Bayesian Reinforcement Learning via Deep, Sparse Sampling

Divya Grover **Debabrota Basu** **Christos Dimitrakakis**
Chalmers University of Technology University of Oslo

Abstract

We address the problem of Bayesian reinforcement learning using efficient model-based online planning. We propose an optimism-free Bayes-adaptive algorithm to induce deeper and sparser exploration with a theoretical bound on its performance relative to the Bayes optimal as well as lower computational complexity. The main novelty is the use of a candidate policy generator, to generate long-term options in the planning tree (over beliefs), which allows us to create much sparser and deeper trees. Experimental results on different environments show that in comparison to the state-of-the-art, our algorithm is both computationally more efficient, and obtains significantly higher reward over time in discrete environments.

1 INTRODUCTION

In Reinforcement Learning (Sutton and Barto, 1998), an agent sequentially interacts with an unknown environment with the objective of maximising its total reward over time. As the environment is unknown to the agent, it must carefully balance its actions in order to learn more about the environment (*exploration*) and obtain reward with high certainty (*exploitation*) as well. This dilemma of balancing exploration in the environment with exploiting the existing knowledge is referred to as the *exploration–exploitation trade-off*.

Bayesian Reinforcement Learning (BRL) solves this trade-off by constructing and using a probability distribution over possible models of the environment and trying to maximise total reward in expectation while marginalising over all possible models. This automatically takes into account the uncertainty about the environment. However, this "Bayes-optimal" policy is generally intractable as it requires performing dynamic programming over an exponentially large tree. Simpler solutions, such as Thompson sampling (Thompson, 1933), are known to be nearly optimal in some settings, such as multi-armed bandits (Kaufmann et al., 2012). Alternatively, one can construct approximate versions of the planning tree through Monte Carlo rollouts, sparse sampling, and limited look-ahead (Dimitrakakis, 2013a; Castro and Precup, 2010; Guez et al., 2012).

In this paper, we introduce the DSS (Deeper Sparser Sampling) algorithm to alleviate problems with existing approximations of the Bayes-optimal planner. DSS uses *policy samples* to create a deep tree with a smaller branching factor. We show that at any step, our algorithm produces an action that is with high probability close to the Bayes-optimal, and demonstrate experimentally that it outperforms the state-of-the-art BRL methods with significantly less computation.

The rest of the paper is organised as follows. In Section 2, we describe the framework of Markov Decision Processes (MDP) and Bayesian reinforcement learning. In Section 2.3, we discuss related work and the outline of our contribution. Section 3 elaborates the DSS algorithm. Then, we follow up by theoretical and experimental analysis of DSS in Section 4 and 5 respectively. Some technical proofs are relegated to the Appendix.

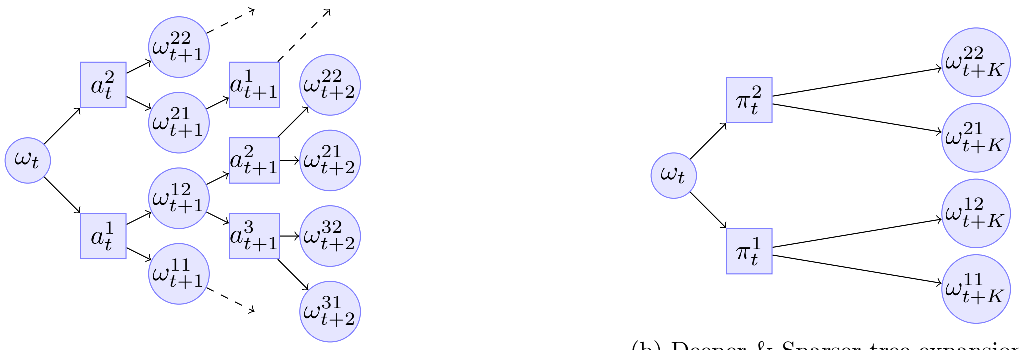
2 BACKGROUND AND RELATED WORK

2.1 Markov Decision Process (MDP)

Markov Decision Process (MDP) is a discrete-time stochastic process that provides a formal framework for reinforcement learning problems.

Definition 1. *An MDP $\mu = (S, A, P, R)$ is composed of a state space S, an action space A, a reward distribution R and a transition function P. The transition function $P \triangleq \mathbb{P}_\mu(s_{t+1}|s_t, a_t)$ dictates the distribution over next states s_{t+1} given the present state-action pair*

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).



(a) Full tree expansion.

(b) Deeper & Sparser tree expansion.

Figure 1: Visualising tree expansion. ω_t^{ij} denotes the information state at time t given action i and having observed state j.

(s_t, a_t). *The reward distribution* $R \triangleq \mathbb{P}_\mu(r_{t+1}|s_t, a_t)$ *dictates the obtained reward with* $r \in [0, 1]$. *We shall also use* $\mathbb{P}_\mu(r_{t+1}, s_{t+1}|s_t, a_t)$ *to denote the joint distribution of next states and actions of MDP* μ.

A policy π belonging to a policy space Π is an algorithm for selecting actions given present state and previous observations. A policy is Markov if at any time t, the action $a_t \in A$ chosen by the policy only depends on the current state s_t, so that the action distribution can be written as $\pi_t(a_t \mid s_t)$.
The value function of a policy for a specific MDP is the expected sum of discouted rewards obtained from time t to T while selecting actions in the MDP μ:

$$V_{t,T}^{\pi,\mu}(s) = \mathbb{E}_\mu^\pi(\sum_{k=1}^{T} \gamma^k r_{t+k} \mid s_t = s), \tag{1}$$

where $\gamma \in (0, 1]$ is called the discount factor and $\mathbb{E}_\mu^\pi$ denotes the expectation under the Markov chain generated by a policy π acting on the MDP μ. Let us define the infinite horizon discounted value function of a policy π on an MDP μ as $V_\mu^\pi \triangleq \lim_{T\to\infty} V_{0,T}^{\pi,\mu}$. Now, we define the *optimal value function* to be $V_\mu^* \triangleq \max_\pi V_\mu^\pi$, and the *optimal policy* to be $\pi_\mu^* \triangleq \arg\max_\pi V_\mu^\pi$. If the MDP is known, the optimal policy and value function is computable via backwards induction (alias, value iteration) (Puterman, 1994).

