Article

The International Journal of Robotics Research
2014, Vol. 33(9) 1288–1302

DOI: 10.1177/0278364914528255
ijr.sagepub.com

Integrated perception and planning in the continuous space: A POMDP approach

Haoyu Bai, David Hsu and Wee Sun Lee

Abstract

The partially observable Markov decision process (POMDP) provides a principled mathematical model for integrating perception and planning, a major challenge in robotics. While there are efficient algorithms for moderately large discrete POMDPs, continuous models are often more natural for robotic tasks, and currently there are no practical algorithms that handle continuous POMDPs at an interesting scale. This paper presents an algorithm for continuous-state, continuous-observation POMDPs. We provide experimental results demonstrating its potential in robot planning and learning under uncertainty and a theoretical analysis of its performance. A direct benefit of the algorithm is to simplify model construction.

Keywords

Planning under uncertainty, continuous POMDPs

1. Introduction

Integrated perception and planning is essential for reliable robot operation and poses a major challenge in robotics. The partially observable Markov decision process (POMDP) provides a mathematical model that connects perception and planning in a principled manner. It has been applied to a range of robotic tasks, including navigation (Roy and Thrun, 1999), grasping (Hsiao et al., 2007), and aircraft collision avoidance (Bai et al., 2011). However, efficient POMDP algorithms existing today typically assume discrete models (e.g., Pineau et al., 2003; Smith and Simmons, 2005; Kurniawati et al., 2008), in which an agent's states, actions, and observations are all discrete, while for robotic tasks, continuous models are often more natural. To our knowledge, there are currently no practical algorithms that handle continuous POMDPs at an interesting scale for robotic tasks. Here we aim to develop an efficient algorithm for *continuous-state*, *continuous-observation*, but discrete-action POMDPs and apply it to robot planning and learning under uncertainty.

Because of uncertainty inherent in robot control and sensing, a robot does not know its state perfectly. To choose an action, it must consider all possible states consistent with actions taken and observations received. In a POMDP, we capture the state uncertainty in a *belief*, which can be represented as a probability distribution over the robot's state space. The set of all valid beliefs forms the belief space $\mathcal{B}$. An offline POMDP algorithm systematically reasons over $\mathcal{B}$ and tries to construct an optimal *policy* $\pi : \mathcal{B} \to A$, which prescribes to every belief in $\mathcal{B}$ a best action from an action set A.

To apply discrete POMDP algorithms to tasks with continuous states and observations, a common approach is to discretize state and observation spaces with a regular grid. This approach is difficult to scale up, as the computational cost increases exponentially for high-dimensional spaces. We avoid such fixed discretization. Conceptually we perform probabilistic sampling to "discretize" state and observation spaces *implicitly*, while computing the policy at the same time. The sampling adapts to the accuracy of policy computation.

One main challenge with continuous POMDPs lies in representing the belief and the policy: the belief is a continuous probability distribution, and the policy maps such a continuous probability function to an action. Developing finite representations for them is difficult. We introduce the *generalized policy graph* (GPG) as an alternative policy representation. Each node of a GPG is labeled with an action. Since the observation space is continuous, each node has an associated *edge classifier*, which is a function that maps an input observation to another GPG node. Intuitively we can think of a GPG as a finite-state controller with

Department of Computer Science, National University of Singapore, Singapore

Corresponding author:
David Hsu, Department of Computer Science, National University of Singapore, 13 Computing Drive, Singapore, 117417, Singapore.
Email: dyhsu@comp.nus.edu.sg

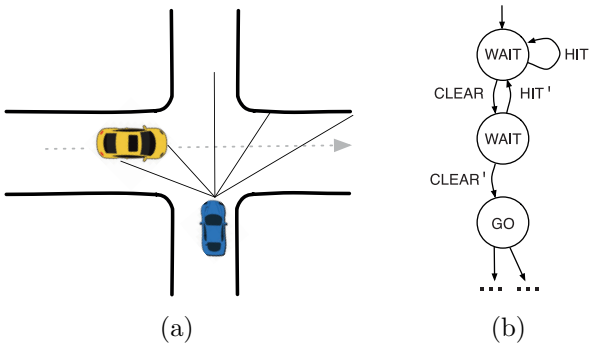


Fig. 1. Intersection navigation. (a) An autonomous vehicle, in blue, navigates through an uncontrolled road intersection. It is equipped with a sensor to measure proximity to obstacles along a set of beams. (b) An example GPG. CLEAR, CLEAR′, HIT, and HIT′ represent subsets of continuous observations.

continuous input and discrete output. Consider the example in Figure 1. Under this policy, the stopped autonomous vehicle moves forward only after receiving two successive observations from CLEAR and CLEAR′, respectively.

We construct a GPG by iteratively applying the Bellman backup equation to an initial policy. This is the same basic idea of the value iteration algorithm, but we perform backup on a GPG rather than a value function. To deal with continuous state and observation spaces, we evaluate the Bellman equation by Monte Carlo sampling. We also provide a performance bound on the number of samples required to compute an approximately optimal policy.

Our algorithm applies without modification to POMDPs with large discrete state or observation space as well.

One immediate benefit of the new algorithm is simplified model construction, as it removes the need to discretize states and observations manually. Consider the example in Figure 1 again. In a discrete POMDP model, we must choose discrete locations as the states for the vehicles. We must also construct observations. One way is to quantize the proximity sensor readings for each beam. Even with a coarse two-level quantization, this still results in 2^K observations for K beams. A more sophisticated observation model calculates the maximum-likelihood vehicle location by preprocessing the sensor readings and uses the estimated location as the observation. This reduces the number of observations, but may lose information during the preprocessing and degrade the quality of the computed policy (see Section 6.2). The new algorithm alleviates the difficulty of these modeling choices by sampling directly from the continuous state and observation spaces during the policy computation.

In addition to control and sensing uncertainty, we may also have model uncertainty. One way of dealing with this in a POMDP is to incorporate unknown model parameters into the state (Duff, 2002), thus performing planning and learning simultaneously. In this setting, one added benefit of the new algorithm is that it handles continuous model parameters directly, without the need to discretize them a priori (Wang et al., 2012).

2. Related work

With continuous states, a main difficulty in POMDP planning is belief representation. One approach is to restrict to a parametric class of beliefs, e.g., the Gaussian (Brooks et al., 2006; Prentice and Roy, 2007) or the Gaussian mixture (Porta et al., 2006). However, robotic tasks often involve beliefs with multiple modes and sharp edges, e.g., when a robot navigates through long, narrow corridors with few features. In this case, the Gaussian mixture has difficulty in scaling up. Other approaches use sampled representations such as the particle filter (Thrun, 2000; Porta et al., 2006; Bai et al., 2010), or aggregate sampled beliefs approximately (Davidson and Hutchinson, 2008). Our algorithm uses a sampled belief representation during the offline policy computation. For online policy execution, it exploits the policy graph representation and does not track the belief explicitly, thus avoiding the belief representation issue there.

Another difficulty with continuous states is policy representation. Instead of representing the policy directly, one may develop a hierarchical representation for the value function associated with the policy (Brechtel et al., 2013). The hierarchical representation would be effective if the value function is sufficiently smooth, but may have difficulty in scaling up to high-dimensional state spaces. We thus choose to use the policy graph, which is more direct and simpler. However, the two approaches are complementary and may be combined.

Continuous observations cause difficulty in a different way. A POMDP policy must condition on all future observations. Clearly it is impossible to enumerate an infinite number of continuous observations. Earlier work addresses this issue by aggregating observations, but it requires discrete states for policy representation (Hoey and Poupart, 2005). In contrast, our algorithm handles both continuous states and continuous observations, but discrete actions.

Instead of restricting the belief space, a different approach is to search a restricted policy class, e.g., finite-state controllers (Meuleau et al., 1999; Poupart and Boutilier, 2003), memoryless reactive policies (Bagnell et al., 2003), or various locally optimal policies (Erez and Smart, 2010; Hauser, 2010; Van den Berg et al., 2012). These methods, however, cannot guarantee the global optimality of the computed policy. We show in Section 5 that under reasonable conditions, our algorithm is guaranteed to find an approximately optimal policy with high probability.

Our algorithm computes offline a policy conditioned on future observations. An orthogonal direction is to perform forward search online (e.g., Ross et al., 2008; Hauser, 2010; Platt et al., 2010; Van den Berg et al., 2010; He et al., 2011), which chooses a single best action for the current belief. It does not compute a policy and completely avoids

the issue of policy representation for continuous state space. Online search and offline policy computation can be combined to solve difficult POMDPs, e.g., by using approximate or partial policies computed offline as default policies for online search.

Our algorithm evaluates the Bellman equation by Monte Carlo sampling. This is one basic idea of approximate dynamic programming (Powell, 2007) and used in various MDP/POMDP planning and reinforcement learning algorithms (e.g., Bagnell et al., 2003; Lagoudakis and Parr, 2003).

Our algorithmic approach builds on the policy search algorithm in (Hansen, 1998) and the MCVI algorithm (Bai et al., 2010). The former is designed for discrete POMDPs. The latter deals with continuous states, but only discrete observations.

3. POMDPs with continuous states and observations

3.1. The model

Formally, a POMDP is a tuple $(S, A, O, T, Z, R, \gamma)$, where S, A, and O denote a robot's state space, action space and observation space, respectively. At each time step, the robot takes an action $a \in A$ to move from a state $s \in S$ to $s' \in S$; it then receives an observation $o \in O$. The model for the system dynamics is specified by a conditional probability function $T(s, a, s') = p(s'|s, a)$, which accounts for uncertainty in robot control, unexpected environment changes, and so on. Similarly, the observation model is specified by a conditional probability function $Z(s', a, o) = p(o|s', a)$, which accounts for sensing uncertainty. The function $R(s, a)$ specifies a real-valued reward for the robot if it takes action a in state s. The robot's goal is to choose a sequence of actions that maximizes the expected total reward $\mathrm{E}\big(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\big)$, where s_t and a_t denote the system's state and action at time t. The discount factor $\gamma \in [0, 1)$ ensures that the total reward is finite, even when a planning task has an infinite horizon.

