Journal of Artificial Intelligence Research 48 (2013) 841-883 Submitted 06/13; published 11/13

Scalable and Efficient Bayes-Adaptive Reinforcement Learning Based on Monte-Carlo Tree Search

Arthur Guez AGUEZ@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
London, WC1N 3AR, UK

David Silver D.SILVER@CS.UCL.AC.UK
Dept. of Computer Science
University College London
London, WC1E 6BT, UK

Peter Dayan DAYAN@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
London, WC1N 3AR, UK

Abstract

Bayesian planning is a formally elegant approach to learning optimal behaviour under model uncertainty, trading off exploration and exploitation in an ideal way. Unfortunately, planning optimally in the face of uncertainty is notoriously taxing, since the search space is enormous. In this paper we introduce a tractable, sample-based method for approximate Bayes-optimal planning which exploits Monte-Carlo tree search. Our approach avoids expensive applications of Bayes rule within the search tree by sampling models from current beliefs, and furthermore performs this sampling in a lazy manner. This enables it to outperform previous Bayesian model-based reinforcement learning algorithms by a significant margin on several well-known benchmark problems. As we show, our approach can even work in problems with an infinite state space that lie qualitatively out of reach of almost all previous work in Bayesian exploration.

1. Introduction

A key challenge in sequential decision-making is to understand how agents can learn to collect rewards — and avoid costs — through interactions with the world. A natural way to characterize these interactions is by a Markov Decision Process (MDP). MDPs consist of a set of states, a set of possible actions, and a transition model that stochastically decides a successor state from a given state and action. In addition, a cost or reward is associated with each state and action. The problem for learning arises when some aspects of the transition model are unknown to the agent, implying uncertainty about the best strategy for gathering rewards and avoiding costs. Exploration is therefore necessary to reduce this uncertainty and ensure appropriate exploitation of the environment. Weighing the benefits of exploring, to identify potentially better actions, against the benefits of exploiting known sources of rewards is generally referred to as the exploration-exploitation trade-off.

The trade-off can be formalized in various different ways. One possible objective is to control the number of suboptimal actions the agent ever performs; algorithms that with high probability can bound the number of such suboptimal steps by a polynomial in the number of states and actions are said to be PAC-MDP (Strehl, Li, & Littman, 2009). Instead of focusing on suboptimal actions, another objective is to minimize the so-called regret, which is the expected loss relative to the optimal policy in the MDP (Jaksch, Ortner, & Auer, 2010). Lastly, Bayesian decision theory prescribes maximizing the expected discounted sum of rewards in the light of a prior distribution over transition models; one way to achieve this is by solving an augmented MDP, called the Bayes-Adaptive MDP (BAMDP), in which the corresponding augmented dynamics are known (Martin, 1967; Duff, 2002). The augmentation is the posterior belief distribution over the dynamics, given the data so far observed. The agent starts in the belief state corresponding to its prior and, by executing the greedy policy in the BAMDP whilst updating its posterior, acts optimally (with respect to its beliefs) in the original MDP. The Bayes-optimal policy is the optimal policy of the BAMDP; it integrates exploration and exploitation in an ideal manner.

In general, these different objectives are not compatible – see for example the work of Kolter and Ng (2009) for the incompatibility between PAC-MDP and the Bayes-optimal solution. PAC-MDP and regret frameworks have gained considerable traction in recent years, while Bayesian exploration has been comparatively ignored. However, the Bayesian framework is attractive because structured prior knowledge can be incorporated into the solution in a principled manner, providing the means to tackle, at least in theory, large and complex unknown environments. Methods tailored for objectives such as PAC-MDP or regret minimization cannot so far easily be adapted to exploit such priors. With no other assumption about the environment, they are thus forced to explore every state and action at least once, which is hopeless in large environments.

Unfortunately, exact Bayesian reinforcement learning (RL) is computationally intractable. Various algorithms have been devised to approximate optimal learning, but often at rather large cost. This computational barrier has restricted Bayesian RL to small domains with simple priors. In this paper, we present a tractable approach that exploits and extends recent advances in Monte-Carlo tree search (MCTS) (Kocsis & Szepesvári, 2006), notably to partially-observable MDPs (Silver & Veness, 2010) of which the BAMDP can be seen as a special case. While MCTS is capable of tackling large MDP problems when the dynamics are known (Gelly, Kocsis, Schoenauer, Sebag, Silver, Szepesvári, & Teytaud, 2012), we show that a naive application of MCTS to the BAMDP is not tractable in general and we propose a set of principled modifications to obtain a practical algorithm, which is called BAMCP for 'Bayes-Adaptive Monte-Carlo Planner'.

At each iteration in BAMCP, as in the POMCP algorithm (Silver & Veness, 2010), a single MDP is sampled from the agent's current beliefs. This MDP is used to simulate a single episode whose outcome is used to update the value of each node of the search tree traversed during the simulation. By integrating over many simulations, and therefore many sample MDPs, the optimal value of each future sequence is obtained with respect to the agent's beliefs. We prove that this process converges to the Bayes-optimal policy, given infinite samples. Since many of the priors that are appropriate in the Bayesian RL setting require some form of approximate inference, we extend the convergence proof to show that BAMCP also converges when combined with a Markov Chain Monte Carlo-based inference scheme. Our algorithm is more efficient than previous sparse sampling methods for Bayes-adaptive planning (Wang, Lizotte, Bowling, & Schuurmans, 2005; Castro, 2007; Asmuth & Littman, 2011), partly because it does not update the posterior belief state during the course of each simulation. It thus avoids repeated applications of Bayes rule, which is expensive for all but the simplest priors over the MDP. To increase computational efficiency further, we introduce an additional innovation: a lazy sampling scheme that only samples the posterior distribution for states traversed during the simulation.

We applied BAMCP to a representative sample of benchmark problems and competitive algorithms from the literature. It consistently and significantly outperformed existing Bayesian RL methods, and also recent non-Bayesian approaches, thus achieving state-of-the-art performance.

Further, BAMCP is particularly well suited to support planning in large domains in which richly structured prior knowledge makes lazy sampling both possible and effective. This offers the prospect of applying Bayesian RL at a realistically complex scale. We illustrate this possibility by showing that BAMCP can tackle a domain with an infinite number of states and a structured prior over the dynamics, a challenging, if not radically intractable, task for existing approaches. This example exploits BAMCP's ability to use Markov chain Monte Carlo methods for inference associated with the posterior distribution over models.

The paper is organized as follows. First, we formally define the Bayesian model-based RL problem and review existing methods; we then present our new algorithm in the context of previous suggestions; and finally we report empirical results on existing domains and on the new, infinite, task. Some of these results appeared in a short conference version of this paper (Guez, Silver, & Dayan, 2012).

2. Model-Based Reinforcement Learning

We first briefly review model-based reinforcement learning and search algorithms in the case that the model is known. We then introduce the formalism of Bayesian model-based RL and provide a survey of existing approximation algorithms that motivate our approach.

2.1 Model-Based Reinforcement Learning with Known Model

An MDP is described as a 5-tuple $M = \langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where S is the discrete set of states, A is the finite set of actions, $\mathcal{P} : S \times A \times S \to \mathbb{R}$ is the state transition probability kernel, $\mathcal{R} : S \times A \to \mathbb{R}$ is a bounded reward function, and γ is the discount factor (Szepesvári, 2010). A deterministic stationary MDP policy π is defined as a mapping $\pi : S \to A$ from states to actions. The *value function* of a policy π at state $s \in S$ is its expected return, defined as:

$$V^\pi(s) \equiv \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right], \tag{1}$$

where r_t is the random reward obtained at time t when following policy π from state s — $\mathbb{E}^\pi$ denotes the expectation operator that averages over all possible paths that policy π implies. A related quantity is the *action-value function* of a policy π for executing a particular action $a \in A$ at state $s \in S$ before executing π:

$$Q^\pi(s,a) \equiv \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s') \mathbb{E}^\pi_M \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_1 = s' \right] \tag{2}$$

$$= \mathcal{R}(s,a) + \gamma \sum_{s' \in S} \mathcal{P}(s,a,s') V^\pi(s'), \tag{3}$$

implying the relation $V^\pi(s) = Q^\pi(s, \pi(s))$. The *optimal* action-value function, denoted Q^*, provides the maximum expected return $Q^*(s,a)$ that can be obtained after executing action a in state s. The optimal value function, V^*, is similarly defined and is related to Q^* as $V^*(s) = \max_{a \in A} Q^*(s,a), s \in S$. An optimal policy π^* achieves the maximum expected return from all states, and can be obtained from Q^* as: $\pi^*(s) = \operatorname{argmax}_{a \in A} Q^*(s,a)$, breaking ties arbitrarily.