2.2 Bayes Adaptive MDP (BAMDP)

In reality, the underlying MDP is unknown to the reinforcement learning algorithm. This amounts to a trade-off between information seeking actions for performing better exploration and acting optimally given the current knowledge i.e. exploitation. This exploration-exploitation trade-off is one of the central issues in reinforcement learning. Bayesian Reinforcement Learning (BRL), specifically the information state formulation (Dearden et al., 1999; Duff, 2002), provides a framework to quantify this trade-off using Bayesian representation.
Following the Bayesian formulation, we maintain a belief distribution β_t over the possible MDP models $\mu \in \mathcal{M}$.[1] With an appropriate prior belief $\beta_0(\mu)$, we obtain a sequence of posterior beliefs $\beta_t(\mu)$ that represents our subjective belief over the MDPs at time t, depending on the latest observations. By Bayes' rule, the posterior belief at time $t + 1$ is

$$\beta_{t+1}(\mu) \triangleq \frac{\mathbb{P}_\mu(r_{t+1}, s_{t+1}|s_t, a_t)\beta_t(\mu)}{\int_{\mathcal{M}} \mathbb{P}'_\mu(r_{t+1}, s_{t+1}|s_t, a_t)\beta_t(\mu')d\mu'}. \tag{2}$$

Now, we define the Bayesian value function v analogously to the MDP value function:

$$v_\beta^\pi(s) \triangleq \int_{\mathcal{M}} V_\mu^\pi(s)\beta(\mu)d\mu. \tag{3}$$

Bayesian value function is the average utility that the decision maker is expected to obtain given its current belief β and policy π for selecting future actions. A policy computed using Bayesian value function can in general be adaptive, and indeed this holds for the Bayes-optimal policy. For completeness, we also define the Bayes-optimal utility $v_\beta^*(s)$, i.e. the utility of the Bayes-optimal policy.

$$v_\beta^*(s) \triangleq \max_{\pi\in\Pi} \int_{\mathcal{M}} V_\mu^\pi(s)\beta(\mu)d\mu. \tag{4}$$

It is well known that by combining the original MDPs state s_t and belief β_t into a hyper-state ω_t, we obtain another MDP called the Bayes Adaptive MDP (BAMDP). The optimal policy for a BAMDP is the same as the Bayes-optimal policy for the corresponding MDP.

[1] More precisely, we can define a measurable space $(\mathcal{M}, \mathfrak{M})$, where $\mathcal{M}$ is the possible set of MDPs, and $\mathfrak{M}$ is a suitable σ-algebra.

Definition 2 (BAMDP)**.** *A Bayes Adaptive Markov Decision Process (BAMDP) $\tilde{\mu} \triangleq (\Omega, A, \nu, \tau)$ is a representation for an unknown MDP $\mu = (S, A, P, R)$ with a space of information states $\Omega = S \times \mathfrak{B}$, where $\mathfrak{B}$ is an appropriate set of belief distributions on $\mathcal{M}$. At time t, the agent observes the information state $\omega_t = (s_t, \beta_t)$ and takes action $a_t \in A$. We denote the transition distribution as $\nu(\omega_{t+1}|\omega_t, a_t)$, the reward distribution as $\tau(r_{t+1}|\omega_t, a_t)$, and A as the common action space.*

For each s_{t+1}, the next hyper-state $\omega_{t+1} = (s_{t+1}, \beta_{t+1})$ is uniquely determined since β_{t+1} is unique given (ω_t, s_{t+1}) and can be computed using equation (2). Therefore the information state ω_t preserves the Markov property. This allows us to treat the BAMDP as an infinite-state MDP with $\nu(\omega_{t+1}|\omega_t, a_t)$, and $\tau(r_{t+1}|\omega_t, a_t)$ defined as the corresponding transition and reward distributions respectively. The transition and reward distributions are defined as the marginal distributions

$$\nu(\omega_{t+1}|\omega_t, a_t) \triangleq \int_{\mathcal{M}} \mathbb{P}_\mu(s_{t+1}|s_t, a_t)\beta_t(\mu)d\mu,$$

$$\tau(r_{t+1}|\omega_t, a_t) \triangleq \int_{\mathcal{M}} \mathbb{P}_\mu(r_{t+1}|s_t, a_t)\beta_t(\mu)d\mu.$$

Though the Bayes-optimal policy is generally adaptive in the original MDP, it is Markov with respect to the hyper-state of the BAMDP. In other words, ω_t represents a sufficient statistic for the observed history.

Since the BAMDP is an MDP on space of hyper-states, we can use backwards induction (alias, value iteration) starting from the set of terminal hyper-states Ω_T and proceeding backwards to $T-1, \ldots, t$ following

$$V^*_t(\omega) = \max_{a \in A} \mathbb{E}[r \mid \omega, a] + \gamma \sum_{\omega^{'} \in \Omega_{t+1}} \nu(\omega^{'}|\omega, a)V^*_{t+1}(\omega^{'}), \tag{5}$$

where Ω_{t+1} is the reachable set of hyper-states from hyper-state ω_t. Equation (4) implies Equation (5) and vice-versa, i.e. that $v^*_\beta(s) = V^*_0(\omega)$ for $\omega = (s, \beta)$ (Appendix B). Hence, we can obtain Bayes-optimal policies through backwards induction. Due to the large hyper-state space, this is only feasible for small horizons T in practice.

2.3 Related Work

BRL was initially investigated in (Silver, 1963; Martin, 1967). The problem of computational intractability of the Bayes-optimal solution motivated researchers to design approximate techniques. Dearden et al. (1998, 1999) proposed Bayesian Q-learning and Duff (2003) proposed a diffusion based approximation of Bayesian Markov chains. A vast research has been conducted towards model based BRL algorithms, which is comprehensively compiled in a survey by Ghavamzadeh et al. (2015). We classify these algorithms in two categories: Myopic and Lookahead.

Algorithm 1 FHTS (Finite Horizon Tree Search)

```
Parameters: Horizon $T$
Input: current hyper-state $\omega_h$ and depth $h$.
if $h = T$ then
  return V$(\omega_h) = 0$
end if
for all actions a do
  for all next states $s_{h+1}$ do
    $\beta_{h+1}$ = UpdatePosterior($\omega_h$,$s_{h+1}$,a) (eq. 2)
    $\omega_{h+1} = (s_{h+1}, \beta_{h+1})$
    $V(\omega_{h+1})$ = FHTS($\omega_{h+1}, h+1$)
  end for
end for
Q$(\omega_h, a) = 0$
for all $\omega_{h+1}, a$ do
  $Q(\omega_h, a) \mathrel{+}= \nu(\omega_{h+1}|\omega_h, a) \times V(\omega_{h+1})$
end for
return $\max_a Q(\omega_h, a)$
```

Myopic: Myopic algorithms do not lookahead in future, rather they take actions depending on present information. Thompson sampling (Thompson, 1933) maintains a posterior distribution over transition models, samples an MDP and chooses the optimal policy for the sample. A reformulation of this for BRL is proposed as Bayesian DP in (Strens, 2000). The Best Of Sampled Set (BOSS) (Asmuth et al., 2009) algorithm generalizes this idea to a multi sample optimistic approach. Monte-Carlo Utility Estimates for BRL (MCBRL) (Dimitrakakis, 2011, 2013b) generalizes these ideas to lower bound policies and gradient based value function estimates for improved performance.