As a modeling language, the POMDP is agnostic about whether S, A, and O are continuous or discrete. The difference is, however, significant for belief and policy representations. In our model, both S and O are continuous, but A is discrete.

3.2. Beliefs

In a POMDP, we capture the robot's state uncertainty in a belief, which is a probability distribution over S. Suppose that b is the current belief on the robot state. If the robot executes action a and receives observation o, the new belief b_{ao} is calculated according to the Bayes's rule:

$$b_a(s') = \int_{s \in S} p(s'|s, a)\, b(s)\, \mathrm{d}s \tag{1}$$

$$b_{ao}(s) = \eta p(o|s, a)\, b_a(s) \tag{2}$$

where η is a normalizing constant. The update uses the system dynamics model and the observation model to integrate information from a and o into the new belief. Since S is continuous, we use a set of particles (Thrun et al., 2005) as a finite belief representation in the offline policy computation.

3.3. Policies

One common POMDP policy representation is a *policy graph*, which is a directed graph. Each node of a policy graph is labeled with an action from A, and each edge is labeled with an observation from O. The policy graph representation is compact, as its size depends only on the complexity of a policy and not on the size of the state space S. It has been used successfully in various algorithms for POMDPs with large discrete or continuous state space (e.g., Poupart and Boutilier, 2003; Bai et al., 2011).

However, each edge of a policy graph corresponds to a single observation, and this is unsuitable for POMDPs with large discrete or continuous observation space. The GPG generalizes the policy graph to POMDPs with continuous observation space by representing outgoing observation edges as a classifier that maps observations to subsequent policy graph nodes. Formally, a GPG G is a set of nodes. Each node $v = (a, \kappa)$ consists of an action $a \in A$ and a mapping $\kappa : O \to G$.

To execute a policy $\pi_{G,v}$ represented as a GPG G, we start at a node $v = (a, \kappa)$ in G and take the action a. Upon receiving an observation o, we move to the next node $v' = \kappa(o)$. The process then repeats at the new node $v' = (a', \kappa')$. We do not track beliefs explicitly using (1) and (2), but instead represent beliefs implicitly as histories of actions and observations.

Each node $v \in G$ induces an α-function $\alpha_v : S \to \mathbb{R}$, which defines the expected total reward of executing $\pi_{G,v}$ starting at an initial robot state $s \in S$. If the initial robot state is uncertain and described as a belief b, the expected total reward is then $\int_{s \in S} b(s)\, \alpha_v(s)\, \mathrm{d}s$. We define the *value* of a belief b with respect to a GPG G as the highest expected total reward, starting at any node in G:

$$V_G(b) = \max_{v \in G} \int_{s \in S} b(s)\, \alpha_v(s)\, \mathrm{d}s \tag{3}$$

4. Algorithm

4.1. Overview

Our algorithm computes a GPG as an approximation to an optimal policy. Following the highly successful point-based approach for discrete POMDPs (Pineau et al., 2003; Smith and Simmons, 2005; Kurniawati et al., 2008), we sample a set B of points from the belief space and perform value iteration asynchronously over B.

We start with an initial GPG G_0. For each action $a \in A$, we create a node $v = (a, \kappa)$ in G_0, with $\kappa(o) = v$ for all $o \in O$. Basically, G_0 corresponds to a set of single-action policies.

Algorithm 1 Perform backup of a GPG G at a belief b.

GPG-BACKUP(G, b, M, N, K)

1: **for** each $a \in A$ **do**
2: $\kappa_a \leftarrow$ BUILD-CLASSIFIER(G, b, a, N, K).
3: $v_a \leftarrow (a, \kappa_a)$.
4: $V_a \leftarrow 0$.
5: **for** $i = 1, 2, \ldots, M$ **do**
6: Sample a state s from the distribution $b(s)$.
7: $V_a \leftarrow V_a +$ SIMULATE($G \cup \{v_a\}, v_a, s$).
8: $a^* \leftarrow \arg\max_{a \in A} V_a$.
9: $G' \leftarrow G \cup \{v_{a^*}\}$.
10: **return** G'

Algorithm 2

BUILD-CLASSIFIER(G, b, a, N, K)

1: Sample a set S' of N states from the distribution $b_a(s)$.
2: $\Gamma \leftarrow \emptyset$.
3: **for** each $v \in G$ **do**
4: **for** each $s \in S'$ **do**
5: $\alpha_v(s) \leftarrow 0$
6: **for** $i = 1, 2, \ldots, K$ **do**
7: $\alpha_v(s) \leftarrow \alpha_v(s) +$SIMULATE($G, v, s$)
8: $\alpha_v(s) \leftarrow \alpha_v(s)/K$
9: $\Gamma \leftarrow \Gamma \cup \{\alpha_v\}$.
10: **return** (S', G, Γ).

We then repeat two main steps:

1. Sample a new belief and add it to B (Section 4.3).
2. Choose a subset of beliefs from B. Apply the Bellman backup at each chosen belief b to improve the current GPG G and obtain a new GPG G' (Section 4.2).

In step 2, the Bellman backup equation is

$$HV_G(b) = \max_{a \in A} \left\{ R(b,a) + \gamma \int_{o \in O} p(o|b,a)\, V_G(b_{ao})\, \mathrm{d}o \right\} \tag{4}$$

where H denotes the backup operator and $R(b,a) = \int_{s \in S} R(s,a)\, b(s)\, \mathrm{d}s$. The backup operation looks ahead one step and chooses the action that maximizes the sum of the expected immediate reward $R(b,a)$ and the expected value of the next belief with respect to G. The result is a new GPG G' with one new node $v = (a^*, \kappa_{a^*})$ added to G, where a^* is the maximizer in (4) and κ_{a^*} is the associated edge classifier that maps an observation o to a node in G. To execute the corresponding policy $\pi_{G',v}$, we start at v and take action a^*. After receiving an observation o, we move to the node $\kappa_{a^*}(o)$ in G and follow $\pi_{G,v}$ from then on. It is important to note that our algorithm does not explicitly represent the value function V_G. It performs lazy evaluation of V_G through sampling, whenever needed.

The algorithm maintains upper and lower bounds on the value of the current policy (Section 4.3). It terminates when the approximation error, measured by the gap between the upper and lower bounds, is sufficiently small or maximum planning time is reached.

4.2. GPG backup and classifier construction

To perform backup of G at b using (4), it involves integrating over continuous state and observation spaces. Our algorithm performs the evaluation approximately through sampling (Algorithms 1 and 2).

First, we construct an edge classifier κ_a for each $a \in A$ (Algorithm 1, line 2). By substituting (3) into (4), it is clear that κ_a must map an observation o to the best node $v^* \in G$, implying that v^* maximizes

$$\int_{s \in S} b_{ao}(s)\, \alpha_v(s)\, \mathrm{d}s = \eta \int_{s \in S} b_a(s)\, p(o|s,a)\, \alpha_v(s)\, \mathrm{d}s \tag{5}$$

To evaluate the above integral, we sample a set S' of states according to the distribution b_a. For each $v \in G$ and each sample $s \in S'$, we estimate the value of $\alpha_v(s)$ with a set of Monte Carlo simulations. The procedure SIMULATE(G, v, s) starts at the initial state s and the node $v = (a, \kappa)$ in G. To simulate taking action a in state s, it samples a state s' from the distribution $T(s,a,s') = p(s'|s,a)$ and an observation o from the distribution $Z(s',a,o) = p(o|s',a)$. The process then repeats from the state s' and the node $\kappa(o) \in G$. The simulation length is chosen so that the estimation error is sufficiently small, as a result of the discount factor γ. We collect all the estimates together in

$$\Gamma_{S',G} = \{\alpha_v \mid \alpha_v \colon S' \to \mathbb{R} \text{ and } v \in G\}$$

which basically contains sampled values of a set of α-functions. The tuple $(S', G, \Gamma_{S',G})$ provides all the information necessary for constructing the classifier κ_a:

$$\kappa_a(o) = \arg\max_{v \in G} \sum_{s \in S'} p(o|s)\, \alpha_v(s) \tag{6}$$

in which the sum approximates the integral in (5). The values in Γ serve as the classifier coefficients.

Geometrically, the classifier κ_a maps an observation o to a n-dimensional feature vector $[p(o|s_1), p(o|s_2), \ldots, p(o|s_n)]$ for $s_i \in S'$ and then performs the classification in this feature space. Keep in mind, however, that the feature space is attached to a specific belief b_a, though we do not make the dependency explicit to simplify the notation.

Although a finite number of samples are used to construct κ_a, our analysis provides a uniform error bound on the classifier's performance for any o from a continuous observation space (Theorem 5.1, see below).

After constructing κ_a for each $a \in A$, we perform another set of Monte Carlo simulations to find the best action a^* and the associated classifier κ_{a^*} (Algorithm 1, lines 3–8), resulting in a new GPG node (a^*, κ_{a^*}).

Algorithm 1 summarizes the backup procedure, which takes $\mathcal{O}(|A||G|NK + |A|M)$ simulations. Of the three parameters M, N, and K that control the number of simulations

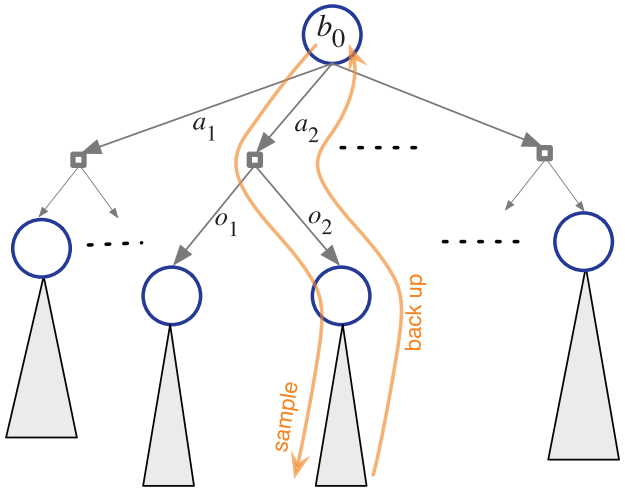


Fig. 2. A belief tree.

and samples, N dominates the running time. It also controls the quality of the computed policy, as it determines the representational complexity and the accuracy of classifiers.