When all the components of the MDP tuple are known — including the model $\mathcal{P}$ — standard MDP planning algorithms can be used to estimate the optimal policy off-line, such as Value Iteration or Policy Iteration (Bellman, 1954; Ross, 1983; Sutton & Barto, 1998; Szepesvári, 2010). However, it is not always practical to find the optimal policy for all states in large MDPs in one fell swoop. Instead, there are methods that concentrate on *searching* online for the best action at just the current state s_t. This is particularly common for model-based Bayesian RL algorithms. We therefore introduce relevant existing online search methods for MDPs that are used as building blocks for Bayesian RL algorithms.

2.1.1 Online Search

Online search methods evaluate a tree of possible future sequences. The tree is rooted at the current state and is composed of state and action nodes. Each state node, including the root, has as its children all the actions that are legal from that state. In turn, each action node has as its children all the successor states resulting from that action. The goal of the forward search algorithm is recursively to estimate the value of each state and action node in the tree. Ultimately, the value of each possible action from the root is used to select the next action, and the process repeats using the new state at the root.

Online search methods may be categorised firstly by the *backup* method by which the value of each node is updated, and secondly by the order in which the nodes of the tree are traversed and backups are applied.

2.1.2 Full-Width Search

Classical online search methods are based on *full-width* backups, which consider all legal actions and all possible successor states, for example using a Bellman backup,

$$V(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') V(s') \tag{4}$$

Search efficiency is then largely determined by the order in which nodes are traversed. One example is 'best-first', for which the current best is usually determined according to an optimistic criterion. This leads to an algorithm resembling A* (Hart, Nilsson, & Raphael, 1968), which applies in the deterministic case. The search tree may also be truncated, using knowledge of the most extreme reward and the discount factor to ensure that this is provably benign (Davies, Ng, & Moore, 1998). If one is prepared to give up guarantees on optimality, an approximate value function (typically described in the online search literature as a heuristic function or evaluation function) can be applied at leaf nodes to substitute for the value of the truncated subtree.

2.1.3 Sample-Based Search

Rather than expanding every tree node completely, sample-based search methods overcome the curse of dimensionality by just sampling successor states from the transition distribution. These have the generic advantage over full-width search that they expend little effort on unlikely paths in the tree.

Sparse Sampling

Sparse Sampling (Kearns, Mansour, & Ng, 1999) is a sample-based online search algorithm. The key idea is to sample C successor nodes from each action node, and apply a Bellman backup to these sampled transitions, so as to update the value of the parent state node from the values of the child nodes:

$$V_d(s) = \max_{a \in \mathcal{A}} \mathcal{R}(s,a) + \frac{\gamma}{C} \sum_{i=1}^{C} V_{d+1}(\text{child}(s,a,i)). \tag{5}$$

The search tree is traversed in a depth-first manner, and an approximate value function is employed at truncated leaf nodes, after some pre-defined depth D. Sparse Sampling converges to a near-optimal policy given an appropriate choice of the parameters C and D.

FSSS

Although Sparse Sampling concentrates on likely transitions, it does not focus search on nodes that have relatively high values or returns. In the work of Walsh, Goschin, and Littman (2010), Forward Search Sparse Sampling (FSSS) extends regular Sparse Sampling by maintaining both lower and upper bounds on the value of each node:

$$L_d(s,a) = \mathcal{R}(s,a) + \frac{\gamma}{C} \sum_{s' \in \text{Child}(s,a)} L_{d+1}(s')\text{Count}(s,a,s'), \tag{6}$$

$$U_d(s,a) = \mathcal{R}(s,a) + \frac{\gamma}{C} \sum_{s' \in \text{Child}(s,a)} U_{d+1}(s')\text{Count}(s,a,s'), \tag{7}$$

$$L_d(s) = \max_{a \in \mathcal{A}} L_d(s,a), \tag{8}$$

$$U_d(s) = \max_{a \in \mathcal{A}} U_d(s,a), \tag{9}$$

where $\text{Child}(s,a)$ is the set of successor states sampled from C draws of $\mathcal{P}(s,a,\cdot)$, and $\text{Count}(s,a,s')$ is the number of times each set element was sampled. Whenever a node is created, the lower and upper bounds are initialized according to $L_d(s,a) = V_{\min}$ and

$U_d(s,a) = V_{\max}$, i.e., the worst and best possible returns. The tree is traversed in a best-first manner according to these value bounds, starting from the root for each simulation through the tree. At each state node, a promising action is selected by maximising the upper bound on value. At each action (or chance) node, successor states are selected from a sampled set of C candidates by maximising the uncertainty (upper minus lower bound). This effectively prunes branches of the tree that have low upper bounds before they are exhaustively explored, while still maintaining the theoretical guarantees of Sparse Sampling.

Monte-Carlo Tree Search

Despite their theoretical guarantees, in practice, sparse sampling and FSSS both suffer from the fact that they truncate the search tree at a particular depth, and so experience bias associated with the approximate value function they use at the leaves. Monte-Carlo Tree Search (MCTS) provides a way of reducing the bias by evaluating leaves exactly using the model, but employing a sub-optimal, rollout policy. More formally, in MCTS, states are evaluated by averaging over many simulations. Each simulation starts from the root and traverses the current tree until a leaf is reached, using a *tree policy* (e.g., greedy action selection) based on information that has so far been gathered about nodes in the tree. This results in a (locally) best-first tree traversal, where at each step the tree policy selects the best child (best according to some exploration criterion) given the current values in the tree. Rather than truncating the search and relying on a potentially biased value function at leaf nodes, a different policy, called a *rollout policy* (e.g., uniform random) is employed from the leaf node until termination or a search horizon. Each node traversed by the simulation is then updated by a Monte-Carlo backup, which simply evaluates that node by the mean outcome of all simulations that passed through that node. Specifically, the Monte-Carlo backups update the value of each action node as follows:

$$Q_d(s,a) \leftarrow Q_d(s,a) + {}^{(R - Q_d(s,a))}/_{N_d(s,a)}, \tag{10}$$

where R is the sampled discounted return obtained from the traversed action node s,a at depth d and $N_d(s,a)$ is the visitation count for the action node s,a (i.e., the update computes the mean of the sampled returns obtained from that action node over the simulations).

A particular tree policy for MCTS that has received much attention, and indeed underlies our algorithm, is the UCT (Upper Confidence bounds applied to Trees) policy (Kocsis & Szepesvári, 2006). UCT employs the UCB1 (Upper Confidence Bounds) algorithm (Auer, Cesa-Bianchi, & Fischer, 2002), designed for multi-armed bandit problems, to select adaptively between actions at every state node according to:

$$\operatorname*{argmax}_{a \in \mathcal{A}} Q_d(s,a) + c\sqrt{{}^{\log(N_d(s))}/_{N_d(s,a)}}, \tag{11}$$

where c is an exploration constant that needs to be set appropriately and $N_d(s)$ is the visitation count for the state node s. This tree policy treats the forward search as a meta-exploration problem, preferring to exploit regions of the tree that currently appear better than others, while continuing to explore unknown or less known parts of the tree. This leads to good empirical results even for small numbers of simulations, because effort is expended where search seems fruitful. Nevertheless all parts of the tree are eventually visited infinitely often, and therefore the algorithm can be shown to converge to the optimal policy in the very long run.

Despite some negative theoretical results showing that UCT can be slow to find optimal policies in carefully designed counterexample MDPs (Coquelin & Munos, 2007), UCT has been successful in many large MDP domains (Gelly et al., 2012).

2.2 Model-Based Bayesian Reinforcement Learning

The methods we have so far discussed depend on the agent having a model of the world, i.e., the dynamics $\mathcal{P}$. The key concern for Bayesian RL is acting when this model is not fully known. We first describe the generic Bayesian formulation of optimal decision-making in an unknown MDP, following Martin (1967) and Duff (2002), and then consider approximations inspired by the intractability of the full problem.

2.2.1 The Formalism

Given that the dynamics $\mathcal{P} \in \mathbb{P}$ (coming from the set of all possible models) are only incompletely known, Bayesian RL treats them as a latent random variable which follows a prior distribution $P(\mathcal{P})$. Observations about the dynamics contained in the history h_t (at time t) of actions and states: $h_t \equiv s_1 a_1 s_2 a_2 \dots a_{t-1} s_t$, duly lead to a posterior distribution over $\mathcal{P}$ via a likelihood.