Lookahead: The simplest algorithm is to calculate and solve the BAMDP up to some horizon T, as outlined in *Algorithm 1* and is illustrated in Figure 1a. A simple modification to it is Sparse sampling by Kearns et al. (1999), which instead only iterates over a set of sampled states. When applied to BAMDP belief tree[2], the Kearns algorithm would still have to consider all primitive actions. Wang et al. (2005) improved upon this by using Thompson sampling to only consider a subset of promising actions. High branching factor of the tree still makes planning with deep horizon computationally expensive. Thus, more scalable algorithms, such as BFS3 (Asmuth and Littman,

[2] We freely use the term 'tree' or 'belief tree' to denote the planning tree generated by the algorithms in the hyper-state space of BAMDP.

2011) and BAMCP (Guez et al., 2012), were proposed. Similar to (Wang et al., 2005), BFS3 also selects a subset of actions but with an optimistic action selection strategy, though the backups are still performed using Bellman equation. BAMCP takes a Monte-Carlo approach to sparse lookahead in belief-augmented version of Markov decision process. BAMCP also uses optimism for action selection.Unlike BFS3, the next set of hyper-states are sampled from an MDP sampled at the root[3]. Since posterior inference is expensive for any non-trivial belief model, BAMCP further applies lazy sampling and rollout policy, inspired by their application in tree search problems Kocsis and Szepesvári (2006).

Our contribution: Unlike other approaches, we focus on reducing the branching factor by considering K-step policies instead of primitive actions when planning. These policies are generated through (possibly approximate) Thompson sampling. This approach is rounded by using Sparse sampling (Kearns et al., 1999). The reduced branching allows us to build a deeper tree. The intuition why this might be desirable is that if the belief changes slowly enough, an adaptive policy that is constructed out of a tree of K-step stationary policies will still be approximately optimal. This intuition is supported by the theoretical analysis in Section 4. In Section 4, we prove that our algorithm results in nearly-optimal planning under certain mild assumptions regarding the belief. Section 5 experimentally shows that we get better policies than the state-of-the-art with less computation time. The freedom to choose a policy generator allows the algorithm scale smoothly. We choose Policy Iteration (PI) and a variant of Real Time Dynamic Programming (RTDP) for different sizes of environments.

3 DEEPER & SPARSER SAMPLING (DSS)

The core idea of DSS algorithm is to plan in the belief tree, not at the individual action level, but at the level of K-step policies. Figure 1b illustrates this concept graphically. At each time-step t, Algorithm 2 is called with the current state s and belief β as input, with additional parameters controlling how the tree is approximated. The algorithm then generates the tree and calculates the value of each policy candidate recursively (for H stages or episodes), in the following manner:

Algorithm 2 DSS

```
1: **Parameters:** Number of stages $H$, steps $K$, no. of policies $N$, no. of samples per policy $M$, policy generator $\mathcal{P}$
2: **Input:** hyper-state $\omega_h = (s_h, \beta_h)$, depth $h$.
3: **if** $h = KH$ **then**
4:     return $V(\omega_h) = 0$
5: **end if**
6: $\Pi_{\beta_h} = \{\mathcal{P}(\mu_i) | \mu_i \backsim \omega_h, i \in \mathbb{Z}, i \leq N\}$
7: **for** all $\pi \in \Pi_{\beta_h}$ **do**
8:     $Q(\omega_h, \pi) = 0$
9:     **for** 1 to $M$ **do**
10:        $R = 0, c = \gamma^h, k = 0$
11:        $\omega_k = \omega_h, s_k = s_h, \beta_k = \beta_h, a_k = \pi(s_h)$
12:        **for** $k = 1, \dots, K$ **do**
13:            $s_{k+1} \backsim \nu(\omega_{k+1} | \omega_k, a_k)$
14:            $r_{k+1} \backsim \tau(r_{k+1} | \omega_k, a_k)$
15:            $R$ += $c \times r_{k+1}$; $c = c \times \gamma$
16:            $\beta_{k+1} = \text{UpdatePosterior}(\omega_k, s_{k+1}, a_k)$ (from eq. 2)
17:        **end for**
18:        $Q(\omega_h, \pi) += R + \text{DSS}(\omega_K, h + K)$
19:    **end for**
20:    $Q(\omega_h, \pi) /= M$
21: **end for**
22: **return** $\arg\max_\pi Q(\omega_h, \pi)$
```

1. Line 6: Generate N MDPs from the current belief β_t, and for each MDP μ_i use the policy generator $\mathcal{P} : \mu \to \pi$ to generate a policy π_i. This gives a policy set Π_β with $|\Pi_\beta| = N$.

2. Line 10-18: Run each policy for K steps, collecting total K-step dicounted reward R in BAMDP. Note that we sample the reward and next-state from the marginal (Line 13-14), and also update the posterior (Line 16).

3. Line 19-21: Make recursive call to DSS at the end of K steps. Repeat the process just described for M times. This gives an M-sample estimate of that policy's utility v^π_β.

Note that the fundamental control unit that we are trying to find here is a policy, hence Q-values are defined over (ω_t,π) tuples. Since we now have policies at any given tree node, we re-branch only after running those policies for K steps. Hence we can increase the effective depth of the belief tree upto HK for the same computational budget. This allows for deeper lookahead and ensures that the approximation error propagated is also smaller as the error is discounted by γ^{HK} instead of γ^H. We elaborate this effect in the theoretical analysis.

[3] Note that ideally the next observations should be sampled from the $P(s_{t+1}|\omega_t)$ instead of $P(s_{t+1}|\omega_{t_o})$, i.e. the next-state marginal at the root belief.

4 THEORETICAL ANALYSIS

The fundamental analysis of Kearns et al. (1999) for any approximate tree based planning algorithm (like Algorithm 1) is due to union bound on sampling approximation of every action-value at each node in the tree, where bound is obtained due to discounting of error with increasing depth. In reality, due to exponential nodes with $|A||S|$ branching per level, computational limit is quickly reached and leaf-approximations are needed. We improve on this approach by imposing certain assumptions about the belief in the planning tree and using the duality between Eq.(4) and Eq.(5).

In order to prove that DSS is nearly optimal, we need two assumptions, and consider an idealized version of the algorithm, ignoring some approximations done for computational simplicity.[4]

Assumption 1. *The belief β_h in the planning tree is such that $\epsilon_h \leq \epsilon_0/h$, where $h \geq 1$, $\epsilon_h = ||\hat{\beta}_h - \beta_h||_1$ and $\hat{\beta}_h$ is the constant belief approximation at the start of episode h.*

The first assumption states that as we go deeper in the planning tree, the belief error reduces. The intuition is that if the belief concentrates at a certain rate, then so does error of Bayes utility for any Markov policy, by the virtue of its definition (shown in Appendix A, Lemma (3)). The ϵ_0 denotes a constant dependent on the current root belief β.