As we perform more backup operations, the increase in the number of GPG nodes and α-functions slows down the computation. Since an α-function in $\Gamma_{S',G}$ is projected to a n-dimensional space, it is just a n-dimensional vector. Standard pruning techniques developed for solving discrete POMDPs can be directly applied to prune these α-functions (Pineau et al., 2003; Smith and Simmons, 2005; Kurniawati et al., 2008). Pruning simplifies the GPG and reduces the number of relevant α-functions when evaluating (6).

4.3. Belief space sampling

There are several approaches to sample the belief space (Pineau et al., 2003; Smith and Simmons, 2005; Kurniawati et al., 2008). We give a very brief description here, as it is not the main focus of this work.

One approach is to spread samples evenly over the belief space $\mathcal{B}$ to cover it and perform synchronous backup at each sampled belief (Pineau et al., 2003). This is practical only if $\mathcal{B}$ is sufficiently small.

Instead, we build a belief tree with an initial belief b_0 as the root (Figure 2). Each node of the tree is labeled with a belief b, together with upper and lower bounds on the optimal value of b. The lower bound is the value of b under the current policy. To compute the upper bound, we relax the model, for example, by assuming the states are fully observable and solving the resulting MDP. Each edge of the tree is labeled with an action–observation pair. If a node b is connected a child b' by an edge (a, o), then $b' = b_{ao}$ (see (1) and (2)).

To sample new beliefs, we repeatedly traverse a single path down the tree until reaching a leaf of the tree. At each internal node b along the path, we choose the action branch with the highest upper bound and choose the observation branch making the largest contribution to the gap between the upper and lower bounds at the root of the tree. These heuristics are designed to bias sampling towards regions that likely lead to improvement in the policy at b_0. Upon reaching a leaf node b, we apply the same criterion to choose a belief b' among all beliefs reachable from b with some action–observation pair and then create a new node for b' as a child of b. To perform backup, we retrace this path back to the root and invoke GPG-BACKUP at each node along the way. See Kurniawati et al. (2008) for more details.

5. Analysis

We now analyze the approximation errors of our algorithm and provide a bound on its performance. The analysis consists of four main steps showing that

1. Given G, b, and a, Algorithm 2 produces a classifier with uniformly bounded error for every observation $o \in O$ with high probability, if sample sizes N and K are sufficiently large.
2. For a given node $v \in G$, the same error integrated over all observations remains bounded, due to the uniform bound from the previous step (Theorem 5.1).
3. Given G and b, the approximation error for a single backup (Algorithm 1) is bounded with high probability if M, N, and K are sufficiently large (Theorem 5.2).
4. Finally, the accumulated approximation error after many backup steps is bounded with high probability, provided the sampled beliefs B approximate $\mathcal{B}$ well (Theorem 5.3).

We then conclude that the computed GPG converges to an optimal policy when M, N, and K are sufficiently large.

In the following analysis, we assume $R(s,a) \leq R_{\max}$ and $Z(s,a,o) = p(o|s,a) \leq P_{\max}$ for all $s \in S$, $a \in A$, and $o \in O$.

Define $V(b,a,o,v)$ to be the expression in (5):

$$V(b,a,o,v) = \int_{s\in S} b_a(s)\, p(o|s,a)\, \alpha_v(s)\, \mathrm{d}s \qquad (7)$$

Given belief b and $a \in A$, the optimal classifier $\kappa^*_{ba}(o)$ produces a node v that maximizes $V(b,a,o,v)$. By (6), Algorithm 2 computes a classifier $\kappa_{ba}(o)$ that produces a node maximizing a sampled approximation of (7)

$$\hat{V}(b,a,o,v) = \frac{1}{|S'|} \sum_{s\in S'} p(o|s,a)\, \alpha_v(s) \qquad (8)$$

We define the error in step 1 as $|V(b,a,o,\kappa_{ba}(o)) - V(b,a,o,\kappa^*_{ba}(o))|$, for a fixed observation o. In step 2, we integrate over all $o \in O$. Define

$$V(b,a,\kappa) = R(b,a) + \gamma \int_{o\in O} V(b,a,o,\kappa(o))\, \mathrm{d}o \qquad (9)$$

The error for step 2 is then $|V(b,a,\kappa_{ba}) - V(b,a,\kappa^*_{ba})|$.

To analyze this error, we need to characterize the complexity of observation functions using a notion called the

covering number. Let X denote a set of points in $\mathbb{R}^n$. Given $\epsilon > 0$, a finite subset $Y \subset \mathbb{R}^n$ *covers* X, if for every $x \in X$, there exists $y \in Y$ with $\|x - y\| < \epsilon$, where $\|x-y\| = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|$. The covering number $\mathcal{C}(\epsilon, X)$ is the minimum number of points required to cover X.

Now consider a set of observation functions, $\mathbf{F}_a = \{f_{a,o} \mid o \in O \text{ and } f_{a,o}(s) = p(o|s,a) \text{ for all } s \in S\}$. Let $\bar{s} = (s_1, s_2, \ldots, s_N)$ be a sequence of N states sampled uniformly at random from S, and $\mathbf{F}_{a|\bar{s}} = \{(f(s_1), f(s_2), \ldots, f(s_N)) \mid f \in \mathbf{F}_a\}$. In our analysis, we bound the complexity of observation functions by the maximum covering number

$$\mathcal{C}_Z(\epsilon, N) = \max_{a \in A} \sup_{\bar{s} \in S^N} \mathcal{C}(\epsilon, \mathbf{F}_{a|\bar{s}}) \tag{10}$$

Let

$$\rho_N(\epsilon, \tau) = \frac{2048(P_{\max}R_{\max})^2}{\epsilon^2(1-\gamma)^2}\left(\ln\left(4|G|\mathcal{C}_Z\left(\frac{\epsilon(1-\gamma)}{32R_{\max}}, N\right)\right) - \ln\tau\right)$$

The following theorem bounds the error between the optimal classifier and the approximate classifier computed. The proofs of all theorems are available in the appendix.

Theorem 5.1. *Given a policy graph G, $b \in \mathcal{B}, a \in A$, a set S' of N states sampled independently from S according to $p(s|b,a)$, and a permissible class*[1] *of observation functions,*

$$p(|V(b,a,\kappa_{ba}) - V(b,a,\kappa^*_{ba})| > \epsilon) \leq \tau \tag{11}$$

for any $\epsilon, \tau > 0$, if $N \geq \rho_N(\epsilon, \tau)$.

The theorem assumes that the observation space $O = [0,1]^n$, i.e., an n-dimensional unit hypercube with the Euclidean metric. To simplify the presentation, we ignore the error of estimating $\alpha(s)$ with K Monte Carlo simulations (Algorithm 2). This error can be made arbitrarily small with sufficiently large K.

For the approximation error to converge to 0, Theorem 5.1 requires that for any $\epsilon > 0$, $\mathcal{C}_Z(\epsilon, N)\,\mathrm{e}^{-C\epsilon^2 N} \to 0$, as $N \to \infty$, for a constant C. This condition is satisfied by many common classes of functions (see Haussler, 1992), e.g., the set of Gaussian observation functions with standard deviation σ:

$$\mathbf{G}_\sigma = \left\{g_o(s) \mid g_o(s) = \mathrm{e}^{-(\frac{s-o}{\sigma})^2}, s \in [0,1],\ o \in [0,1]\right\}$$

Consider two functions g_o and $g_{o+\epsilon\sigma}$ from this set. By the Mean Value Theorem, for any $\epsilon > 0$,

$$\begin{aligned}|g_o(s) - g_{o+\epsilon\sigma}(s)| &= \left|\mathrm{e}^{-(\frac{s-o}{\sigma})^2} - \mathrm{e}^{-(\frac{s-(o+\epsilon\sigma)}{\sigma})^2}\right| \\ &= \left|\epsilon\sigma \cdot \frac{2(s-c)}{\sigma}\mathrm{e}^{-(\frac{s-c}{\sigma})^2}\right|\end{aligned}$$

for some $c \in (o, o+\epsilon\sigma)$. Noting $|2x\mathrm{e}^{-x^2}| \leq 1$ for all x, we have $|g_o(s) - g_{o+\epsilon\sigma}(s)| \leq \epsilon\sigma$ for all $s \in [0,1]$ and thus $\mathcal{C}(\epsilon, N) \leq 1/\epsilon\sigma$. Finally, it is easy to see that $\mathcal{C}(\epsilon, N)\, e^{-C\epsilon^2 N} \leq (1/\epsilon\sigma)\, e^{-C\epsilon^2 N} \to 0$ as $N \to \infty$.

Interestingly, the bound $\mathcal{C}(\epsilon, N) \leq 1/\epsilon\sigma$ seems to suggest that the space of noisy observation functions has a smaller covering number, which reduces the difficulty of planning. This may be counter-intuitive, but true. However, with noisy observations, it may take more steps to gather useful information and act effectively, resulting in increased planning horizon. So the planning problem does not necessarily become easier.

Theorem 5.1 bounds the error of a classifier computed for a given action. Algorithm 1 then performs backup and constructs the best policy graph node by selecting from the $|A|$ candidate action–classifier combinations by Monte Carlo simulation. Simulation introduces additional error in the backup.

In step 3, we bound the error of approximate backup: $|HV(b) - \hat{H}V(b)|$, where $HV(b)$ denotes the exact backup using (4) and $\hat{H}V(b)$ denotes the approximate backup produced by Algorithm 1. Let

$$\rho_M(\epsilon, \tau) = \frac{(P_{\max}R_{\max})^2}{2\epsilon^2(1-\gamma)^2}\left(\ln 2 - \ln\tau\right)$$

The following theorem bounds the error of a single step of point-based backup, as computed in Algorithm 1. The error has two parts, one part from errors in steps 1 and 2 and the other part from Monte Carlo evaluation of the classifiers (Algorithm 1, lines 5–7).