The history h_t influences the posterior distribution. Thus policies $\tilde{\pi}$ that integrate exploration and exploitation (called EE policies) for a Bayesian RL problem will generically have to take this history into account, along with the current state, in order to specify what action to take. That is, whereas when $\mathcal{P}$ is known, a policy π can be defined as a mapping $\pi : S \times A \to [0,1]$ from just the current state and actions to a probability (of execution), for Bayesian RL, EE policies are defined as mappings from history, current state, and action to a probability $\tilde{\pi} : S \times \mathcal{H} \times A \to [0,1]$, where $\mathcal{H}$ is the set of possible histories.[1] We denote by $\tilde{\Pi}$ the set of all EE policies.

The objective for an EE policy under the Bayesian formulation is to maximize the expected return (sum of discounted rewards), where the expectation is taken over the distribution of environments $P(\mathcal{P}) = P(\mathcal{P}\,|\emptyset)$, in addition to taking the usual expectation over the stochasticity of the return induced by the dynamics. Formally, we define this expected discounted return v starting from a state s after seeing history h when following an EE policy $\tilde{\pi}$ as:

$$v(s,h,\tilde{\pi}) = \mathbb{E}^{\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, h_0 = h\right] \tag{12}$$

$$= \int_{\mathcal{P}} \mathrm{d}\mathcal{P}\, P(\mathcal{P}\,|h)\, \mathbb{E}^{\tilde{\pi}}_{M(\mathcal{P})}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, h_0 = h\right] \tag{13}$$

$$= \sum_{a_0 \in A} \mathcal{R}(s,a_0) + \gamma \sum_{s' \in S} \tilde{\pi}(s,h,a_0) v(s', h a_0 s', \tilde{\pi}) \bar{\mathcal{P}}(s,a_0,s',h), \tag{14}$$

1. The redundancy in the state-history notation throughout this paper, namely that the current state could be extracted from the history, is only present to ensure clarity of exposition.

where $\bar{\mathcal{P}}(s, a, s', h) \equiv \int_{\mathcal{P}} \mathrm{d}\mathcal{P}\, P(\mathcal{P}\,|h)\, \mathcal{P}(s, a, s')$ denotes the probability of transitioning from state s to s' after executing a under a distribution of dynamics $P(\mathcal{P}\,|h)$, and $M(\mathcal{P})$ denotes the MDP associated with dynamics $\mathcal{P}$.

Definition 1 *Given S, A, $\mathcal{R}$, γ, and a prior distribution $P(\mathcal{P})$ over the dynamics of the* MDP *M, let*

$$v^*(s, \emptyset) = \sup_{\tilde{\pi} \in \tilde{\Pi}} v(s, \emptyset, \tilde{\pi}). \tag{15}$$

Martin (1967, Thm. 3.2.1) shows that there exists a strategy $\tilde{\pi}^* \in \tilde{\Pi}$ that achieves that expected return (i.e., $v(s, \emptyset, \tilde{\pi}^*) = v^*(s, \emptyset)$). Any such EE strategy $\tilde{\pi}^*$ is called a ***Bayes-optimal policy****.*[2]

This formulation prescribes a natural recipe for computing the Bayes-optimal policy. After observing history h_t from the MDP, the posterior belief over $\mathcal{P}$ is updated using Bayes' rule $P(\mathcal{P}|h_t) \propto P(h_t|\mathcal{P})P(\mathcal{P})$ (or in recursive form $P(\mathcal{P}|h_t) \propto \mathcal{P}(s_{t-1}, a_{t-1}, s_t)P(\mathcal{P}|h_{t-1})$). Thus, the uncertainty about the dynamics of the model can be transformed into certainty about the current state inside an augmented state space $S^+ = S \times \mathcal{H}$, where S is the state space in the original problem and $\mathcal{H}$ is the set of possible histories. The dynamics associated with this augmented state space are described by

$$\mathcal{P}^+(\langle s, h\rangle, a, \langle s', h'\rangle) = \mathbb{1}[h' = has'] \int_{\mathcal{P}} \mathcal{P}(s, a, s')P(\mathcal{P}|h)\, \mathrm{d}\mathcal{P}, \tag{16}$$

and the reward function is simply the projected reward function in the original MDP:

$$\mathcal{R}^+(\langle s, h\rangle, a) = \mathcal{R}(s, a). \tag{17}$$

Together, the 5-tuple $M^+ = \langle S^+, A, \mathcal{P}^+, \mathcal{R}^+, \gamma\rangle$ forms the Bayes-Adaptive MDP (BAMDP) for the MDP problem M. Since the dynamics of the BAMDP are *known*, it can in principle be solved to obtain the optimal value function associated with each action:

$$Q^*(\langle s_t, h_t\rangle, a) = \max_{\tilde{\pi}} \mathbb{E}^{\tilde{\pi}}_{M^+}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} | a_t = a\right] \tag{18}$$

from which the optimal action for each state can be readily derived. Optimal actions in the BAMDP are executed greedily in the real MDP M and constitute the best course of action for a Bayesian agent with respect to its prior belief over $\mathcal{P}$:

Proposition 1 (Silver, 1963; Martin, 1967) *The optimal policy of the* BAMDP *is the Bayes-optimal policy, as defined in Definition 1.*

It is obvious that the expected performance of the BAMDP policy in the MDP M is bounded above by that of the optimal policy obtained with a fully-observable model, with

2. The proof by Martin (1967) only covers finite state spaces, but it can be extended to specific kinds of infinite state spaces such as the one we consider in Section 4.2, see Appendix D for details.

equality occurring, for example, in the degenerate case in which the prior only has support on the true model.

The Bayes-optimal policy is stationary as a function of the augmented state, but evolves over time when observed in the context of the original MDP — as a function of the state in S only. Since the uncertainty about the dynamics is taken into account in the optimization of the return, the Bayes-optimal policy integrates exploration and exploitation optimally.

It can be useful to observe that the BAMDP is a particular form of Partially Observable MDP (POMDP). The state space of this POMDP is $S \times \mathbb{P}$, where $\mathbb{P}$ is the set of all possible models $\mathcal{P}$. The second component of the state space is static and hidden, and partially observed through experienced transitions. Planning can be conducted in the belief space, or equivalently in the space of sufficient statistics of the belief distribution, allowing decisions to be taken in the light of their likely outcomes in gathering exploitable information about the hidden state. In the case of BAMDP, such actions gather information about the hidden model $\mathcal{P}$. However, the POMDP is not a discrete POMDP since its state space is continuous (with discrete observations). Therefore, as pointed out by Duff (2002), many classical solutions to POMDPs cannot be directly applied to the BAMDP.

From a practical perspective, solving the BAMDP exactly is computationally intractable, even for small state spaces. First, the augmented state space contains all possible histories and is therefore infinite. Second, the transitions of the BAMDP, described in Equation 16, require an integration of transition models over the posterior. Although this operation can be trivial for some simple probabilistic models (e.g., independent Dirichlet-Multinomial), it is intractable for most priors of interest (see Section 4.2 for an example). However, certain special cases of the BAMDP are known to be somewhat more tractable. For example, the celebrated Gittins indices provide a shortcut solution for bandit problems (Gittins, Weber, & Glazebrook, 1989), although calculating these indices remains a challenge in general. Further, the optimal solution to at least some finite-horizon linear-Gaussian control problems can be computed exactly (Tonk & Kappen, 2010). Nevertheless, it appears unlikely that there exists a tractable exact algorithm to solve general BAMDPs, justifying a search for sound and efficient approximations.

2.2.2 Approximate Bayes-Adaptive Algorithms

Three coarse classes of approximation methods have been developed, which we now review. Note that all of them have analogues in solution methods for POMDPs.

First are offline methods that toil mightily to provide execution policies that can be used for any observed augmented state. Second and third are two sets of online methods that concentrate on just the current augmented state. One set of methods uses sparse sampling in the full tree of future states and actions associated with the BAMDP, starting from the current augmented state. The other samples and solves one or more MDPs from the current posterior over $\mathcal{P}$, possibly correcting for the bias towards exploitation to which this typically leads.

After describing these classes, we highlight what they currently lack, and so establish the basis for our new algorithm, BAMCP.

Offline Methods

One idea is to solve the entire BAMDP offline, for every state and belief (or history). This obviates the need for anything other than a simple value/policy lookup during execution. However, this avenue for approximation has not led to much practical success — presumably because of the difficulties associated with the size of the BAMDP, including the fact that gargantuan amounts of computation may be performed to find good policies in parts of the space of histories that are actually not sampled in practice.

