184.

Owen AB (2013) *Monte Carlo Theory, Methods and Examples*. Unpublished manuscript.

Papadimitriou C and Tsisiklis J (1987) The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3): 441–450.

Pineau J, Gordon G and Thrun S (2003) Point-based value iteration: an anytime algorithm for POMDPs. In: *18th international joint conference on artificial intelligence (IJCAI)*, Acapulco, Mexico, 9–15 August 2003, San Francisco: Morgan Kaufmann, pp. 1025–1032.

Ross S, Pineau J, Paquet S and Chaib-Draa B (2008) Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32(1): 663–704.

Shelton CR (2001) Policy improvement for POMDPs using normalized importance sampling. In: *17th conference on uncertainty in artificial intelligence (UAI)*, Seattle, 2–5 August 2001. San Francisco: Morgan Kaufmann, pp. 496–503.

Silver D and Veness J (2010) Monte-Carlo planning in large POMDPs. In: *Neural information processing systems conference 2010 (NIPS)* Vancouver, Canada, 6–9 December 2010, NY: Curran Associates Inc., pp. 2164–2172.

Smallwood R and Sondik E (1973) The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21(2): 282–304.

Smith T and Simmons R (2004) Heuristic search value iteration for POMDPs. In: *20th conference on uncertainty in artificial intelligence (UAI)*, Banff, Canada, 7–11 July 2004. Arlington: AUAI Press, pp. 520–527.

Somani A, Ye N, Hsu D and Lee WS (2013) DESPOT: online POMDP planning with regularization. In: *Neural information processing systems conference 2013 (NIPS)* Lake Tahoe, USA, 5–8 December 2013, NY: Curran Associates, Inc., pp. 1772–1780.

Spaan M and Vlassis N (2005) Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research* 24: 195–220.

Stanley TS (2006) The robot that won the DARPA grand challenge: research articles. *Journal of Robotic Systems* 23(9): 661–692.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge: MIT Press.

van den Berg J, Abbeel P and Goldberg K (2011) LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal of Robotics Research* 30(7): 895–913.

van den Berg J and Overmars M (2005) Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. *International Journal of Robotics Research* 24(12): 1055–1071.

van den Berg J, Patil S and Alterovitz R (2012) Motion planning under uncertainty using iterative local optimization in belief space. *International Journal of Robotics Research* 31(11): 1263–1278.

Veach E (1997) *Robust Monte Carlo methods for light transport simulation*. PhD Thesis, Stanford University, USA.

Wilmarth S, Amato N and Stiller P (1999) MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space. In: *1999 IEEE international conference on robotics & automation (ICRA)*, Michigan, 10–15 May 1999, Piscataway: IEEE Press, pp. 1024–1031.

Wu K, Lee WS and Hsu D (2015) POMDP to the rescue: boosting performance for RoboCup rescue. In: *IEEE/RSJ international conference on intelligent robots & systems*. Hamburg, Germany, 28 September-2 October 2015. Piscataway: IEEE Press, pp. 5294–5299.

Ye N, Somani A, Hsu D and Lee WS (2017) DESPOT: online POMDP planning with regularization. *Journal of Artificial Intelligence Research* 58: 231–266.

A. Appendix: Proofs

A.1. Theorems 1 and 2

To prove Theorem 1, we rely on an earlier result (Somani et al., 2013).

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 subsequence 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 the DESPOT algorithm on this new POMDP model and running IS-DESPOT on the original model generates exactly the same simulation sequences with the same total rewards. It then follows from Theorem 5 that

$$V'_\pi(b_0) \geq \frac{1-\alpha}{1+\alpha}\hat{V}_\pi(b_0) - \frac{R'_{\max}}{(1+\alpha)(1-\gamma)} \cdot \frac{\ln(4/\tau) + |\pi| \ln(KD|A||Z|)}{\alpha K} \quad (17)$$

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

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

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

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

Hence

$$\begin{aligned} R'_{\max} &\le \left\{ \max_{\substack{s,s'\in S \\ a\in A, z\in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^t R_{\max} \\ &\le \left\{ \max_{\substack{s,s'\in S \\ a\in A, z\in Z}} \frac{p(s,z|s',a)}{q(s,z|s',a)} \right\}^D R_{\max} \\ &= W_{\max} R_{\max} \end{aligned} \tag{18}$$

as $t \le D$. Combining equations (17) and (18), we have

$$\begin{aligned} V'_\pi(b_0) \ge & \frac{1-\alpha}{1+\alpha}\hat{V}_\pi(b_0) - \frac{R_{\max}W_{\max}}{(1+\alpha)(1-\gamma)} \\ & \cdot \frac{\ln(4/\tau) + |\pi|\ln(KD|A||Z|)}{\alpha K} \end{aligned} \tag{19}$$

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

$$\begin{aligned} V'_\pi(b_0) &= \sum_{t=0}^{\infty} \int \frac{p(\boldsymbol{\zeta}_{0:t}|b_0,\pi)}{q(\boldsymbol{\zeta}_{0:t}|b_0,\pi)} \gamma^t R(s_t, a_{t+1}) q(\boldsymbol{\zeta}_{0:t}|b_0,\pi)\, \mathrm{d}\boldsymbol{\zeta}_{0:t} \\ &= \sum_{t=0}^{\infty} \int p(\boldsymbol{\zeta}_{0:t}|b_0,\pi) \gamma^t R(s_t, a_{t+1})\, \mathrm{d}\boldsymbol{\zeta}_{0:t} \\ &= \int \sum_{t=0}^{\infty} \gamma^t R(s_t, a_{t+1}) p(\boldsymbol{\zeta}|b_0,\pi)\, \mathrm{d}\boldsymbol{\zeta} \\ &= V_\pi(b_0) \end{aligned} \tag{20}$$

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

Finally, putting together equations (19) and (20), 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 Somani et al. (2013). 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 belief b, with the importance distribution $q(s)$. The variance of the estimator $\frac{b(s)}{q(s)}v$ is

$$\begin{aligned}
\mathrm{Var}\left(\frac{b(s)}{q(s)}v\middle|\pi\right) &= \mathbb{E}\left(\frac{b(s)^2}{q(s)^2}v^2\middle|\pi\right) - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)^2}v^2 p(v|s,\pi)\,q(s)\;\mathrm{d}v\,\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)}\left(\int v^2 p(v|s,\pi)\;\mathrm{d}v\right)\;\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2 \\
&= \int \frac{b(s)^2}{q(s)}\left([\mathbb{E}\,(v|s,\pi)]^2 + \mathrm{Var}(v|s,\pi)\right)\;\mathrm{d}s - \left[\mathbb{E}\left(\frac{b(s)}{q(s)}v\middle|\pi\right)\right]^2
\end{aligned} \tag{21}$$

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 equation (21), we get

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

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

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

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 equation (21). 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

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

Similarly to equation (21), 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

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

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

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

Similar to equation (21), we have

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

Hence

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

□