Theorem 5.2. *Given a policy graph G, a belief $b \in \mathcal{B}$, and a permissible class of observation functions,* GPG-BACKUP(G, b, M, N, K) *produces an improved policy graph such that for any $\epsilon, \tau > 0$,*

$$p(|HV(b) - \hat{H}V(b)| > \epsilon) \leq \tau$$

if $N \geq \rho_N(\epsilon/5, \tau/2|A|)$ and $M \geq \rho_M(\epsilon/5, \tau/2|A|)$.

Finally, in step 4, we combine all sources of error. We analyze the case where the algorithm runs GPG-BACKUP on a sampled belief set $B \subset \mathcal{B}$ synchronously for t iterations. Let $\delta_B = \sup_{b \in \mathcal{B}} \min_{b' \in B} \int_s |b(s) - b'(s)|ds$ denote the largest distance for any belief in $\mathcal{B}$ to its nearest point in the set B where backup is performed. We bound the approximation error between the value function V_t in the tth iteration and the optimal value function V^*. The result is similar to that for the MCVI algorithm (Bai et al., 2011).

By bounding the error propagation across the backup iterations, we obtain the following theorem:

Theorem 5.3. *Given a POMDP with a permissible class of observation functions, choose $N \geq \rho_N(\epsilon/5, \tau/2|A||B|t)$, $M \geq \rho_M(\epsilon/5, \tau/2|A||B|t)$, and perform t iterations of backup over a sampled belief set $B \subset \mathcal{B}$ synchronously. Then for every $b \in \mathcal{B}$ and every $\epsilon, \tau > 0$,*

$$|V^*(b) - V_t(b)| \leq \frac{\epsilon}{1-\gamma} + \frac{2R_{\max}\delta_B}{(1-\gamma)^2} + \frac{2\gamma^t R_{\max}}{1-\gamma}$$

with probability at least $1 - \tau$,

Theorem 5.3 shows that the approximation error comes from three main sources: GPG-BACKUP, the approximation

of the belief space by a finite set of belief, and the finite number of back-up iterations. The error from GPG-BACKUP can be reduced by increasing the number of samples in the Monte Carlo sampling. The approximation of the belief space can be improved by using more belief points, and error decreases exponentially with the number of back-up iterations.

In general, asynchronous backup, which is used in our experiments, is more efficient than synchronous backup. The analysis is similar, but involves more technical details.

6. Experiments

We evaluated our algorithm on three tasks. In *Linear-quadratic-Gaussian (LQG) control*, we can solve for the optimal policy analytically and use it to calibrate the performance of the new algorithm (Section 6.1). In *Intersection navigation*, we examine various aspects of our algorithm and also compare it with a well-established alternative algorithm, MC-POMDP (Thrun, 2000) (Section 6.2). Finally, in *Acrobot with model uncertainty*, we use the algorithm for Bayesian reinforcement learning in order to handle model uncertainty (Section 6.3).

6.1. LQG control

An LQG system is basically a POMDP with linear system dynamics, Gaussian noise, and a quadratic reward function. Our simple LQG problem is given by

$$x_t = -x_{t-1} + u_{t-1} + w_t$$
$$y_t = x_t + v_t$$

where x_t, u_t, and y_t are the state, the action, and the observation at time t, and $w_t \sim \mathcal{N}(0, 10)$ and $v_t \sim \mathcal{N}(0, 10)$ represent zero-mean Gaussian system noise and observation noise. The goal is to minimize the infinite-horizon average cost $C = \lim_{N\to\infty} \frac{1}{N} \sum_{t=0}^{N} \left(x_t^2 + u_t^2\right)$. A linear feedback policy has the form $u_t = \lambda \hat{x}_t$, where $\hat{x}_t$ is the estimated mean state at time t and λ is the control gain. The optimal policy has $\lambda^* = 0.618$.

To recast the problem as a POMDP, we choose 17 equally spaced actions in the range $[-24, 24]$ and set the discount factor to 0.99 to approximate the infinite-horizon cost function. The state space and the observation remain unchanged. The computed policy contains 1024 GPG nodes.

We evaluated the POMDP policy and several linear feedback policies with different λ by performing 10,000 simulations for each. Figure 3 shows their costs and behaviors. The POMDP policy computation neither exploits the linearity of system dynamic nor possesses prior knowledge of the linear form of the optimal policy. Nevertheless, it discovers a policy that has a roughly linear form, up to action discretization and has a cost close to the minimum.

Table 1 shows the size of policies computed for this and the other two tasks, as well as their online execution speed and offline planning time. When multiple policies were computed, Table 1 shows the results for the worst case. The running times were obtained on a PC with a 2.83 GHz CPU and 4 GB memory. For offline planning, the main performance measures are policy quality and online execution speed. Offline planning time is a secondary issue, provided it is practical. The results confirm one main benefit of the policy graph representation, very fast policy execution, which is important for applications such as aircraft collision avoidance (Bai et al., 2011).

Table 1. The size, execution speed and planning time of computed policies.

Task	N	$\|G\|$	Speed (KHz)	Time (h)
LQG	50	1024	25.5	22.8
Navigation	500	20	1.7	2.7
Acrobot	100	825	14.8	23.6

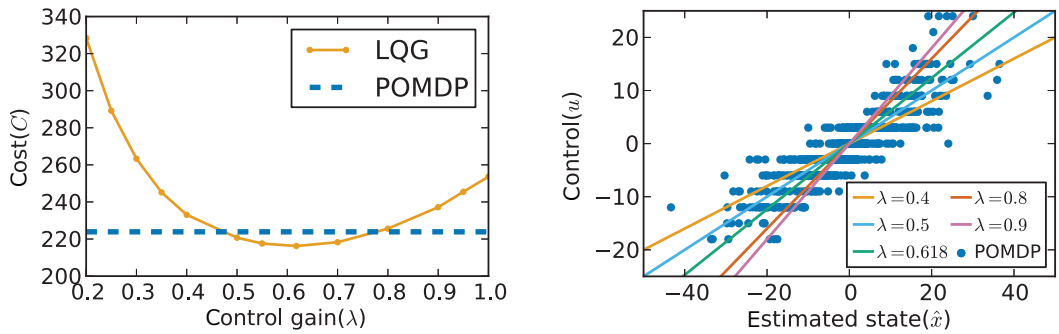
N: number of samples for each α-function in the classifier
$|G|$: number of GPG nodes
Speed: policy execution speed in the number of GPG nodes processed per second
Time: time required for computing the policy offline

The aim of this evaluation on LQG control is to calibrate the algorithm's performance on a task with a known optimal solution. Our algorithm is not ideal for LQG control, in comparison with the analytical method. To reduce the approximation error, we may increase the number of discretized actions, but the offline planning time also increases as a result. However, some tasks have a small discrete action set, e.g., bang-bang control or mobile robot navigation, which often uses eight discretized actions. These tasks are better suited for our algorithm.

6.2. Intersection navigation

Recall the example in Figure 1. The autonomous vehicle R, in blue, stops at the intersection and waits for the other vehicle R' to clear before proceeding. R' cannot be localized accurately, as the measurements from R's proximity sensor are noisy. R wants to go through the intersection as fast as possible, while maintaining safety. So it must carefully balance exploration and exploitation by hedging against the noisy observations.

6.2.1. Observation modeling. Our main objective here is to investigate the effect of observation modeling on the policy and not necessarily a high-fidelity model for vehicle navigation. We make a few simplifications to stay on the main issue. Assume that the vehicles move within a lane. The state space $S = [-10, 10]$ encodes the position of R', which is sufficient to decide the action of R. The initial belief on the position of R' is uniform over $[-10, 0]$. R has two actions. WAIT keeps R stopped. GO moves R forward through the intersection. There is no emergency stop in our



(a) (b)

Fig. 3. Comparing the LQG POMDP policy and the linear feedback policies. (a) Policy costs estimated from 10,000 simulations. The dashed line indicates the cost of the POMDP policy. The solid curve plots the cost of linear feedback policies with different control gain λ. The standard errors of the estimated costs are all less than 1 and not visible on the plots. (b) Behaviors of the POMDP policy and several linear feedback policies. For the POMDP policy, we plot the mean of the belief and the action associated with each policy graph node.

simple model. If R goes through the intersection successfully, it gets a reward 1. If a collision occurs, it gets a large penalty R_{p}. Hence the reward function

$$R(s,a)=\begin{cases}0 & \text{if } a=\text{WAIT}\\ 1 & \text{if } a=\text{GO and } s\notin[-1,1]\\ -R_{\text{p}} & \text{if } a=\text{GO and } s\in[-1,1]\end{cases}$$

We tested two observation models. The first one follows the standard beam model for proximity sensing (Thrun et al., 2005). An observation $o=(h_1,h_2,\ldots,h_{30})$ consists of readings along 30 beams equally spaced over 160° field of view. We quantize each reading h_i into a binary value: $h_i=1$ indicates that the ith beam hits R', and $h_i=0$ indicates that the beam does not. There are false positives, due to unexpected obstacles, and false negatives, due to, e.g.,, total reflection or glass. Let h_i^* denote true value for the ith beam. Our test uses a high-noise environment with $p(h_i=1|h_i^*=1)=0.7$ and $p(h_i=0|h_i^*=0)=0.9$. The beam model assumes that readings along the beams are independent: $p(o|s,a)=\prod_{i=1}^{30}p(h_i|s,a)$ (Thrun et al., 2005).

The main difficulty with the beam model above is the high-dimensional observation space. With 2^{30} observations, no POMDP algorithm can cope. To avoid reasoning directly with the high dimensional observation space, our second model calculates the maximum-likelihood (ML) location x of R' from $o=(h_1,h_2,\ldots,h_{30})$, with x discretized into bins $X=\{-10,-9,\ldots,9,10\}$. Specifically, we have $x=\zeta(o)=\arg\max_{x\in X}p(x|o)=\arg\max_{x\in X}p(o|x)p(x)/p(o)$, where the prior $p(x)$ is uniform over X. We then use x as the observation for the POMDP model, resulting in only 21 observations in total. This drastic reduction in the number of observations, however, comes at a cost, as we see next.