Assumption 2. $\beta_t(\mu)\beta_t(\mu') \leq \frac{C}{D(\mu,\mu')}$, *where* $D(\mu, \mu') \triangleq \max_{s,a} \|P_\mu(\cdot \mid s,a) - P_{\mu'}(\cdot \mid s,a)\|_1$.

The second assumption states that belief correlation across similar MDPs is higher than across dissimilar ones.

Our algorithm finds a near-Bayes-optimal policy, as stated in Theorem 1.

Theorem 1. *Under Assumptions 1 and 2, $\forall s \in S$*

$$v_\beta^{DS}(s) \geq v_\beta^*(s) - \left(2\epsilon_0 K \ln \frac{1}{1-\gamma^K} + \frac{2(KC+\gamma^K)}{(1-\gamma)}\right) - \sqrt{\frac{\ln M/\delta}{2N(1-\gamma)^2}}$$

with probability $1-\delta$. Here, T is the horizon, divided by parameter K into H stages, i.e, $T = KH$. In addition, at each node of the sparse tree, we evaluate N policies for M times.

At the same time, the algorithm is significantly less computationally expensive than basic Sparse sampling (Kearns et al., 1999) which would take $O((|A|M)^T)$ calls to the generative BAMDP model, while we require only $O((NM)^{T/K})$ calls for a T-horizon problem.

4.1 Proof Overview

Let's consider the planning process to be computed till horizon T, which is divided in H episodes each of length K. Thus, we get $T = KH$. Let Π_K be the set of all policies $\pi_1^H \triangleq \{\pi_i\}_{i=1,\dots,H}$. Each π_1^H is a concatenation of H, K-horizon policies. Hereafter, we refer to such policies as K-step policies. Since planning is divided into episodes, we define the episodic utility in episode $h+1$ as:

$$v_{\beta_h}^\pi(s) \triangleq \int_{\mathcal{M}} V_{0,K}^{\pi,\mu}(s)\beta_h(\mu)d\mu.$$

Here, β_h is the belief at start of episode h. Similar to the definition of overall utility in Equation (3), episodic utility of π defines the expected utility of taking K steps in the BAMDP starting from belief β_h . Let π_β^* be the Bayes-optimal policy, π_β^{DS} be the DSS policy, π_β^K the Bayes-optimal adaptive policy that is restricted to K-step policies, and π_β^{TS} the Thompson sampling policy, with respective utilities $v_\beta^*, v_\beta^{\mathrm{DS}}, v_\beta^K, v_\beta^{\mathrm{TS}}$.

Now, we write the Bayesian regret of DSS policy relative to the Bayes optimal policy and decompose it in terms of relative regret of the the aforementioned policies:

$$\begin{aligned}\left\|v_\beta^* - v_\beta^{\mathrm{DS}}\right\|_\infty &= \left\|v_\beta^* - v_\beta^K + v_\beta^K - v_\beta^{\mathrm{TS}} + v_\beta^{\mathrm{TS}} - v_\beta^{\mathrm{DS}}\right\|_\infty \\ \leq \left\|v_\beta^* - v_\beta^K\right\|_\infty &+ \sum_h \gamma^{Kh} \left\|v_{\beta_h}^K - v_{\beta_h}^{\mathrm{TS}}\right\|_\infty + \left\|v_\beta^{\mathrm{TS}} - v_\beta^{\mathrm{DS}}\right\|_\infty\end{aligned} \tag{6}$$

We bound the first and second term of (6) by Lemmas 1 and 2 below.

Lemma 1 (Anytime Error)**.** *Under Assumption 1,*

$$\left\|v_\beta^* - v_\beta^K\right\|_\infty \leq 2\epsilon_0 K \ln \frac{1}{1-\gamma^K}.$$

Lemma 2 (Error of Thompson-sampling-distributed Policy)**.** *For any episode belief β, under Assumption 2:*

$$\left\|v_\beta^K - v_\beta^{TS}\right\|_\infty \leq \frac{2(KC+\gamma^K)}{(1-\gamma)}.$$

Theorem 1 (sketch). Merging the errors due to Anytime error and Thompson-sampling-distributed error from Lemmas (1) and (2), we obtain $v_\beta^{\pi_\beta^{\mathrm{TS}}}(s) \geq v_\beta^*(s) - \left(2\epsilon_0 K \ln \frac{1}{1-\gamma^K} + \frac{2(KC+\gamma^K)}{(1-\gamma)}\right)$ for all s. Combined with an additional Hoeffding inequality for last term of eq.(6) we obtain Theorem (1). □

[4] In particular, the sampled policies are not strictly coming from the Thompson sampling distribution, due to the use of partial policy iteration or RTDP.

5 EXPERIMENTAL ANALYSIS

Experimental protocol. We empirically evaluate performance of DSS in comparison with three different algorithms on four different environments. Each algorithm has a number of hyperparameters to choose. Some of which, such as the prior belief, are common to all algorithms. The remainder are unique to each algorithm which are tuned in the following manner:
For each environment μ and algorithm π combination, we evaluate the algorithm's hyperparameter λ over 10 experiments with horizon T and select the value maximising average cumulative reward over them, i.e. $\lambda^*(\mu, \pi) = \arg\max_\lambda \sum_{i=1}^{10} \sum_{t=1}^{T} r_t^{(i)}$, where $r_t^{(i)}$ is the reward sequence of the i-th experiment. The parameter sets for each algorithm are detailed in Appendix C. The final evaluation, and results shown, was performed over 100 runs using the chosen λ^*. This is done to avoid selection of the best parameter in hindsight.

Algorithms. In our experiments, we consider four lookahead algorithms, all of which expand the BAMDP to a finite horizon.[5]

Sparser: Alg. 2 with two variants of policy generators: PI and RTDP. PI refers to exact discounted Policy-iteration while RTDP refers Barto et al. (1995), where the RTDP horizon can intuitively be taken as K as we run the generated policy for next K-steps in the belief tree.

BAMCP[6]: The current state-of-the-art. It applies UCT algorithm in belief tree, combined with root-sampling and lazy sampling for faster computation. (Guez et al., 2012)

SBOSS[6]: A more effective variant of BOSS algorithm. BOSS algorithm samples multiple MDPs from the belief, creates an extended MDP using the samples, then solving it to yeild an optimistic policy. (Castro and Precup, 2010)

BFS3[6]: An optimistic follow-up to Wang et al. (2005), it performs optimistic action selection in belief tree planning. It main advantage lies in non-uniform trajectory selection. (Asmuth and Littman, 2011)

Environments.