Existing approaches in this class include an actor-critic algorithm (Duff, 2003), which does learning, and a point-based value iteration algorithm, called BEETLE (Bayesian Exploration Exploitation Tradeoff in LEarning) (Poupart, Vlassis, Hoey, & Regan, 2006). BEETLE builds an approximate policy off-line by exploiting facets of the structure of the value functions for BAMDPs, which they inherit from their broader, parent, class of POMDPs. More recently, Wang, Won, Hsu, and Lee (2012) propose to solve an offline POMDP by representing the latent dynamics as a discrete partially-observed state component, where the value of this state component corresponds to one of K possible models sampled from the prior. Their approach can fail if the true model is not well-represented in these K sampled models.

Offline methods are particularly poorly suited to problems such as the infinite state task we consider in section 4.2.

Online Methods: Sparse Sampling

Online methods reduce the dependency on the size of the BAMDP by approximating the BAMDP solution around the current (augmented) state of the agent and running a planning algorithm at each step.

One idea is to perform forms of forward search from the current state. Although these methods concentrate on the current state, the search tree is still large and it can be expensive to evaluate a given path in the tree. In partial alleviation of this problem, most approaches rely on some form of sparse, non-uniform, tree exploration to minimize the search effort (but see also Fonteneau, Busoniu, & Munos, 2013). While Section 2.1.3 described search algorithms for MDPs, here we present existing extensions to the BAMDP setting. Analogous methods for POMDPs are reviewed by Ross, Pineau, Paquet, and Chaib-Draa (2008).

Wang et al. applied Sparse Sampling to search online in BAMDPs (Wang et al., 2005), expanding the tree non-uniformly according to sampled trajectories. At each state node, a promising action is selected via Thompson sampling (Thompson, 1933; Agrawal & Goyal, 2011) — i.e., sampling an MDP from the belief-state, solving the MDP and taking the optimal action. As in Sparse Sampling, this fails to exploit information about the values of nodes in prioritizing the sampling process. At each chance (action) node, a successor belief-state is sampled from the transition dynamics of the BAMDP.

Castro et al. applied Sparse Sampling to define a relevant region of the BAMDP for the current decision step. This leads to an optimization problem that is solved using Linear Programming (Castro & Precup, 2007).

Asmuth and Littman's BFS3 algorithm (Asmuth & Littman, 2011) adapts Forward Search Sparse Sampling (Walsh et al., 2010) to the BAMDP (treated as a particular MDP). Although BFS3 is described as Monte-Carlo tree search, it in fact uses a Bellman backup rather than Monte-Carlo evaluation. As in FSSS, each Bellman backup updates both lower and upper bounds on the value of each node.

Online Methods: Dual Optimism

Instead of applying sparse sampling methods in the tree of future states and actions, an alternative collection of methods derives one or more simpler MDPs from the posterior at a current augmented state, whose solution is often computationally straightforward. By itself, this leads to over-exploitation: corrections are thus necessary to generate sufficient exploration. Exploration can be seen as coming from optimism in the face of uncertainty – actions that have yet to be tried sufficiently must look more attractive than their current mean. Indeed, there are various heuristic forms of exploration bonus (Sutton, 1990; Schmidhuber, 1991; Dayan & Sejnowski, 1996; Kearns et al., 1999; Meuleau & Bourgine, 1999; Brafman & Tennenholtz, 2003) that generalize the optimism inherent in optimal solutions such as Gittins indices.

One such approximation was first derived in the work of Cozzolino, Gonzalez-Zubieta, and Miller (1965), where the mean estimate of the transition probabilities (i.e., the mean of the posterior) was employed as a certainty equivalence approximation. Solving the corresponding mean MDP induces some form of optimism, but it is not always sufficient to drive exploration. This idea was revisited and linked to reinforcement learning formulations by Dayan and Sejnowski (1996).

Another way to induce optimism is to exploit the variance in the posterior when sampling MDPs at an augmented state. One of these approaches is the Bayesian DP algorithm (Strens, 2000). At each step (or after every couple of steps), a single model is sampled from the posterior distribution over transition models, and the action that is optimal in that model is executed. Although a popular approach in practice, this algorithm does not have a known theoretical guarantee relating it to the Bayes-optimal solution. In the Bandit case, this reduces to Thompson Sampling. Optimism is generated because solving posterior samples is likely to yield optimistic values in some unknown parts of the MDP (where posterior entropy is large) and that will force the agent to visit these regions. The Best Of Sampled Set (BOSS) algorithm generalizes this idea (Asmuth, Li, Littman, Nouri, & Wingate, 2009). BOSS samples a number of models from the posterior and combines them optimistically. This drives sufficient exploration to guarantee some finite-sample performance guarantees, but these theoretical guarantees cannot be easily related to the Bayes-optimal solution. BOSS is quite sensitive to its parameter that governs the sampling criterion, which is unfortunately difficult to select. Castro and Precup proposed a variant, referred to as SBOSS, which provides a more effective adaptive sampling criterion (Castro & Precup, 2010).

One can also see certain non-Bayesian methods in this light. For instance, Bayesian Exploration Bonus (BEB) solves the posterior mean MDP, but with an additional reward bonus that depends on visitation counts (Kolter & Ng, 2009). This bonus is tailored such that the method satisfies the so-called PAC-BAMDP property, which generalizes the PAC-MDP mentioned in the introduction, and implies limiting to a polynomial factor the number of steps in which the EE policy is different than the Bayes-optimal policy. A more recent approach is the BOLT algorithm, which merges ideas from BEB and BOSS, enforces optimism in the transitions by (temporarily) adding fictitious evidence that currently poorly-known actions lead to currently poorly-known states (Araya-López, Buffet, & Thomas, 2012). BOLT also has the PAC-BAMDP property.

Discussion of Existing Methods

Despite the recent progress in approximation algorithms, tackling large domains with complex structured priors remains out of computational reach for existing Bayesian RL methods. Unfortunately, it is exactly in these structured domains that Bayesian methods should shine, since they have the statistical capacity to take advantage of the priors.

Methods that tackle the BAMDP directly such as forward-search methods, suffer from the repeated computation of the BAMDP dynamics inside the search tree for most priors. That is, to compute a single BAMDP transition in Equation 16, one needs to apply Bayes' rule and perform an integration over all possible models. This can be done cheaply for simple priors, but can be rather expensive for arbitrary priors.

On the other hand, optimism-based methods are attractive because they appear more tractable — since they are dealing with smaller MDPs. However, it turns out to be hard to translate sophisticated prior knowledge into the form of a bonus — existing methods are only compatible with simple Dirichlet-Multinomial models. Moreover, the behavior in the early steps of exploration can be very sensitive to the precise parameter inducing the optimism.

We therefore developed an approximation algorithm for Bayesian RL that is compatible with complex priors, while maintaining efficiency and (Bayesian) soundness, so that large EE tasks can be approached in a principled way. Our approach adapts the POMCP Monte-Carlo tree search algorithm (Silver & Veness, 2010), which avoids expensive applications of Bayes rule by only sampling at the root of the tree. It also extends the approach by introducing a novel scheme for lazy sampling. This makes it possible to search locally in finite portions of large or even infinite domains.

3. The BAMCP Algorithm

To reiterate, the goal of a BAMDP planning method is to find, for each decision point $\langle s, h\rangle$ encountered, the action a that at least approximately maximizes the future expected return (i.e., find an optimal EE policy $\tilde{\pi}^*(s, h)$). Our algorithm, Bayes-Adaptive Monte-Carlo Planning (**BAMCP**), does this by performing a forward-search in the space of possible future histories of the BAMDP using a tailored Monte-Carlo tree search.

We employ the UCT algorithm, as presented in Section 2.1.3, to allocate search effort to promising branches of the state-action tree, and use sample-based rollouts to provide value estimates at each node. For clarity, let us denote by Bayes-Adaptive UCT (BA-UCT) the algorithm that applies vanilla UCT to the BAMDP (i.e., the particular MDP with dynamics described in Equation 16).[3] Sample-based search in the BAMDP using BA-UCT requires the generation of samples from $\mathcal{P}^+$ for every step of each simulation — an expensive procedure for all but the simplest generative models $P(\mathcal{P})$. We avoid this cost by only sampling a single transition model $\mathcal{P}^i$ from the posterior at the root of the search tree at the start of

3. While using UCT to solve BAMDPs is mentioned by Asmuth and Littman (2011), we are not aware of any published work that evaluated the performance of BA-UCT.

each simulation i, and using $\mathcal{P}^i$ to generate all the necessary samples during this simulation. Sample-based tree search then acts as a filter, ensuring that the correct distribution of state successors is obtained at each of the tree nodes, as if it was sampled from $\mathcal{P}^+$. This root sampling method was originally introduced in the POMCP algorithm (Silver & Veness, 2010), developed to solve discrete-state POMDPs.