For the beam model, our new algorithm was the only option available to solve the resulting POMDP. For the ML model, we used the MCVI algorithm (Bai et al., 2010), which is specialized for continuous-state, discrete-observation POMDPs.

Table 2. Performance comparison of POMDP policies with two different observation models for intersection navigation.

R_{F}	Observation model	$\lvert G\rvert$	Time	Accident rate
10	Beam	14	2.61 ± 0.0095	0.0029 ± 0.00053
	ML	127	4.27 ± 0.0012	0.0093 ± 0.00010
100	Beam	20	3.12 ± 0.014	0.0009 ± 0.00030
	ML	83	9.22 ± 0.0039	0.0028 ± 0.00005
1000	Beam	18	5.03 ± 0.030	0.0002 ± 0.00014
	ML	80	12.84 ± 0.00036	0.0002 ± 0.00001

$|G|$: number of GPG nodes
Time: time to cross the intersection

We solved several POMDP models with different values for the collision penalty R_{p}. Each policy was computed with a maximum of 3 hours of offline planning time and evaluated with 1,000,000 simulations. The results are reported in Table 2. Clearly the new algorithm with the beam model achieved consistently better results with lower accident rate and faster crossing time.

The performance gap results from *information loss* during the maximum-likelihood calculation. To understand this, consider a particular state $s=-4$ and choose two high-probability beam observations o_1 and o_2 from $p(o|s=-4)$ such that $\zeta(o_1)=\zeta(o_2)=x$. That is, we have the same ML location estimate for both o_1 and o_2 and cannot differentiate them in the ML observation model. Now consider the posterior beliefs b_1, b_2, and b (Figure 4) for o_1, o_2, and x in their respective models, after R executes a single WAIT action and receives the observation. The posterior beliefs all have the same general shape. However, a careful comparison of b_1 and b_2 reveals a small secondary peak for b_2 in the region $[-1,1]$, indicating the likely presence of R' in the intersection. A good policy must handle

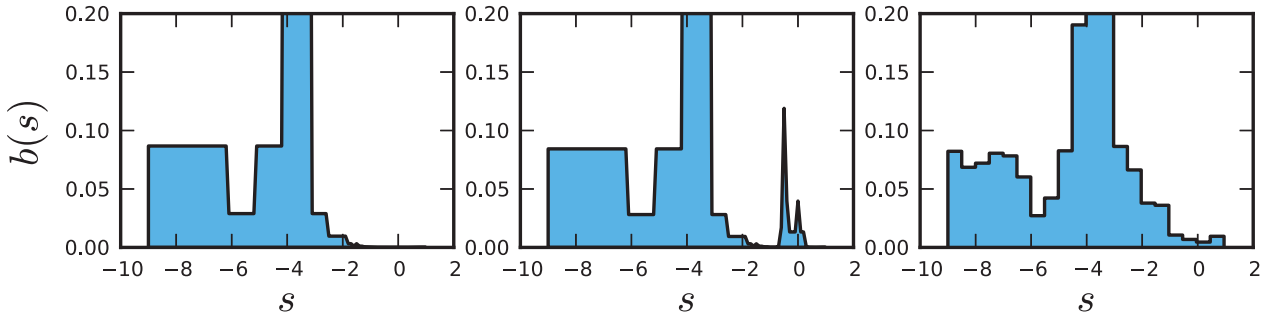


Fig. 4. Posterior beliefs b_1, b_2, and b, from left to right.

Table 3. Performance with increasing sensor noise.

P_f	$\|G\|$	Average reward	Time	Accident rate
0.1	20	669.1 ± 20.5	3.12 ± 0.014	0.0009 ± 0.00030
0.2	38	440.9 ± 12.2	4.73 ± 0.010	0.0027 ± 0.00002
0.4	16	272.2 ± 4.5	11.48 ± 0.006	0.0006 ± 0.00008

this low-probability, but critical event properly. Otherwise the vehicle will either get into an accident or unnecessarily wait. However, the ML model provides the same observation x whether it is actually o_1 or o_2, and the posterior belief b does not have a secondary peak. In general, there are 2^{30} beam observations, but only 21 ML observations. Many beam observations map into the same ML observation and cannot be differentiated in the ML model. The loss of information is a main contributor of the performance gap and cannot be resolved even if the ML model uses a more finely discretized observation space.

6.2.2. The effect of sensor noise. Next, we investigate the effect of sensor noise on the policy behavior. We computed and evaluated policies for beam observation models with different false positive probability $P_f = p(h_i = 1|h_i^* = 0)$. The results are shown in Table 3. As expected, higher noise decreases the average total reward and increases the time for the vehicle to cross the intersection. At $P_f = 0.2$, both the crossing time and the accident rate are much larger, compared with those at $P_f = 0.1$. The policy size $|G|$ is also larger, indicating that the model is more difficult and more complex decision making is required. Somewhat surprisingly, at $P_f = 0.4$, the accident rate and the policy size are both small. The reason is that the observations are too noisy to be exploited for effective decision making. This results in a simple, but very conservative policy, which merely waits for a sufficiently long time in order to cross the intersection.

6.2.3. Scalability. We now look at the scalability of our algorithm with large state space, observation space, or action space.

Theorem 5.3 shows that the overall approximate error does not depend explicitly on the dimension of the state

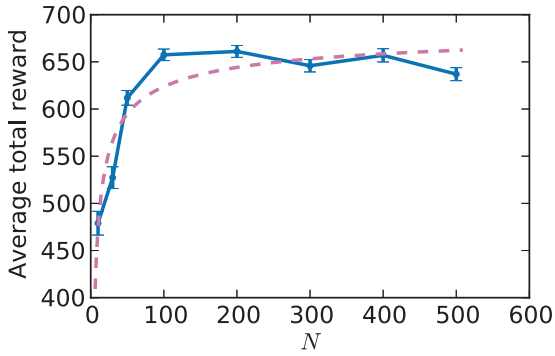


Fig. 5. Empirical convergence rates with respect to N. A solid line indicates the average total reward obtained from 100,000 simulations. A dashed line indicates the best-fit curve $a/\sqrt{N} + b$.

space. Although the intersection navigation task has an 1-dimensional state space, the acrobot swing-up task in the next subsection has a 5-dimensional state space, and our algorithm easily scales up. Theorem 5.1 further indicated that when the covering number $\mathcal{C}_Z(\epsilon, N)$ grows slowly with ϵ and N, the approximation error resulting from state-space sampling decreases roughly at the rate of $\mathcal{O}(1/\sqrt{N})$. Figure 5 shows the empirical convergence rate for the intersection navigation task, and it correlates well with the theoretical analysis. Indeed, probabilistic sampling is a powerful tool for handling high-dimensional state spaces, as its success in robot motion planning amply demonstrates (Choset et al., 2005).

We could use the above error bound to set the parameter N in the algorithm, but this is often overly conservative. In our experiments, we performed several trials, starting with a small N and increasing it until the performance improvement is insignificant. More generally, we can set N adaptively in each backup operation by estimating the sample variance.

Our algorithm also scales up well with large observation space. The beam model contains 2^{30} observations, while our algorithm uses a relatively small number of sampled states and observations to construct a GPG and the associated classifiers. How do these classifiers perform on observations not explicitly sampled during the GPG construction? To shed some light on this question, we took a computed

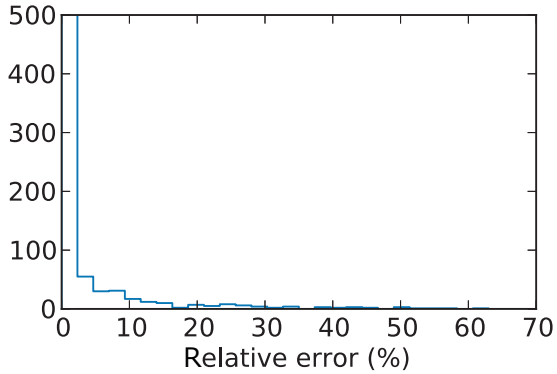


Fig. 6. A histogram of classifier errors.

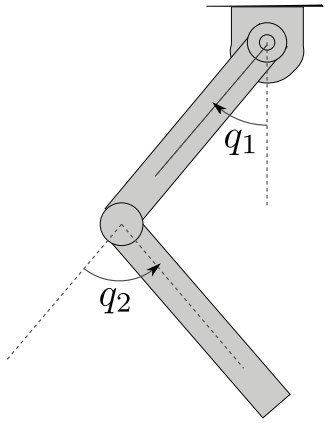


Fig. 7. The acrobot is a two-link articulated robot actuated only at the joint connecting the two links and thus unactuated. It resembles a gymnast swing on a high bar. In the standard acrobot, each link has mass $m = 1.0$ and length $\ell = 1.0$.

GPG and examined the classifier associated with its start node $v = (a, \kappa)$. We evaluated the performance of κ at the initial belief b by sampling 1000 observations from the distribution $p(o|b, a)$. For each sampled observation o, we estimated the value of the policy $\pi_{G,\kappa(o)}$ at b_{ao} by simulation. We compared this value to $V(b_{ao})$, obtained by performing a sufficiently large number of simulations. The difference defines the error of the classifier κ for the observation o. Figure 6 shows a histogram of errors for the 1000 sampled observations. The occurrence of large errors decays almost exponentially. Although there is a somewhat long tail, most errors are very small.

Our algorithm builds a search tree in the belief space. Large action space poses the most significant challenge, as it increases the branching factor of the search tree. For this reason, we maintain upper and lower bound at each belief tree node and apply the branch-and-bound technique to prune suboptimal parts of the tree (Kurniawati et al., 2008).

6.2.4. Comparison with MC-POMDP. We also compared our algorithm with MC-POMDP, a well-established earlier algorithm (Thrun, 2000). Both methods use particle-based belief representation. However, MC-POMDP represents a policy by its value function over a set B of sampled beliefs. To calculate the value at a belief $b \notin B$, MC-POMDP finds k-nearest neighbors of b in B according to KL-divergence as the distance function, and then estimates the value of b using distance-weighted interpolation.