1. *Chain*: An MDP consisting of 5 states, connected in a linear chain, with a big reward opposite to the start state at one corner Dearden et al. (1998).[7]
2. *DoubleLoop* Dearden et al. (1998), a 9-state MDP consisting of two seperate loops, sharing one state in common.
3. *Grid*: Two sparse-reward environments, represented by square grids, of size 5x5 (Grid5) and 10x10 (Grid10), with reward only at goal state. Initial state is always diagnoally opposite to the goal state.
4. *Maze* Dearden et al. (1998). A grid world with 264 states, consisting of flags to be collected a various locations, which inturn decide the reward value when goal state is finally reached. The states are encoded by location of agent, as well as flag status.

Shared parameters. Some parameters are shared by all algorithms. When possible, we reuse the ones used in (Guez et al., 2012):

- We impose a limit 0.25sec/step for Chain and DoubleLoop, 1.5sec/step for Maze and 1sec/step for the grid environments. Hyperparameter values exceeding those limits were excluded from the hyperparameter search.
- We assume known rewards. We recompute the optimal action at each step in simulation.
- Experiments last for $T = 1000$ steps in Chain, DoubleLoop and Grid5, $T = 2000$ in Grid10 steps and in $T = 20000$ in Maze.
- We use a hierarchical Dirichlet (Friedman and Singer, 1999) on the transition probabilities.
- We use the environment simulators from (Guez et al., 2012).[8]

5.1 Analysis of Results

We measure three quantities over 100 trials for each environment: the mean total reward (Table 1), the per-step average reward (Figure 2), and the CPU time (Table 2). The CPU time denotes the time taken per episode for the best performing parameters.

Table 1 shows the average cumulative reward, a comparison metric used in previous works, for each of the algorithms on each of the environments. The standard error incurred by both DSS variants is small enough. This implies that both DSS variants outperform the current state of the art for all environments tested;

[5] Myopic algorithms like Thompson sampling were not excluded. In particular, TS is a special case of *SBOSS*, but in our hyperparameter search it was always automatically excluded.

[6] We use the implementations from BAMCP paper.

[7] Note that Chain was not compared in the BAMCP paper. For all other environments, we used a configuration identical to experiments in Guez et al. (2012).

[8] Code: https://github.com/acguez/bamcp

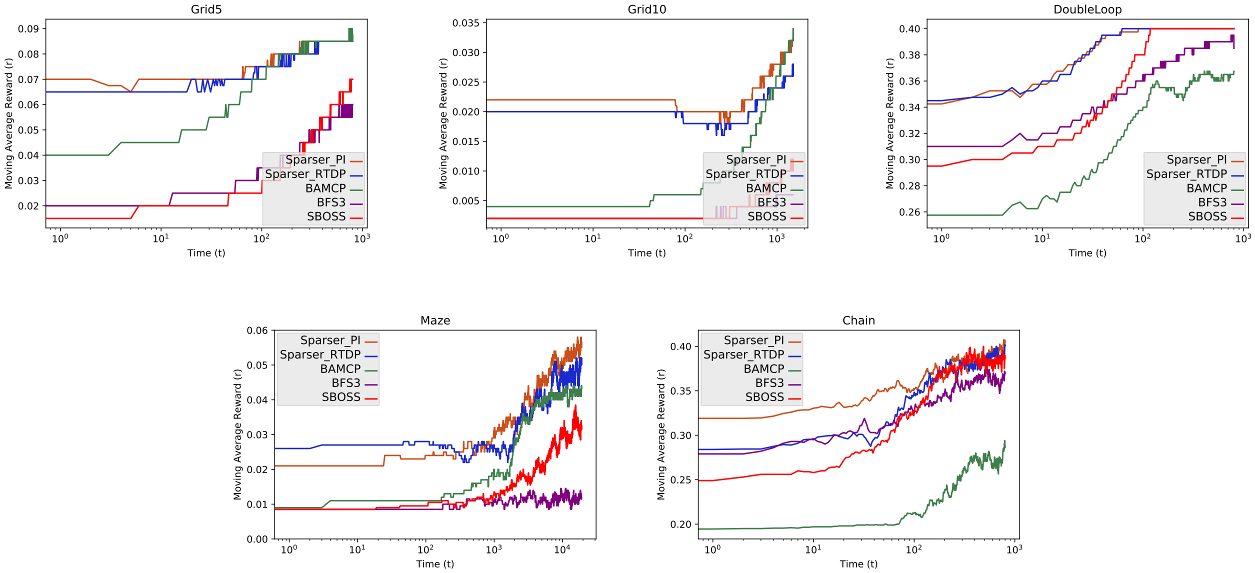


Figure 2: Moving average performance on log time scale

Algorithm	Chain	DoubleLoop	Grid5	Grid10	Maze
Sparser-RTDP	358.97±5.15	**387.20±0.64**	78.74±0.80	44.32±0.89	849.99±20.68
Sparser-PI	**370.06±4.71**	380.60±0.62	**79.01±0.47**	**50.91±0.50**	**944.99±19.36**
BAMCP	267.63±5.72	309.32±4.26	73.92±0.96	37.07±0.72	738.2±21.96
BFS3	340.57±4.51	367.95±0.74	44.94±0.88	8.60±0.28	225±6.88
SBOSS	351.49±4.28	371.11±1.77	47.50±0.36	13.32±0.35	513.25±5.59

Table 1: Total reward obtained, averaged over 100 experiments, shown with standard errror.

following a rigorous and unbiased hyperparameter selection process.

Figure 2, which shows the time evolution of the average reward. For clear illustration, the average reward is smoothed over a window of 200 steps (500 steps for the Maze). DSS outperforms all other algorithms initially, due to better exploratory actions. In most cases there exists at least one (different) algorithm that achieves the asymptotic performance of DSS. This phenomenon is expected since beliefs of all competing algorithms converge to the true model, but they are generally unable to converge as fast as DSS for all the environments.

The advantage of DSS is not only in terms of performance but also in terms of efficiency. Table 2 shows that DSS takes significantly less time per episode for larger environments than its immediate predecessor, BFS3, and also often manages to outperform the state-of-the-art BAMCP in terms of computation time.

It is important to note that, although we impose a per-step time limit on computation, the performance of the tested algorithms does not necessarily increase with computation time. For example, we observe that performance of BAMCP actually drops when number of root samples are increased from 10^5 to 10^6 while keeping other parameters constant. This reinforces the need for using an unbiased experimental methodology for tuning hyperparameters, as advocated in this paper. Similar observations were made in (Guez et al., 2012) regarding SBOSS and BFS3. For DSS, in practice, the performance generally increases with parameters N and M but plateaus quite fast. For further reference, the chosen hyperparameters are shown in Table 3 (Appendix C).