Combining BA-UCT with a version of root sampling forms the basis of the proposed BAMCP algorithm; this is detailed in Section 3.1. In addition, BAMCP also takes advantage of lazy sampling to reduce sampling complexity at the root, this is detailed in Section 3.2. Finally, BAMCP integrates rollout learning to improve the rollouts online, this is detailed in Section 3.3. In Section 3.4, we show that BAMCP converges to the Bayes-optimal solution. Subsequent sections provide empirical results.

3.1 BA-UCT with Root Sampling

The root node of the search tree at a decision point represents the current state of the BAMDP. The tree is composed of state nodes representing belief states $\langle s, h\rangle$ and action nodes representing the effect of particular actions from their parent state node. The visit counts: $N(\langle s, h\rangle)$ for state nodes, and $N(\langle s, h\rangle, a)$ for action nodes, are initialized to 0 and updated throughout search. A value, $Q(\langle s, h\rangle, a)$, which is initialized to 0, is also maintained for each action node.

Each simulation traverses the tree without backtracking by following the UCT policy at state nodes defined by $\operatorname{argmax}_a Q(\langle s, h\rangle, a) + c\sqrt{\log(N(\langle s, h\rangle))/N(\langle s, h\rangle, a)}$, where c is an exploration constant that needs to be set appropriately. Given an action, the transition distribution $\mathcal{P}^i$ corresponding to the current simulation i is used to sample the next state. That is, at action node $(\langle s, h\rangle, a)$, s' is sampled from $\mathcal{P}^i(s, a, \cdot)$, and the new state node is set to $\langle s', has'\rangle$.

When a simulation reaches a leaf, the tree is expanded by attaching a new state node with its connected action nodes, and a rollout policy π_{ro} is used to control the MDP defined by the current $\mathcal{P}^i$. This policy is followed to some fixed total depth (determined using the discount factor). The rollout provides an estimate of the value $Q(\langle s, h\rangle, a)$ from the leaf action node. This estimate is then used to update the value of all action nodes traversed during the simulation: if R is the sampled discounted return obtained from a traversed action node $(\langle s, h\rangle, a)$ in a given simulation, then we update the value of each action node to $Q(\langle s, h\rangle, a) + (R - Q(\langle s, h\rangle, a))/N(\langle s, h\rangle, a)$ (i.e., the mean of the sampled returns obtained from that action node over the simulations).

A detailed description of the BAMCP algorithm is provided in Algorithm 1. A diagram example of BAMCP simulations is presented in Figure 1. In Section 3.4, we show BAMCP eventually converges to the Bayes-optimal policy.

Finally, note that the history of transitions h is generally not the most compact sufficient statistic of the belief in fully observable MDPs. It can, for instance, be replaced with unordered transition counts ψ, considerably reducing the number of states of the BAMDP and, potentially the complexity of planning. BAMCP can search in this reduced search space, which takes the form of an expanding lattice rather than a tree. We found this version of BAMCP to offer only a marginal improvement. This is a common finding for MCTS, stemming from its tendency to concentrate search effort on one of several equivalent paths (up

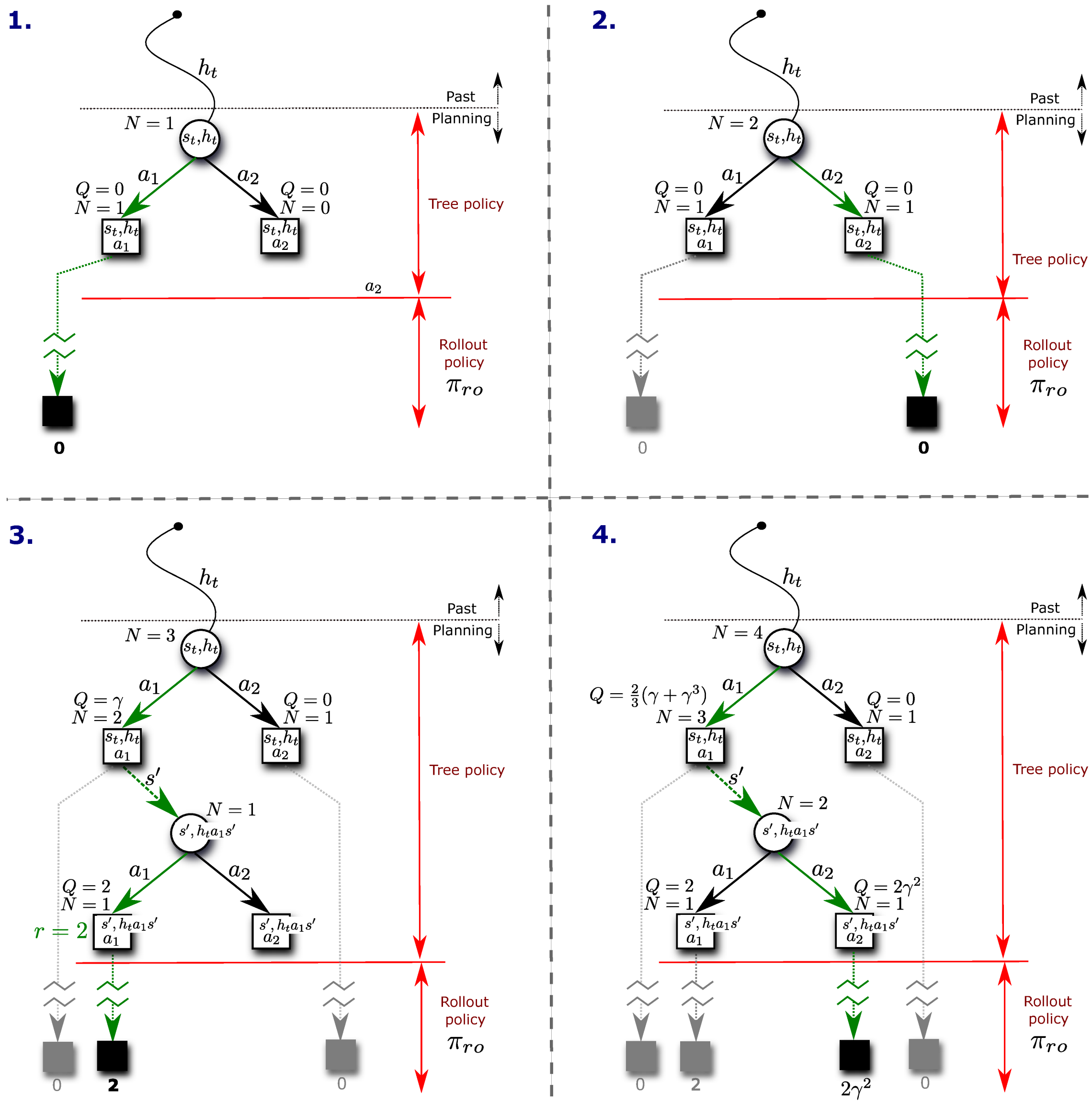


Figure 1: This diagram presents the first 4 simulations of BAMCP in an MDP with 2 actions from state $\langle s_t, h_t \rangle$. The rollout trajectories are represented with dotted lines (green for the current rollouts, and greyed out for past rollouts). **1.** The root node is expanded with two action nodes. Action a_1 is chosen at the root (random tie-breaking) and a rollout is executed in $\mathcal{P}^1$ with a resulting value estimate of 0. Counts $N(\langle s_t, h_t \rangle)$ and $N(\langle s_t, h_t \rangle, a_1)$, and value $Q(\langle s_t, h_t \rangle, a_1)$ get updated. **2.** Action a_2 is chosen at the root and a rollout is executed with value estimate 0. Counts and value get updated. **3.** Action a_1 is chosen (tie-breaking), then s' is sampled from $\mathcal{P}^3(s_t, a_1, \cdot)$. State node $\langle s', h_t a_1 s' \rangle$ gets expanded and action a_1 is selected, incurring a reward of 2, followed by a rollout. **4.** The UCB rule selects action a_1 at the top, the successor state s' is sampled from $\mathcal{P}^4(s_t, a_1, \cdot)$. Action a_2 is chosen from the internal node $\langle s', h_t a_1 s' \rangle$, followed by a rollout using $\mathcal{P}^4$ and π_{ro}. A reward of 2 is obtained after 2 steps from that tree node. Counts for the traversed nodes are updated and the MC backup updates $Q(\langle s', h_t a_1 s' \rangle, a_1)$ to $R = 0 + \gamma 0 + \gamma^2 2 + \gamma^3 0 + \cdots = 2\gamma^2$ and $Q(\langle s_t, h_t \rangle, a_1)$ to $\gamma + {}^{2\gamma^3 - \gamma}/_{3} = \frac{2}{3}(\gamma + \gamma^3)$.