We applied both our algorithm and MC-POMDP on intersection navigation POMDPs with the beam observation model. We solved the POMDPs with three different values for the collision penalty R_p and then evaluated the resulting policies through simulation. The results are reported in Table 4. For all three models, our algorithm performed substantially better than MC-POMDP. MC-POMDP policy computation was much faster. The computed policies, however, have poor quality with overly aggressive behavior resulting in accident rate 3–10 times higher.

The two algorithms differ significantly in how they represent the policy and estimate the value $V(b)$. The difference is one main contributor to the performance gap. Our algorithm represents the policy as a GPG and estimates $V(b)$ by simulating the policy. MC-POMDP represents the value function V explicitly over B and estimates $V(b)$ by nearest-neighbor interpolation. In general, the belief space $\mathcal{B}$ is a very high-dimensional space. The distance between a belief $b \in \mathcal{B}$ and its nearest neighbors is likely far, causing potentially large interpolation error. Furthermore, in a high-dimensional space, we may need a large number of neighbors for effective interpolation, with increased computational cost.

Another advantage of our approach is fast online policy execution. While GPG policies were executed at the rate of about 1700 actions per second, MC-POMDP policies were much slower to execute, roughly at the rate of 1 action per second, because interpolating the value function incurs high computational cost. Computing the KL divergence takes $O(n^2)$ time, where n is the number of particles representing the beliefs. Sometimes large n is required for the particle filter to converge.

6.3. Acrobot with model uncertainty

Acrobot is a well-studied underactuated system (Figure 7). In the swing-up task, the acrobot must get its tip above the height 1.95 and achieve the almost fully stand-up configuration. Our acrobot variant assumes that a key model parameter, the mass m of the acrobot's second link, is not known exactly, thus introducing model uncertainty.

This task is particularly challenging, because the acrobot's dynamics are sensitive to m. An open-loop control policy that successfully swings up an acrobot with $m = 1.0$ fails completely on an acrobot with $m = 1.01$ (Bai et al., 2013). To succeed, a control policy must simultaneously learn the acrobot's unknown parameter and plan the

Table 4. Performance comparison with MC-POMDP.

R_p	Algorithm	Average reward	Time	Accident rate
10	GPG	738.9 ± 4.6	2.61 ± 0.0095	0.0029 ± 0.00053
	MC-POMDP	697.4 ± 7.8	2.52 ± 0.0081	0.0086 ± 0.00092
100	GPG	669.1 ± 20.5	3.12 ± 0.014	0.0009 ± 0.00030
	MC-POMDP	532.6 ± 37.1	3.06 ± 0.010	0.0028 ± 0.00053
1000	GPG	489.8 ± 90.3	5.03 ± 0.030	0.0002 ± 0.00014
	MC-POMDP	-645.4 ± 299.5	3.65 ± 0.011	0.0020 ± 0.00044

actions under an uncertain model. We apply the model-based Bayesian reinforcement approach (Duff, 2002) and formulate the task as a POMDP.

The POMDP state is $s = (q_1, q_2, \dot{q}_1, \dot{q}_2, m)$, where $q_1, q_2 \in [-\pi, \pi]$, $\dot{q}_1 \in [-4\pi, 4\pi]$, and $\dot{q}_2 \in [-9\pi, 9\pi]$ represent the joint angels and the angular velocities of the two links (Figure 7). All the state variables, including m, are continuous. The acrobot can apply a torque $\tau \in \{-1, 0, +1\}$ at the elbow joint. We use the system dynamics equations in Sutton and Barto (1998) and assume no action noise. An observation consists of the two joint-angle values under Gaussian noise with variance 0.1. The angular velocities $\dot{q}_1$ and $\dot{q}_2$ and the model parameter m cannot be observed directly. The reward is 10 if the acrobot reaches the specified height, and 0 for other states and actions. The discount factor is 0.95. The initial belief for m is uniform over $[0.95, 1.05]$.

Our new algorithm can solve this POMDP without a priori discretization of the state and observation spaces. State-space discretization is difficult in general, because it introduces modeling errors that are difficult to quantify. It is exacerbated here by the acrobot's sensitive non-linear dynamics. Observation space discretization is also difficult, as it may lose information and degrade the quality of the computed policy (Section 6.2). We will see further evidence of the difficulty here.

We solved the acrobot POMDP with different values for the sampling parameter N and evaluated each resulting policy with $10,000$ simulations. For comparison, we also evaluated an oracle policy, for which the model parameter m and the system state are fully observable. Table 5 shows that a relatively small N is sufficient to produce a good policy in this case. Increasing N consistently improves the results, as N controls the accuracy of edge classifiers and, in turn, policies.

Figure 8 visualizes a particular edge classifier κ from the policy with $N = 100$. Each point in the plot represents an observation o collected from a simulation trace going through κ. The point is colored according to the output GPG node $\kappa(o)$. The observations fall into six classes, with very different sizes. The smallest one has width about 0.001. To obtain the same result with a regular discretization of the observation space, we have to use very fine resolution, roughly 0.001, in order to capture the small classes sandwiched between large ones. The resulting $1,000,000$ observations are beyond the reach of any discrete POMDP algorithm.

Table 5. The performance of acrobot POMDP policies with different values of sample parameter N.

Policy	N	$\lvert G \rvert$	Average height
oracle	-	-	1.97 ± 0.0000
POMDP	100	825	1.90 ± 0.0021
	50	871	1.87 ± 0.0027
	25	591	1.86 ± 0.0029
	10	815	1.84 ± 0.0031
	5	123	1.78 ± 0.0036
	3	244	1.66 ± 0.0144

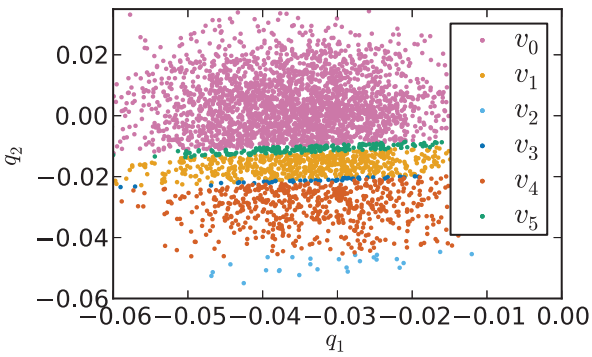


Fig. 8. Visualization of an edge classifier κ. Each point is a sampled observation o and colored according to the output GPG node $\kappa(o)$.

This example confirms again the difficulty of observation discretization. In practice, some observation discretization or aggregation is probably necessary. However, a priori discretization without a good understanding of its effect should be avoided. The new algorithm helps to reduce the need for aggressive discretization.

7. Conclusion

This paper presents a new algorithm for solving POMDPs with continuous states and observations. These continuous models are natural for robotic tasks that require integrated perception and planning. We provide experimental results demonstrating the potential of this new algorithm for robot

planning and learning under uncertainty. We also provide a theoretical analysis on the convergence of the algorithm.

Our algorithm uses sampling instead of fixed discretization to handle continuous state and observation spaces. Sampling opens up a range of new opportunities to scale up the algorithm for complex robot planning and learning tasks. Currently our algorithm performs a huge number of Monte Carlo simulations. It often takes hours to compute a policy for tasks at a scale similar to those in our experiments. Some of these simulations are redundant, and we are looking for ways to *reuse* simulations. We will apply standard techniques to prune the dominated α-functions and the GPG, thereby reducing the simulations needed. We will also explore parallelization techniques, such as general purpose GPU computing (Lee and Kim, 2013), to increase the practical performance of the algorithm.

Funding

This work was supported by Singapore Ministry of Education Academic Research Fund (grant number 2010-T2-2-071), the Singapore National Research Foundation through the SMART IRG program, and the US Air Force Research Laboratory (grant number FA2386-12-1-4031).

Note

1. Measurability conditions that usually hold in practice (see Haussler, 1992).

References

Bagnell J, Kakade S, Ng A, et al. (2003) Policy search by dynamic programming. In: *Advances in Neural Information Processing Systems (NIPS)*, Vol. 16. Vancouver, Canada, 9–11 December 2003. Cambridge, MA: The MIT Press.

Bai H, Hsu D, Kochenderfer MJ, et al. (2011) Unmanned aircraft collision avoidance using continuous-state POMDPs. In: *Proceedings of Robotics: Science and systems*, Los Angeles, CA, USA, 27 June-1 July 2011.

Bai H, Hsu D and Lee W (2013) Planning how to learn. In: *Proceedings of the IEEE International conference on robotics and automation*, Karlsruhe, Germany, 6–10 May 2013, pp. 2853–2859.

Bai H, Hsu D, Lee W, et al. (2010) Monte Carlo value iteration for continuous-state POMDPs. In: Hsu D et al. (eds) *Algorithmic Foundations of Robotics IX—Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Berlin; Heidelberg: Springer, pp. 175–191.

Brechtel S, Gindele T and Dillmann R (2013) Solving continuous POMDPs: Value iteration with incremental learning of an efficient space representation. In: *Proceedings of the 30th International conference on machine learning*, Atlanta, GA, USA, 16–21 June 2013, pp. 370–378.

Brooks A, Makarendo A, Williams S and Durrant-Whyte H (2006) Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems* 54(11): 887–897.

Choset H, Lynch K, Hutchinson S, et al. (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: The MIT Press.

Davidson J and Hutchinson S (2008) Hyper-particle filtering for stochastic systems. In: *Proceedings of the IEEE International conference on robotics and automation*, Pasadena, CA, USA, 19–23 May 2008, pp. 2770–2777.

Duff M (2002) *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD Thesis, University of Massachusetts Amherst, MA, USA.

Erez T and Smart W (2010) A scalable method for solving high-dimensional continuous POMDPs using local approximation. In: *Proceedings of the 26th conference on uncertainty in artificial intelligence*, Catalina Island, CA, USA, 8–11 July 2010.