6 DISCUSSION AND FUTURE WORK

We propose an optimism-free algorithm that induces deeper and sparser exploration, with a PAC planning process, and also achieves state-of-the-art results with lower computational complexity. The PAC guarantee provides DSS with theoretical strength relative to other state-of-the-art algorithms (c.f. Table 4.1

Algorithm	Chain	DoubleLoop	Grid5	Grid10	Maze
Sparser-RTDP	0.93	1.31	5.32	97.4	1267.2
Sparser-PI	2.72	1.70	5.73	142.2	1532.0
BAMCP	0.56	1.25	172.46	315.7	1789.4
BFS3	6.25	2.26	54.03	>2000*	3558.7
SBOSS	0.01	0.01	0.28	300.95	3695.5

Table 2: Time taken in seconds per epsiode. (*Time limit exceeded)

in Ghavamzadeh et al. (2015)). The analysis also shows how the gap between the Bayes-optimal policy and DSS depends on the main hyperparameter K.

In comparison, BAMCP is Bayes optimal-policy only asymptotically (Guez et al., 2012). The guarantees for BOSS are PAC-MDP (i.e. that there is only a polynomial number of steps for which its takes an action with unbounded utility error), However, Araya et al. (2012) argue that PAC-MDP is not the most suitable for evaluating BAMDP algorithms. Finally, the theoretical properties of BFS3, which can be regarded as the immediate predecessor to DSS, are not known.

Experimental results on different environments show that, compared to the state-of-the-art, our algorithm is both more efficient and obtains higher reward. In practice, we drastically reduce the computation time compared to its immediate Forward Search predecessor BFS3, as can be seen in Table 2. This is because we only compute policies every K-step while planning in belief tree. And unlike BFS3, instead of maintaining upper and lower bounds on observation nodes, we simply select them by sampling from the current posterior in the tree branch.

Future extensions to this work can be to provide tighter bounds for Thompson policies, similar to very recent work by (Efroni et al., 2019); reinforcing this approach of planning at policy level instead of individual action level. Preliminary result of DSS on 'MountainCar' environment is also given (Appendix D), hinting at possible ways of achieving full scalability. DSS can also be extended to continuous state-spaces, by keeping a prior over models other than discrete MDPs, such as linear state-space model or a non-linear Neural Network model. However, this would require us to strike a delicate balance between approximations in inference and planning, and is left for a subject for future work.

References

Mauricio Araya, Olivier Buffet, and Vincent Thomas. Near-optimal brl using optimistic local transitions. *arXiv preprint arXiv:1206.4613*, 2012.

J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *UAI 2009*, 2009.

John Asmuth and Michael L Littman. Approaching bayes-optimalilty using monte-carlo tree search. In *Proc. 21st Int. Conf. Automat. Plan. Sched., Freiburg, Germany*, 2011.

Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.

Pablo Samuel Castro and Doina Precup. Smarter sampling in model-based bayesian reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 200–214. Springer, 2010.

Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998. URL `citeseer.ist.psu.edu/dearden98bayesian.html`.

Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 150–159, San Francisco, CA, July 30–August 1 1999. Morgan Kaufmann, San Francisco, CA.

Christos Dimitrakakis. Robust bayesian reinforcement learning through tight lower bounds. In *European Workshop on Reinforcement Learning*, page arXiv:1106.3651v2. Springer, 2011.

Christos Dimitrakakis. Monte-carlo utility estimates for bayesian reinforcement learning. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 7303–7308. IEEE, 2013a.

Christos Dimitrakakis. Monte-carlo utility estimates for bayesian reinforcement learning. In *IEEE 52nd Annual Conference on Decision and Control (CDC 2013)*, 2013b. arXiv:1303.2506.

Michael O Duff. Diffusion approximation for bayesian markov chains. In *Proceedings of the 20th Interna-*

tional Conference on Machine Learning (ICML-03), pages 139–146, 2003.

Michael O'Gordon Duff. *Optimal Learning Computational Procedures for Bayes-adaptive Markov Decision Processes.* PhD thesis, University of Massachusetts at Amherst, 2002.

Yonathan Efroni, Mohammad Ghavamzadeh, and Shie Mannor. Multi-step greedy and approximate real time dynamic programming. *arXiv preprint arXiv:1909.04236*, 2019.

Eyal Even-Dar and Yishai Mansour. Approximate equivalence of markov decision processes. In *Learning Theory and Kernel Machines. COLT/Kernel 2003*, Lecture notes in Computer science, pages 581–594, Washington, DC, USA, 2003. Springer.

Nir Friedman and Yoram Singer. Efficient bayesian parameter estimation in large discrete domains. In *Advances in neural information processing systems*, pages 417–423, 1999.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2012.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An optimal finite time analysis. In *ALT-2012*, 2012.

Michael J. Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In Thomas Dean, editor, *IJCAI*, pages 1324–1231. Morgan Kaufmann, 1999. ISBN 1-55860-613-0.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

James John Martin. *Bayesian decision problems and Markov chains.* Wiley, 1967.

Marting L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New Jersey, US, 1994.

Edward A Silver. Markovian decision processes with uncertain transition probabilities or rewards. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE OPERATIONS RESEARCH CENTER, 1963.

Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, pages 943–950, 2000.

Richard S. Sutton and Andrew G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

W.R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of two Samples. *Biometrika*, 25(3-4):285–294, 1933.

Tao Wang, Daniel Lizotte, Michael Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML '05*, pages 956–963, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: http://doi.acm.org/10.1145/1102351.1102472.

Bayesian Reinforcement Learning via Deep, Sparse Sampling

Supplementary material

A Proofs of Section 4

The following Lemma shows how episodic error is directly proportional to the belief-error:

Lemma 3 (Per-episode Error)**.** *The error in reward under the true and approximate belief in any episode is bounded by*

$$\Delta_h \triangleq \sup_{\pi} \left\| v^{\pi}_{\beta_h} - v^{\pi}_{\hat{\beta}_h} \right\|_\infty \le \frac{1-\gamma^K}{1-\gamma}\epsilon_h \le K\epsilon_h.$$

Proof. Due to the fact that the two BAMDPs induced by the beliefs are ϵ_k close in L1 norm, we can use the argument in Theorem 1 of Dimitrakakis (2011). □

We shall also use a trivial lemma from Analysis:

Lemma 4. *If $\|f-g\|_\infty \le \epsilon$ and $f(x^*) \ge f(x)$, $g(y^*) \ge g(y)$ then $f(y^*) \ge f(x^*) - 2\epsilon$.*

We are now ready to prove the main results:

Lemma 1 (Anytime Error)**.** *Under Assumption 1,*

$$\left\| v^*_\beta - v^K_\beta \right\|_\infty \le 2\epsilon_0 K \ln \frac{1}{1-\gamma^K}.$$

Proof. First note that if the belief is only changing every K steps, then the Bayes-optimal policy is Markovian over K steps. This means that finding a K-step Markovian policy starting from belief β_h is the same as finding the optimal policy for a fixed belief $\hat{\beta}_h$.This allows us to use Lemma 3 to bound the error of the K-step policy.