Algorithm 1: BAMCP

procedure `Search(` $\langle s,h \rangle$ `)`
  repeat
    $\mathcal{P} \sim P(\mathcal{P}|h)$
    `Simulate(`$\langle s,h \rangle, \mathcal{P}, 0$`)`
  until `Timeout()`
  return $\operatorname{argmax}_a Q(\langle s,h \rangle, a)$
end procedure

procedure `Rollout(`$\langle s,h \rangle, \mathcal{P}, d$ `)`
  if $\gamma^d R_{\max} < \epsilon$ **then**
    return 0
  end
  $a \sim \pi_{ro}(\langle s,h \rangle, \cdot)$
  $s' \sim \mathcal{P}(s,a,\cdot)$
  $r \leftarrow \mathcal{R}(s,a)$
  return
  $r + \gamma$`Rollout(`$\langle s', has' \rangle, \mathcal{P}, d{+}1$`)`
end procedure

procedure `Simulate(` $\langle s,h \rangle, \mathcal{P}, d$`)`
  if $\gamma^d R_{\max} < \epsilon$ **then return** 0
  if $N(\langle s,h \rangle) = 0$ **then**
    for all $a \in A$ **do**
      $N(\langle s,h \rangle, a) \leftarrow 0,$
      $Q(\langle s,h \rangle, a)) \leftarrow 0$
    end
    $a \sim \pi_{ro}(\langle s,h \rangle, \cdot)$
    $s' \sim \mathcal{P}(s,a,\cdot)$
    $r \leftarrow \mathcal{R}(s,a)$
    $R \leftarrow r + \gamma$ `Rollout(`$\langle s', has' \rangle, \mathcal{P}, d$`)`
    $N(\langle s,h \rangle) \leftarrow 1,\ N(\langle s,h \rangle, a) \leftarrow 1$
    $Q(\langle s,h \rangle, a) \leftarrow R$
    return R
  end
  $a \leftarrow \operatorname{argmax}_b Q(\langle s,h \rangle, b) + c\sqrt{\frac{\log(N(\langle s,h \rangle))}{N(\langle s,h \rangle, b)}}$
  $s' \sim \mathcal{P}(s,a,\cdot)$
  $r \leftarrow \mathcal{R}(s,a)$
  $R \leftarrow r + \gamma$ `Simulate(`$\langle s', has' \rangle, \mathcal{P}, d{+}1$`)`
  $N(\langle s,h \rangle) \leftarrow N(\langle s,h \rangle) + 1$
  $N(\langle s,h \rangle, a) \leftarrow N(\langle s,h \rangle, a) + 1$
  $Q(\langle s,h \rangle, a) \leftarrow Q(\langle s,h \rangle, a) + \frac{R - Q(\langle s,h \rangle, a)}{N(\langle s,h \rangle, a)}$
  return R
end procedure

to transposition), implying a limited effect on performance of reducing the number of those paths.

Note that an algorithm very similar to BA-UCT with root sampling also appeared in the work of Vien and Ertel (2012), shortly after BAMCP was originally published by Guez et al. (2012).

3.1.1 Root Sampling at Work in a Simple Example

We illustrate the workings of BAMCP, in particular root sampling, in a simulated example that showcases a crucial component of Bayes-adaptivity.

Consider a simple prior distribution on two MDPs ($\mathcal{P}_0$ and $\mathcal{P}_1$), illustrated in Figure 2, where $P(\mathcal{P} = \mathcal{P}_0) = P(\mathcal{P} = \mathcal{P}_1) = \frac{1}{2}$. The MDPs are episodic and stop at the leaves, and an episode starts in s_0. From state s_1 or s_2, any action has an expected reward of 0 *under the prior distribution over* MDP*s*. Nevertheless, the outcome of a transition from action a_0 in state s_0 carries information about the identity of the MDP, and allows a Bayes-adaptive agent to take an informed decision in state s_1 or s_2. Using Bayes-rule, we have that $P(\mathcal{P} = \mathcal{P}_0 \,|s_0 a_0 s_1) \propto P(s_0 a_0 s_1 |\, \mathcal{P} = \mathcal{P}_0) P(\mathcal{P} = \mathcal{P}_0) = 0.8$.

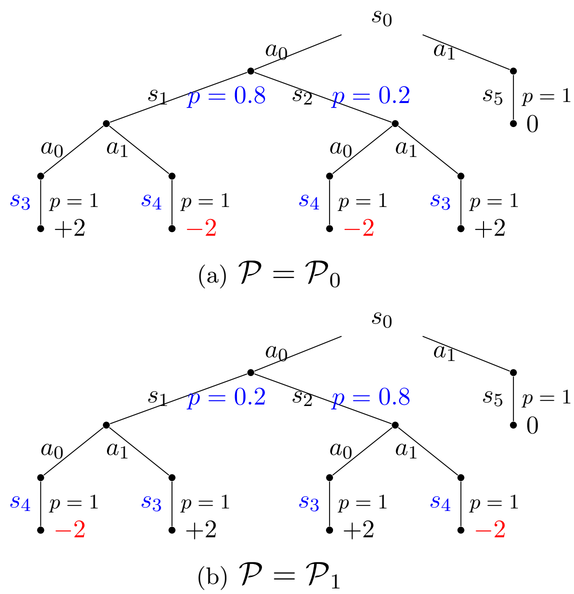


Figure 2: The two MDPs of Section 3.1.1, with prior probability $P(\mathcal{P} = \mathcal{P}_0) = P(\mathcal{P} = \mathcal{P}_1) = \frac{1}{2}$. Differences between the two MDPs are highlighted in blue.

We can therefore compute the optimal values:

$$\begin{aligned} V^*(h = s_0 a_0 s_1) &= \max \begin{cases} 2P(\mathcal{P} = \mathcal{P}_0 \,|h) - 2P(\mathcal{P} = \mathcal{P}_1 \,|h) \\ 2P(\mathcal{P} = \mathcal{P}_1 \,|h) - 2P(\mathcal{P} = \mathcal{P}_0 \,|h) \end{cases} \\ &= 2 \cdot 0.8 - 2 \cdot 0.2 = 1.2 \;\; (= V^*(h = s_0 a_0 s_2)) \\ V^*(h = s_0) &= \max\{0, 1.2\gamma\} = 1.2\gamma. \end{aligned} \tag{19}$$

We now simulate BAMCP on this simple example for the first decision in state s_0. With root sampling, BAMCP only samples either $\mathcal{P}_0$ or $\mathcal{P}_1$ with equal probability at the root of the tree, and does not perform any explicit posterior update inside the tree. Yet, as suggested by Lemma 1, we expect to find the correct distribution $P(\mathcal{P} = \mathcal{P}_0 \,|s_0 a_0 s_1)$ of samples of $\mathcal{P}$ at the tree node $\langle s_0 a_0 s_1 \rangle$. Moreover, BAMCP should converge to the optimal values V^* according to Theorem 1. This is what is observed empirically in Figure 3.

In the second row of Figure 3, we observe that $\hat{Q}(s_0 a_0 s_1, a_1)$ is slower to converge compared to other values. This is because time is ticking more slowly for this non-optimal node (i.e., a small fraction of simulations reach this node) so the value stays put for many simulations.