Hansen E (1998) Solving POMDPs by searching in policy space. In: *Proceedings of the fourteenth conference on uncertainty in artificial intelligence*, Madison, WI, USA, 24–26 July 1998 Madison, WI, pp. 211–219.

Hauser K (2010) Randomized belief-space replanning in partially-observable continuous spaces. In: *Algorithmic Foundations of Robotics IX—Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Berlin; Heidelberg: Springer, pp. 193–209.

Haussler D (1992) Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation* 100(1): 78–150.

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

Hoey J and Poupart P (2005) Solving POMDPs with continuous or large discrete observation spaces. In: *Proceedings of the International joint conference on artificial intelligence*, Edinburgh, UK, 30 July–5 August 2005, pp. 1332–1338.

Hsiao K, Kaelbling L and Lozano-Pérez T (2007) Grasping POMDPs. In: *Proceedings of the IEEE International conference on robotics and automation*, Roma, Italy, 10–14 April 2007, pp. 4685–4692.

Kurniawati H, Hsu D and Lee W (2008) SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Proceedings of robotics: science and systems*, Zurich, Switzerland, 25–28 June 2008.

Lagoudakis M and Parr R (2003) Reinforcement learning as classification: Leveraging modern classifiers. In: *Proceedings of the International conference on machine learning*, Washington DC, USA, 21–24 August 2003, pp. 424–431.

Lee T and Kim YJ (2013) GPU-based motion planning under uncertainties using POMDP. In: *Proceedings of the IEEE International conference on robotics and automation*, Karlsruhe, Germany, 6–10 May 2013, pp. 4576–4581.

Meuleau N, Peshkin L, Kim K, et al. (1999) Learning finite-state controllers for partially observable environments. In: *Proceedings of the fifteenth conference on uncertainty in artificial intelligence*, Stockholm, Sweden, 30 July–1 August 1999, pp. 427–436.

Pineau J, Gordon G and Thrun S (2003) Point-based value iteration: An anytime algorithm for POMDPs. In: *Proceedings of the International joint conference on artificial intelligence*, Acapulco, Mexico, 9–15 August 2003, pp. 477–484.

Platt R Jr, Tedrake R, Kaelbling L, et al. (2010) Belief space planning assuming maximum likelihood observations. In: *Proceedings of robotics: science and systems*, Zaragoza, Spain, 27–30 June 2010.

Porta J, Vlassis N, Spaan M, et al. (2006) Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7: 2329–2367.

Poupart P and Boutilier C (2003) Bounded finite state controllers. In: *Advances in Neural Information Processing Systems (NIPS)*, Vol. 16. Cambridge, MA: The MIT Press, pp. 823–830.

Powell W (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ: Wiley-Interscience.

Prentice S and Roy N (2007) The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In: *Proceedings of the International symposium on robotics research*.

Ross S, Pineau J, Paquet S, et al. (2008) Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32(1): 663–704.

Roy N and Thrun S (1999) Coastal navigation with mobile robots. In: *Advances in Neural Information Processing Systems (NIPS)*, Vol. 12. Cambridge, MA: The MIT Press, pp. 1043–1049.

Smith T and Simmons R (2005) Point-based POMDP algorithms: Improved analysis and implementation. In: *Proceedings of the twenty-first conference on uncertainty in artificial intelligence*, Edinburgh, UK, 26–29 July 2005, pp. 542–547.

Sutton RS and Barto AG (1998) *Reinforcement Learning*. Cambridge, MA: MIT Press.

Thrun S (2000) Monte Carlo POMDPs. In: *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: The MIT Press.

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

Van den Berg J, Abbeel P and Goldberg K (2010) LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In: *Proceedings of robotics: science and systems*, Zaragoza, Spain, 27–30 June 2010.

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.

Wang Y, Won K, Hsu D, et al. (2012) Monte Carlo Bayesian reinforcement learning. In: *Proceedings of the International conference on machine learning*, Edinburgh, UK, 26 June–1 July 2012.

Appendix: Proofs of theorems

To prove the Theorems 5.1–5.3, we need the following result (Haussler, 1992), which bounds the error in the sampled means of a set of functions.

Theorem A.1. *Let* $\mathbf{F}$ *be a permissible class of bounded functions over a set S, with* $0 \le f(s) \le B$ *for all* $f \in \mathbf{F}$ *and* $s \in S$. *Let* $\bar{s} = (s_1, \ldots, s_n)$ *be a sequence of n points drawn independently at random according to any distribution over S and*

$$\mathbf{F}_{|\bar{s}} = \{(f(s_1), \ldots, f(s_n)) \mid f \in \mathbf{F}\}$$

Then for any $\epsilon > 0$,

$$p\big(\exists f \in \mathbf{F}\colon |\hat{\mathrm{E}}_{\bar{s}}(f) - \mathrm{E}(f)| > \epsilon\big) \le 4\mathrm{E}\big(\mathcal{C}(\epsilon/16, \mathbf{F}_{|\bar{s}})\big)\mathrm{e}^{-\epsilon^2 n/128B^2}$$

where $\hat{\mathrm{E}}_{\bar{s}} = \sum_{i=1}^{n} f(s_i)/n$.

We now apply Theorem A.1 to show that the error $|V(b,a,o,\kappa_{ba}(o)) - V(b,a,o,\kappa^*_{ba}(o))|$ is bounded uniformly for all $o \in O$ with high probability (Theorem 5.1). We start by introducing the function classes under consideration. For every $a \in A$, let

$$\mathbf{F}_a = \{f_{a,o} \mid o \in O \text{ and } f_{a,o}(s) = p(o|s,a) \text{ for all } s \in S\}$$

which contains all observation functions for a fixed $a \in A$. Define the covering number

$$\mathcal{C}_a(\epsilon, N) = \sup_{\bar{s} \in S^N} \mathcal{C}\big(\epsilon, \mathbf{F}_{a|\bar{s}}\big)$$

where $\mathbf{F}_{a|\bar{s}}$ consists of a set of vectors, each obtained by evaluating a function $f \in \mathbf{F}_a$ over a sequence $\bar{s}$ of N points sampled from S according to the distribution $b_a(s)$. Similarly, for $a \in A$ and $v \in G$, let

$$\mathbf{F}_{a,v} = \{f_{a,o,v} \mid o \in O \text{ and } f_{a,o,v}(s) = p(o|s,a)\,\alpha_v(s) \text{ for all } s \in S\}$$

and $\mathbf{F}_{a,G} = \bigcup_{v \in G} \mathbf{F}_{a,v}$. Define

$$\mathcal{C}_{a,G}(\epsilon, N) = \sup_{\bar{s} \in S^N} \mathcal{C}\big(\epsilon, \mathbf{F}_{a,G|\bar{s}}\big)$$

Our first lemma establishes the the relationship between $\mathbf{F}_a$ and $\mathbf{F}_{a,G}$ in terms of covering numbers.

Lemma A.2. $\mathcal{C}_{a,G}(\epsilon, N) \le |G|\,\mathcal{C}_a(\frac{\epsilon(1-\gamma)}{R_{\max}}, N)$

Proof. For a fixed α-vector α, consider any two vectors $x, x' \in \mathbf{F}_{a|\bar{s}}$ and the two corresponding vectors $y, y' \in \mathbf{F}_{a,v|\bar{s}}$, where $y_i = x_i\alpha_i$ and $y'_i = x'_i\alpha_i$ for $i = 1, 2, \ldots, N$. Recall the covering metric defined earlier: $\|x - x'\| = \frac{1}{N}\sum_{i=1}^{N} |x_i - x'_i|$. We have

$$\begin{aligned}\|y - y'\| &= \frac{1}{N}\sum_{i=1}^{N} |x_i\alpha_i - x'_i\alpha_i| \\ &\le \Big(\max_{i=1,\ldots,N} |\alpha_i|\Big)\frac{1}{N}\sum_{i=1}^{m} |x_i - x'_i| \\ &\le \frac{R_{\max}}{1-\gamma}\|x - x'\|\end{aligned}$$

Therefore, if a point set covers $\mathbf{F}_{a|\bar{s}}$ at the scale ϵ, a corresponding point set covers $\mathbf{F}_{a,v|\bar{s}}$ at the scale $(R_{\max}/(1-\gamma))\epsilon$. Since $\mathbf{F}_{a,G|\bar{s}} = \bigcup_{v \in G} \mathbf{F}_{a,v|\bar{s}}$, we get

$$\mathcal{C}_{a,G}(\epsilon, N) \le |G|\,\mathcal{C}_a(\frac{\epsilon(1-\gamma)}{R_{\max}}, N)$$

where $|G|$ is the number of nodes in G. □

The next lemma bounds the error in the approximate maximum of a sequence of numbers.

Lemma A.3. *Let* $x_1, x_2, \ldots, x_n$ *be a set of real numbers, and* $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n$ *be their approximations with* $|x_i - \hat{x}_i| < \epsilon$ *for* $i = 1, 2, \ldots, n$. *Let* $j = \arg\max_{i=1,\ldots,n} x_i$ *and* $k = \arg\max_{i=1,\ldots,n} \hat{x}_i$. *Then* $|x_j - x_k| < 2\epsilon$.