Let $f(\pi) \triangleq v^{\pi}_{\beta_h}$ and $g(\pi) \triangleq v^{\pi}_{\hat{\beta}_h}$. Therefore,

$$
\begin{aligned}
\left\|v^*_\beta - v^K_\beta\right\|_\infty &= \sum_{h=1}^{H} \gamma^{hK} (v^*_{\beta_h} - v^K_{\beta_h}) \\
&\overset{(a)}{\leq} 2\sum_{h=1}^{H} \gamma^{hK} \Delta_h \\
&\overset{(b)}{\leq} 2\sum_{h=1}^{H} \gamma^{hK} K \epsilon_h \\
&\overset{(c)}{\leq} 2K \times \sum_{h=1}^{H} \gamma^{hK} \epsilon_0 / h \\
&\leq 2\epsilon_0 K \times \sum_{h=1}^{\infty} \gamma^{Kh} / h \\
&\overset{(d)}{=} 2\epsilon_0 K \times \ln \frac{1}{1-\gamma^K}
\end{aligned}
$$

(a) is obtained using Lemma 4.

(b) is a consequence of Lemma 3.

(c) is a consequence of Assumption 1.

(d) is derived from the fact that $\sum_{x=1}^{\infty} \frac{a^x}{x} = -\log(1-a)$ for $1 > a \geq 0$.

Therefore, $\left\|v^*_\beta - v^K_\beta\right\|_\infty \leq 2\epsilon_0 K \ln \frac{1}{1-\gamma^K}$. □

Lemma 2 (Error of Thompson-sampling-distributed[9] episodic Policy)**.** *Under Assumption 2*

$$
\left\|v^K_\beta - v^{TS}_\beta\right\|_\infty \leq \frac{2(KC + \gamma^K)}{(1-\gamma)}.
$$

Proof. We want to show that the value of the TS-episodic policy is not much worse than the Bayes-optimal K-step stationary policy.

$$
\begin{aligned}
&v^K_\beta - v^{\mathrm{TS}}_\beta \\
&\overset{(a)}{\leq} v^*_\beta(s) - v^{\mathrm{TS}}_\beta(s) \\
&\overset{(b)}{\leq} \int_{\mathcal{M}} d\mu \beta(\mu) \left[V^{\pi^*_\beta, \mu}_{0,K}(s) - \int_{\mathcal{M}} d\mu' V^{\pi^*_{\mu'}, \mu'}_{0,K}(s) \beta(\mu') \right] \\
&+ \frac{2\gamma^K}{1-\gamma} \\
&= \int_{\mathcal{M}} \int_{\mathcal{M}} \left[V^{\pi^*_\mu, \mu}_{0,K}(s) - V^{\pi^*_{\mu'}, \mu'}_{0,K}(s) \right] \beta(\mu)\beta(\mu') d\mu d\mu' \\
&+ \frac{2\gamma^K}{1-\gamma} \\
&\overset{(c)}{\leq} \int_{\mathcal{M}} \int_{\mathcal{M}} \frac{2KD(\mu, \mu')}{(1-\gamma)} \beta(\mu)\beta(\mu') d\mu d\mu' \\
&+ \frac{2\gamma^K}{1-\gamma} \\
&\overset{(d)}{\leq} \frac{2(KC + \gamma^K)}{(1-\gamma)}.
\end{aligned}
$$

(a) follows by the definition of the Bayes-optimal policy.

(b) follows by truncating the reward sequence to K steps.

(c) follows from the approximate MDP Lemma (Even-Dar and Mansour, 2003, Lemma 4) by definition of the MDP distance $D(\mu, \mu')$.

(d) is a direct consequence of Assumption 2. □

Proof of Theorem 1. For the final proof, we add the effect of sampling in the total error from the two lemmas:

Proof. Merging the errors due to Thompson-sampling-distributed error and the anytime-error from Lemma (2) and (1), we obtain for all s

$$
v^{\pi^{\mathrm{TS}}_\beta}_\beta(s) \geq v^*_\beta(s) - \left(2\epsilon_0 K \ln \frac{1}{1-\gamma^K} + \frac{2(KC + \gamma^K)}{(1-\gamma)} \right).
$$

We can then use Hoeffding's bound since utility of π^{DS}_β is just sampled utility of π^{TS}_β. For simplicity, let $\bar{\rho}$ be the expected error of a TS policy and ρ_i of the i-th sampled policy and let

$$
\varepsilon = \sqrt{\frac{\ln(n/\delta)}{2N}} (1-\gamma)^{-1}.
$$

Then, we bound the probability of minimal-error policy among samples has an error more than ε than the

[9] i.e. the optimal polices of MDPs distributed according to the current belief.

expectation:

$$\begin{aligned}&\mathbb{P}\left(\min\{\rho_i \mid i = 1, \ldots, N\} \geq \bar{\rho} + \varepsilon\right)\\ &\leq \mathbb{P}\left(\frac{1}{N}\sum_{i=1}^{N} \rho_i \geq \bar{\rho} + \varepsilon\right) \leq \delta/n,\end{aligned}$$

where the last inequality is from Hoeffding, and the boundedness of rewards. Since there are n such policies, and with a union bound, the probability that any policy has an error of more than ε worse than the expected, is bounded by δ. □

B Root sampling and look-ahead view equivalence

Denote $\mathbb{E}_\beta$ as the expectation under marginals ν and τ. The optimal for Bayesian value function can be calculated by noting the following equivalence relation:

$$\begin{aligned}
&V_t^\pi(s_t, \beta_t) \equiv v_{\beta_t}^\pi(s_t)\\
&= \int_{\mathcal{M}} V_\mu^\pi(s_t)\beta_t(\mu)d\mu\\
&= \int_{\mathcal{M}} r_{t+1}\,\mathbb{P}_\mu(r_{t+1}|s_t, a_t)\beta_t(\mu)d\mu\\
&+ \gamma \int_{\mathcal{M}} \sum_{s' \in s_{t+1}} \mathbb{P}_\mu(s'|s_t, a_t)V_\mu^\pi(s')\beta_t(\mu)d\mu\\
&= \int_{\mathcal{M}} r_{t+1}\,\mathbb{P}_\mu(r_{t+1}|s_t, a_t)\beta_t(\mu)d\mu\\
&+ \gamma \sum_{s' \in s_{t+1}} \int_{\mathcal{M}} V_\mu^\pi(s')\,\mathbb{P}_\mu(s'|s_t, a_t)\beta_t(\mu)d\mu \qquad (7)\\
&= r_{t+1}\tau(r_{t+1}|\omega_t, a_t)\int_{\mathcal{M}} \beta_{t+1}(\mu)d\mu\\
&+ \gamma \sum_{s' \in s_{t+1}} \nu(\omega_{t+1}|\omega_t, a_t)\int_{\mathcal{M}} V_\mu^\pi(s')\beta_{t+1}(\mu)d\mu \qquad (8)\\
&= r_{t+1}\tau(r_{t+1}|\omega_t, a_t) + \gamma \sum_{\omega_{t+1}} v_{\beta_{t+1}}^\pi(s_{t+1})\nu(\omega_{t+1}|\omega_t, a_t)\\
&= \mathbb{E}_\beta[r_{t+1} + \gamma v_{\beta_{t+1}}^\pi(s_{t+1})]\\
&= \mathbb{E}_\beta[r_{t+1} + \gamma V_{t+1}^\pi(s_{t+1}, \beta_{t+1})]
\end{aligned}$$

We obtain Eq.(8) from Eq.(7) using Eq.(2) and the definitions of marginal distributions ν and τ.