3.2 Lazy Sampling

In previous work on sample-based tree search, indeed including POMCP (Silver & Veness, 2010), a complete sample state is drawn from the posterior at the root of the search tree. However, this can be computationally very costly. Instead, we sample $\mathcal{P}$ lazily, generating

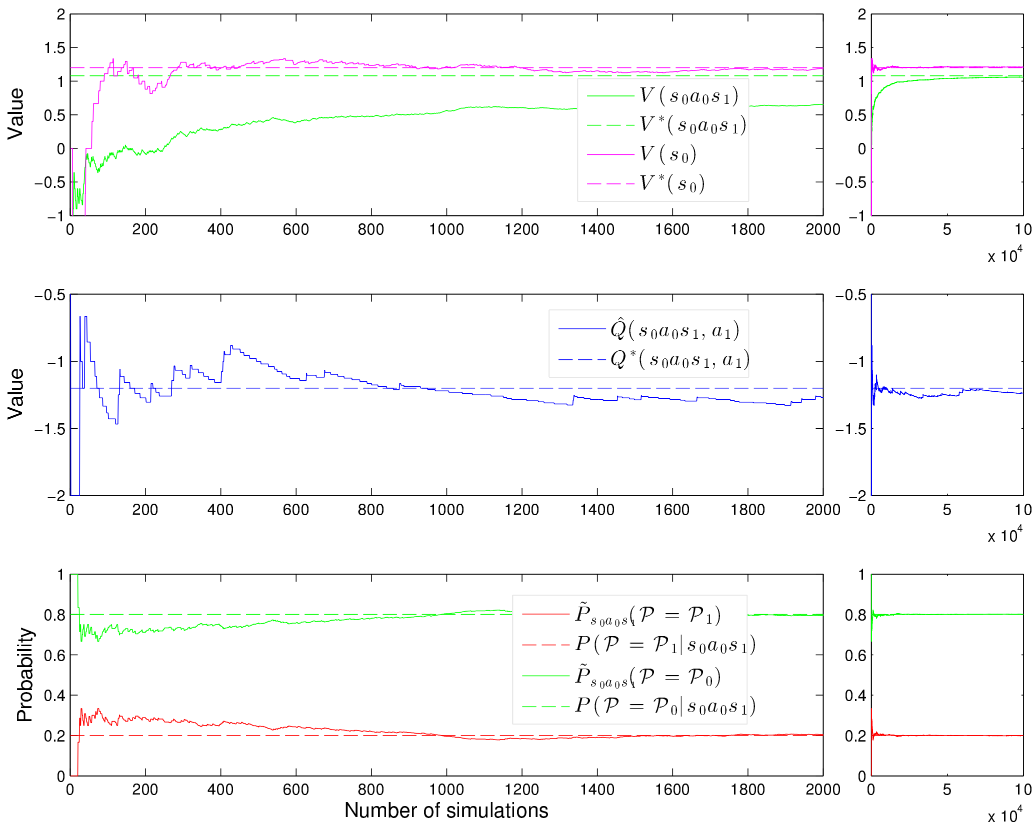


Figure 3: Tracking of different internal variables of BAMCP for the example of Section 3.1.1 with $\gamma = 0.9$. BAMCP is run at the starting state for a number of simulations (x-axis) and with $c = 20$. The first two rows show the evolution of values at tree nodes corresponding to different histories, along with target values as computed in Equation 20. The bottom row shows the evolution of $\tilde{P}_{s_0a_1s_1}(\mathcal{P} = \mathcal{P}_0) = 1 - \tilde{P}_{s_0a_1s_1}(\mathcal{P} = \mathcal{P}_1)$, the empirical distribution of MDPs seen going through tree node $\langle s_0a_1s_1\rangle$ (i.e., $\frac{1}{N(\langle s_0a_1s_1\rangle)} \sum_{i=0}^{N(\langle s_0a_1s_1\rangle)} \mathbf{1}[\mathcal{P}^i = \mathcal{P}_0]$). (Left) The first 2000 simulations (Right) Zoomed out view of 100,000 simulations, displaying empirical convergence to target values.

only the particular transition probabilities that are required as the simulation traverses the tree, and also during the rollout.

Consider $\mathcal{P}(s, a, \cdot)$ to be parametrized by a latent variable $\theta_{s,a}$ for each state and action pair. These may depend on each other, as well as on an additional set of latent variables ϕ. The posterior over $\mathcal{P}$ can be written as $P(\Theta|h) = \int_\phi P(\Theta|\phi, h)P(\phi|h)$, where $\Theta = \{\theta_{s,a}|s \in S, a \in A\}$. Define $\Theta_t = \{\theta_{s_1,a_1}, \cdots, \theta_{s_t,a_t}\}$ as the (random) set of θ parameters required during the course of a BAMCP simulation that starts at time 1 and ends at time t. Using the chain rule, we can rewrite

$$\begin{aligned}
P(\Theta|\phi, h) = & P(\theta_{s_1,a_1}|\phi, h) \\
& P(\theta_{s_2,a_2}|\Theta_1, \phi, h) \\
& \quad \vdots \\
& P(\theta_{s_T,a_T}|\Theta_{T-1}, \phi, h) \\
& P(\Theta \setminus \Theta_T|\Theta_T, \phi, h)
\end{aligned}$$

where T is the length of the simulation and $\Theta \setminus \Theta_T$ denotes the (random) set of parameters that are not required for a simulation. For each simulation i, we sample $P(\phi|h_t)$ at the root and then lazily sample the θ_{s_t,a_t} parameters as required, conditioned on ϕ and all Θ_{t-1} parameters sampled for the current simulation. This process is stopped at the end of the simulation, typically long before all θ parameters have been sampled. For example, if the transition parameters for different states and actions are independent, we can simply draw any necessary parameters individually for each state-action pair encountered during a simulation. In general, transition parameters are not independent for different states, but dependencies are likely to be structured. For example, the MDP dynamics could arise from a mixture model where ϕ denotes the mixture component and $P(\phi|h)$ specifies the posterior mixture proportion. Then, if the transition parameters θ are conditionally independent given the mixture component, sampling ϕ^i at the root for simulation i allows us to sample the required parameters $\theta_{s,a}$ independently from $P(\theta_{s,a}|\phi^i, h)$ just when they are required during the i-th simulation. This leads to substantial performance improvement, especially in large MDPs where a single simulation only requires a small subset of parameters (see for example the domain in Section 4.2 for a concrete illustration). This lazy sampling scheme is not limited to shallow latent variable models; in deeper models, we can also benefit from conditional independencies to save on sampling operations for each simulation by sampling only the necessary latent variables — as opposed to sampling all of ϕ.

3.3 Rollout Policy Learning

The choice of rollout policy π_{ro} is important if simulations are few, especially if the domain does not display substantial locality or if rewards require a carefully selected sequence of actions to be obtained. Otherwise, a simple uniform random policy can be chosen to provide noisy estimates. In this work, we learn Q_{ro}, the optimal Q-value in the real MDP, in a model-free manner, using Q-learning, from samples (s_t, a_t, r_t, s_{t+1}) obtained off-policy as a result of the interaction of the BAMCP agent with the MDP at time t. For each real transition

(s_t, a_t, r_t, s_{t+1}) observed, we update

$$Q_{ro}(s_t, a_t) \leftarrow Q_{ro}(s_t, a_t) + \alpha(r_t + \gamma \max_a Q_{ro}(s_{t+1}, a) - Q_{ro}(s_t, a_t)), \tag{20}$$

where α is some learning rate parameter; this is the standard Q-learning rule (Watkins, 1989). Acting greedily according to Q_{ro} translates to pure exploitation of gathered knowledge. A rollout policy in BAMCP following Q_{ro} could therefore over-exploit. Instead, similar to the work of Gelly and Silver (2007), we select an ϵ-greedy policy with respect to Q_{ro} as our rollout policy π_{ro}. In other words, after t steps in the MDP, we have updated Q_{ro} t times and we use the following stochastic rollout policy for all MCTS simulations at the $t+1$ decision step:

$$\pi_{ro}(s, a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \operatorname{argmax}_{a'} Q_{ro}(s, a') \\ \frac{\epsilon}{|A|} & \text{otherwise,} \end{cases} \tag{21}$$

where $\pi_{ro}(s, a)$ is the probability of selecting action a when in the MDP state s (i.e., history is ignored) during a rollout. This biases rollouts towards observed regions of high rewards. This method provides valuable direction for the rollout policy at negligible computational cost. More complex rollout policies can be considered, for example rollout policies that depend on the sampled model $\mathcal{P}^i$ or on the history h_t. However, these usually incur computational overhead, which may be less desired than running more simulations with worse estimates.

3.4 Theoretical Properties

In this section, we show that BAMCP converges to the Bayes-optimal policy. We first present theoretical results in the case that exact posterior inference can be conducted to obtain posterior samples of the dynamics (Section 3.4.1), we then extend the convergence guarantee to the case where approximate inference (MCMC-based) is necessary to produce posterior samples (Section 3.4.2).

The proof of Theorem 1 was present in the supplementary material by Guez et al. (2012). Theorem 2 is a novel contribution of this paper.