Proof.

$$\begin{aligned}|x_j - x_k| &= x_j - \hat{x}_j + \hat{x}_j - x_k \\ &\le x_j - \hat{x}_j + \hat{x}_k - x_k \\ &\le |x_j - \hat{x}_j| + |\hat{x}_k - x_k| \\ &< 2\epsilon\end{aligned}$$

□

Proof (Theorem 5.1). The expected value of a function $f_{a,o,v}$ with respect to belief b_a is

$$\begin{aligned}\mathrm{E}(f_{a,o,v}) &= \int_{s\in S} f_{a,o,v}(s)\, b_a(s)\, \mathrm{d}s \\ &= \int_{s\in S} b_a(s)\, p(o|s,a)\, \alpha_v(s)\, \mathrm{d}s\end{aligned}$$

and its approximation with a set S' of points sampled according to b_a is

$$\hat{\mathrm{E}}(f_{a,o,v}) = \frac{1}{|S'|}\sum_{s\in S'} f_{a,o,v}(s)$$

To bound the approximation error using Theorem A.1, we need a class of bounded positive functions. Since $-P_{\max}R_{\max}/(1-\gamma) \le f_{a,o,v}(s) \le P_{\max}R_{\max}/(1-\gamma)$, we shift $f_{a,o,v}$ by a constant amount to $f'_{a,o,v}$ so that $0 \le f'_{a,o,v}(s) \le 2P_{\max}R_{\max}/(1-\gamma)$. We apply Theorem A.1 to the shifted functions and then transfer the result back to $\mathbf{F}_{a,G}$. Then, for any $\epsilon' > 0$,

$$\begin{aligned}&p(\exists o \in O, v \in G\colon |\hat{\mathrm{E}}(f_{a,o,v}) - \mathrm{E}(f_{a,o,v})| > \epsilon') \\ &\le 4\mathcal{C}_{a,G}(\epsilon'/16, N)\, \mathrm{e}^{-\epsilon'^2 N/128(\frac{2P_{\max}R_{\max}}{1-\gamma})^2} \\ &\le 4|G|\mathcal{C}_Z\left(\frac{\epsilon'(1-\gamma)}{16R_{\max}}, N\right)\mathrm{e}^{-\epsilon'^2(1-\gamma)^2 N/512(P_{\max}R_{\max})^2}\end{aligned} \tag{12}$$

where the inequality in the last line follows from Lemma A.2 and the definition of $\mathcal{C}_Z$ in (10). Now, for every $a \in A$, $o \in O$, and $v \in G$,

$$|V(b,a,o,v) - \hat{V}(b,a,o,v)| \le \epsilon'$$

with high probability, by definition. For every $a \in A$ and $o \in O$,

$$|V(b,a,o,\kappa_{ba}(o)) - V(b,a,o,\kappa^*_{ba}(o))| \le 2\epsilon' \tag{13}$$

with high probability, by Lemma A.3, as $\kappa^*_{ba}(o)$ maximizes $V(b,a,o,v)$ over all nodes v in G and $\kappa_{ba}(o)$ maximizes the sampled approximation of $V(b,a,o,v)$.

Next, we integrate the error in (13) over all observations and get

$$\begin{aligned}&|V(b,a,\kappa_{ba}) - V(b,a,\kappa^*_{ba})| \\ &\le \int_{o\in O} |V(b,a,o,\kappa_{ba}(o)) - V(b,a,o,\kappa^*_{ba}(o))|\, \mathrm{d}o \\ &\le 2\epsilon'\mu(O)\end{aligned}$$

where $\mu(O)$ denotes the measure of O. Since O is assumed to be an n-dimensional unit hypercube, we have $|V(b,a,\kappa_{ba}) - V(b,a,\kappa^*_{ba})| \le 2\epsilon'$.

Finally, we set $\epsilon = 2\epsilon'$, τ to be error bound in (12), and work out the number of samples required in term of ϵ and τ:

$$N \ge \frac{2048(P_{\max}R_{\max})^2}{\epsilon^2(1-\gamma)^2}\left(\ln\left(4|G|\mathcal{C}_Z\left(\frac{\epsilon(1-\gamma)}{32R_{\max}}\right)\right) - \ln\tau\right)$$

□

Proof (Theorem 5.2). First, we bound the error between the optimal value $V(b,a,\kappa^*_{ba})$ and its sampled approximation $\hat{V}(b,a,\kappa_{ba})$, which is obtained by running M simulations (Algorithm 1, lines 5–7):

$$\begin{aligned}&|V(b,a,\kappa^*_{ba}) - \hat{V}(b,a,\kappa_{ba})| \\ &\le |V(b,a,\kappa^*_{ba}) - V(b,a,\kappa_{ba})| \\ &\qquad + |V(b,a,\kappa_{ba}) - \hat{V}(b,a,\kappa_{ba})|\end{aligned}$$

We bound the first term above using Theorem 5.1. Choose $N \ge \rho_N(\epsilon/5, \tau/2|A|)$ and get

$$|V(b,a,\kappa^*_{ba}) - V(b,a,\kappa_{ba})| > \epsilon/5 \tag{14}$$

with probability at most $\tau/2|A|$. We bound the second term using Hoeffding's inequality. Choose $M \ge \rho_M(\epsilon/5, \tau/2|A|)$ and get

$$|V(b,a,\kappa_{ba}) - \hat{V}(b,a,\kappa_{ba})| > \epsilon/5 \tag{15}$$

with probability at most $\tau/2|A|$. Then, by the union bound,

$$p\big(\forall a \in A\colon |V(b,a,\kappa^*_{ba}) - \hat{V}(b,a,\kappa_{ba})| > 2\epsilon/5\big) \le \tau \tag{16}$$

Next, let $a^* = \arg\max_a V(b,a,\kappa^*_{ba})$ denote the optimal action and $\hat{a} = \arg\max_a \hat{V}(b,a,\kappa_{b,a})$ denote the approximately optimal action. Applying Lemma A.3 to (16), we get

$$p\big(|V(b,a^*,\kappa^*_{ba^*}) - V(b,\hat{a},\kappa^*_{b\hat{a}})| > 4\epsilon/5\big) < \tau$$

Finally,

$$\begin{aligned}|HV(b) - \hat{H}V(b)| &= |V(b,a^*,\kappa^*_{ba^*}) - V(b,\hat{a},\kappa_{b\hat{a}})| \\ &\le |V(b,a^*,\kappa^*_{ba^*}) - V(b,\hat{a},\kappa^*_{b\hat{a}}))| \\ &\qquad + |V(b,\hat{a},\kappa^*_{b\hat{a}}) - V(b,\hat{a},\kappa_{b\hat{a}})| \\ &\le 4\epsilon/5 + \epsilon/5\end{aligned}$$

The inequality in the last line holds with probability at least $1-\tau$, provided that (14) and (15) hold for all actions $a \in A$, with the same probability. Hence

$$p\big(|HV(b) - \hat{H}V(b)| \le \epsilon\big) > 1 - \tau$$

□

To prove Theorem 5.3, we need a Lipschitz condition (Bai et al., 2010).

Lemma A.4. *Suppose that a POMDP value function V can be represented as or approximated arbitrarily closely by a set of α-functions. For any $b, b' \in \mathcal{B}$, if $\|b - b'\|_1 \leq \delta$, then $|V(b) - V(b')| \leq \frac{R_{\max}}{1-\gamma}\delta$.*

Proof (Theorem 5.3). This proof is similar to the one for Theorem 2 in Bai et al. (2010). Let $\lambda_t = \max_{b \in B} |V^*(b) - V_t(b)|$ be the maximum error of $V_t(b)$ over the sampled beliefs in B. We first bound the maximum error of $V_t(b)$ at any arbitrary belief $b \in \mathcal{B}$ in terms of λ_t. For any point $b \in \mathcal{B}$, let b' be the closest point in B to b. Then

$$\begin{aligned}|V^*(b) - V_t(b)| \leq |V^*(b) - V^*(b')| + |V^*(b') - V_t(b')| \\ + |V_t(b') - V_t(b)|\end{aligned}$$

Applying Lemma A.4 twice to V^* and V_t, respectively, and observing that $|V^*(b') - V_t(b')| \leq \lambda_t$, we get

$$|V^*(b) - V_t(b)| \leq \frac{2R_{\max}\delta_B}{1-\gamma} + \lambda_t \tag{17}$$

Next, we bound the error λ_t. For any $b' \in B$,

$$\begin{aligned}|V^*(b') - V_t(b')| \leq & |HV^*(b') - \hat{H}_{b'}V_{t-1}(b')| \\ \leq & |HV^*(b') - HV_{t-1}(b')| \\ & + |HV_{t-1}(b') - \hat{H}_{b'}V_{t-1}(b')|\end{aligned} \tag{18}$$

where $\hat{H}_{b'}$ denotes invoking GPG-BACKUP at b'. The inequality in the first line in (18) holds, because by definition, $V^*(b') = HV^*(b')$, $V^*(b') \geq V_t(b')$, and $V_t(b') \geq \hat{H}_{b'}V_{t-1}(b')$. It is well known that the backup operator H is a contraction. The contraction property and (17) together imply

$$\begin{aligned}&|HV^*(b') - HV_{t-1}(b')| \\ &\quad \leq \gamma |V^*(b) - V_t(b)| \leq \gamma\left(\frac{2R_{\max}\delta_B}{1-\gamma} + \lambda_{t-1}\right)\end{aligned} \tag{19}$$

Theorem 5.2 guarantees that a single invocation of GPG-BACKUP at a belief b' has small approximation error with high probability, if N and M are sufficiently large. To obtain V_t, we perform t iterations of backup over the set B synchronously. Thus there are $|B|t$ invocations of GPG-BACKUP in total. Applying the union bound together with Theorem 5.2, every GPG-BACKUP invocation achieves

$$|HV_{t-1}(b') - \hat{H}_{b'}V_{t-1}(b')| < \epsilon \tag{20}$$

with probability $1 - \tau$, if we choose $N \geq \rho_N(\epsilon/5, \tau/2|A||B|t)$ and $M \geq \rho_N(\epsilon/5, \tau/2|A||B|t)$. We then combine (18–20) together with the definition of λ_t and get

$$\lambda_t \leq \gamma\left(\frac{2R_{\max}\delta_B}{1-\gamma} + \lambda_{t-1}\right) + \epsilon$$

For any initial policy graph, the error is bounded by

$$\lambda_0 \leq 2R_{\max}/(1-\gamma)$$

After solving the recurrence relation, we have

$$\begin{aligned}\lambda_t &= \frac{(1-\gamma^t)\epsilon}{1-\gamma} + \frac{2\gamma(1-\gamma^t)R_{\max}\delta_B}{(1-\gamma)^2} + \frac{2\gamma^t R_{\max}}{1-\gamma} \\ &\leq \frac{\epsilon}{1-\gamma} + \frac{2\gamma R_{\max}\delta_B}{(1-\gamma)^2} + \frac{2\gamma^t R_{\max}}{1-\gamma}\end{aligned}$$

Substituting it into (17) gives us the final result. □