C Parameter Selection for Experiments

Here we describe the chosen hyperparameters for each algorithm shown in Table 3. For each algorithm, these are:

1. BAMCP: (depth,no. of simulations): No. of simulations range from 10 to 10^5, or until the environment time-limit is reached. Depth is between {15,50,*auto*}, using the original implementation.

2. SPARSER : (no.of sampled policies,no.of samples per policy,depth parameter K, Horizon). PI is performed upto 1e-4 accuracy, while RTDP performs lookahead planning of depth 15 for all environments, except larger Grid10 and Maze, where depth is set to 50.

3. BFS3: (branching-factor,no.of simulations). Depth is fixed at 15 for all except larger Grid10 and Maze environment, for which it is 50. Branching factor is between {5,10,15} and no. of simulations between {10,10,1000}.

4. SBOSS: (no.of samples,sampling threshold) Cross-validated against {2,4,8,16,32} and {3,5,7} respectively.

D MountainCar

The MountainCar enviroment consists of a vector of two continuous observations, with an episode limit of 200 steps. Figure 3 shows cumulative reward of DSS, Thompson sampling (TS) and Q-learning (QL) algorithm on standard MountainCar environment using the 'OpenAI gym' simulator. We mainly focus on the effects of Discretization, noting that the best performing algorithms in this enviroment are Q-learning and SARSA, with Tile-encoding discretization (refer 'github.com/openai/gym/wiki/Leaderboard'). Note that since discretization of observation-space is done, Alg-16x16 and Alg-20x20 denote the performance of algorithm-'Alg' in an observation-space discretized into 16x16 and 20x20 bins respectively. This can equivalently be though of as Tile-encoding with just a single tile, along with 256 and 400 features each respectively.

The algorithms[10] used are described as follow:

1. DSS: Since the computation-time increases proportionally with (N,M,K,H) parameters, we select a suitable parameter-set with reasonable computation time (0.5 seconds/step). We tried (4,10,10,1) and (4,10,100,1), but lookahead parameter K=10 performed better. The result is thus shown with fixed parameters (4,10,10,1).

2. QL: Denotes Q-learning algorithm. QL-opt denotes optimal Tile-encoding, wih no limit on number of tiles, or features. This is unlike the simple 16x16 and 20x20 **single** tile encoding used for all other algorithms.

[10] Note that we tried SARSA, but its performace was unstable at low levels of discretization.

Algorithm	Chain	DoubleLoop	Grid5	Grid10	Maze
Sparser-RTDP	(8,4,10,2)	(4,4,18,2)	(4,2,50,1)	(4,2,200,2)	(4,4,500,1)
Sparser-PI	(4,4,5,2)	(4,4,18,2)	(2,2,25,1)	(2,2,100,2)	(4,2,100,1)
BAMCP	(auto,100)	(15,100)	(50,10000)	(50,10000)	(50,1000)
BFS3	(10,100)	(10,10)	(10,10)	(5,10)	(5,10)
SBOSS	(8,3)	(2,3)	(2,3)	(2,3)	(2,3)

Table 3: Best parameters obtained from the initial 10 tuning runs.

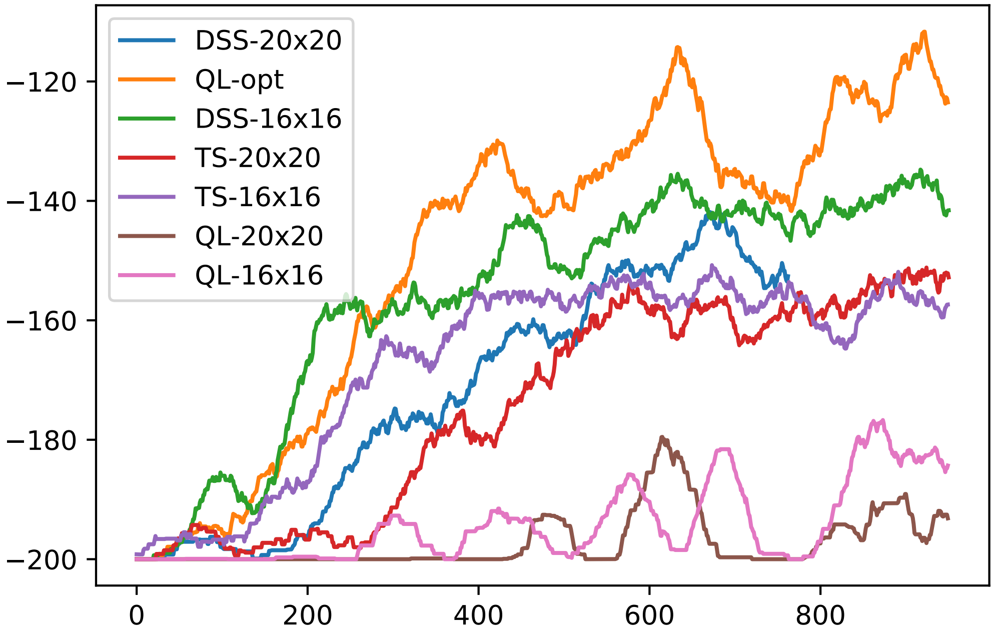


Figure 3: Cumulative reward in MountainCar environment for first 1000 episodes.

3. TS: Denotes Thompson sampling.

D.1 DISCUSSION

First of all, these preliminary results do not represent a rigorous comparison between algorithms, as they are based on the output of a single run, with no parameter tuning. However, it is evident that both Bayesian algorithms learn the enviroment quite fast (less than 1000 episodes). At least equally important is the discretisation used. With a sufficiently fine tile-encoding, even the simplest (QL-opt) can work very well. But on the other hand, for the much simpler discretization of 16x16 and 20x20, the Bayesian algorithms are able to perform much better than their Qlearning counterparts (QL-16x16, QL-20x20). Finally, DSS seems to perform much better than TS. We conjecture that DSS could perform better with an appropriate level of discretization. However, the current version of DSS can't fully utilize Tile-encoding as we would need to maintain a prior over discretisations. We therefore leave this, and all other work with approximate inference, for future work.