3.4.1 Exact Inference Case

The main step is proving that root sampling does not alter the behavior of BA-UCT. Our proof is an adaptation of the POMCP proof by Silver and Veness (2010). We then provide some intuition and some empirical evidence of convergence on simple Bandit problems — where the Bayes-optimal solution is known.

Consider the BA-UCT algorithm: UCT applied to the Bayes-Adaptive MDP (its dynamics are described in Equation 16). Let $\mathcal{D}^\pi(h_T)$ be the *rollout distribution* of BA-UCT: the probability that history h_T is generated when running the BA-UCT search from $\langle s_t, h_t\rangle$, with h_t a prefix of h_T, $T - t$ the effective horizon in the search tree, and π is an arbitrary EE policy. Similarly define the quantities $\tilde{\mathcal{D}}^\pi(h_T)$: the probability that history h_T is generated when running the BAMCP algorithm, and $\tilde{P}_h(\mathcal{P})$: the distribution of $\mathcal{P}$ at node h when running BAMCP. The following lemma shows that these rollout statistics are the same under BAMCP as BA-UCT.[4]

Lemma 1 $\mathcal{D}^\pi(h_T) = \tilde{\mathcal{D}^\pi}(h_T)$ *for all EE policies* $\pi : \mathcal{H} \to A$.

Proof Let π be arbitrary. We show by induction that for all suffix histories h of h_t, (a) $\mathcal{D}^\pi(h) = \tilde{\mathcal{D}^\pi}(h)$; and (b) $P(\mathcal{P}\,|h) = \tilde{P}_h(\mathcal{P})$, where $P(\mathcal{P}\,|h)$ denotes (as before) the posterior distribution over the dynamics given h.

Base case: At the root ($h = h_t$, suffix history of size 0), it is clear that $\tilde{P}_{h_t}(\mathcal{P}) = P(\mathcal{P}\,|h_t)$ since we are sampling from the posterior at the root node and $\mathcal{D}^\pi(h_t) = \tilde{\mathcal{D}^\pi}(h_t) = 1$ since all simulations go through the root node.

Step case:

Assume proposition true for all suffices of size j. Consider any suffix has' of size $j+1$, where $a \in A$ and $s' \in S$ are arbitrary and h is an arbitrary suffix of size j ending in s. The following relation holds:

$$\begin{aligned}
\mathcal{D}^\pi(has') &= \mathcal{D}^\pi(h)\pi(h,a) \int_{\mathcal{P}} \mathrm{d}\mathcal{P}\, P(\mathcal{P}\,|h)\, \mathcal{P}(s,a,s') && (22)\\
&= \tilde{\mathcal{D}}^\pi(h)\pi(h,a) \int_{\mathcal{P}} \mathrm{d}\mathcal{P}\, \tilde{P}_h(\mathcal{P})\, \mathcal{P}(s,a,s') && (23)\\
&= \tilde{\mathcal{D}^\pi}(has'), && (24)
\end{aligned}$$

where the second line is obtained using the induction hypothesis, and the rest from the definitions. In addition, we can match the distribution of the samples $\mathcal{P}$ at node has':

$$\begin{aligned}
P(\mathcal{P}\,|has') &= P(has'|\,\mathcal{P})P(\mathcal{P})/P(has') && (25)\\
&= P(h|\,\mathcal{P})P(\mathcal{P})\,\mathcal{P}(s,a,s')/P(has') && (26)\\
&= P(\mathcal{P}\,|h)P(h)\,\mathcal{P}(s,a,s')/P(has') && (27)\\
&= Z P(\mathcal{P}\,|h)\,\mathcal{P}(s,a,s') && (28)\\
&= Z\tilde{P}_h(\mathcal{P})\,\mathcal{P}(s,a,s') && (29)\\
&= Z\tilde{P}_{ha}(\mathcal{P})\,\mathcal{P}(s,a,s') && (30)\\
&= \tilde{P}_{has'}(\mathcal{P}), && (31)
\end{aligned}$$

where Equation 29 is obtained from the induction hypothesis, Equation 30 is obtained from the fact that the choice of action at each node is made independently of the samples $\mathcal{P}$. Finally, to obtain Equation 31 from Equation 30, consider the probability that a sample $\mathcal{P}$ arrives at node has', it first needs to traverse node ha (this occurs with probability $\tilde{P}_{ha}(\mathcal{P})$) and then, from node ha, the state s' needs to be sampled (this occurs with probability $\mathcal{P}(s,a,s')$); therefore, $\tilde{P}_{has'}(\mathcal{P}) \propto \tilde{P}_{ha}(\mathcal{P})\,\mathcal{P}(s,a,s')$. Z is the normalization constant: $Z = 1/(\int_{\mathcal{P}} d\mathcal{P}\; \mathcal{P}(s,a,s')P(\mathcal{P}\,|h)) = 1/(\int_{\mathcal{P}} d\mathcal{P}\; \mathcal{P}(s,a,s')\tilde{P}_h(\mathcal{P}))$. This completes the induction. $\square$

4. For ease of notation, we refer to a node with its history only, as opposed to its state and history as in the rest of the paper.

The proof of Lemma 1 does not make explicit the use of lazy sampling, since this method for realizing the values of relevant random variables does not affect the rollout distribution and so does not affect what is being computed, only how.

Define $V(\langle s,h\rangle) = \max_{a\in A} Q(\langle s,h\rangle, a) \quad \forall \langle s,h\rangle \in S \times \mathcal{H}$. We now show that BAMCP converges to the Bayes-optimal solution.

Theorem 1 *For all $\epsilon > 0$ (the numerical precision, see Algorithm 1) and a suitably chosen c (e.g. $c > \frac{R_{\max}}{1-\gamma}$), from state $\langle s_t, h_t\rangle$, BAMCP constructs a value function at the root node that converges in probability to an ϵ'-optimal value function, $V(\langle s_t, h_t\rangle) \xrightarrow{p} V^*_{\epsilon'}(\langle s_t, h_t\rangle)$, where $\epsilon' = \frac{\epsilon}{1-\gamma}$. Moreover, for large enough $N(\langle s_t, h_t\rangle)$, the bias of $V(\langle s_t, h_t\rangle)$ decreases as $O(\log(N(\langle s_t, h_t\rangle))/N(\langle s_t, h_t\rangle))$.*

Proof The UCT analysis by Kocsis and Szepesvári (2006) applies to the BA-UCT algorithm, since it is vanilla UCT applied to the BAMDP (a particular MDP). It also applies for arbitrary rollout policies, including the one developed in Section 3.3. By Lemma 1, BAMCP simulations are equivalent in distribution to BA-UCT simulations. The nodes in BAMCP are therefore evaluated exactly as in BA-UCT, providing the result. □

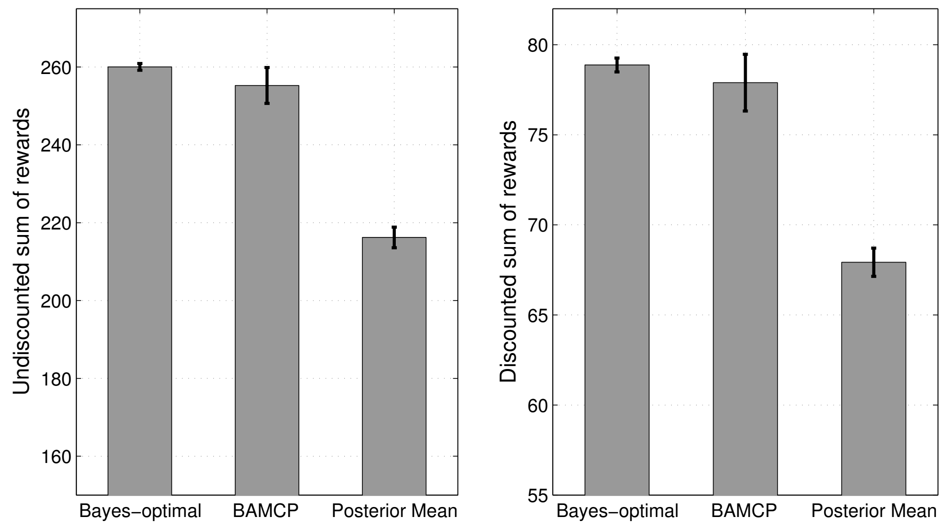
Lemma 1 provides some intuition for why belief updates are unnecessary in the search tree: the search tree filters the samples from the root node so that the distribution of samples at each node is equivalent to the distribution obtained when explicitly updating the belief. In particular, the root sampling in POMCP (Silver & Veness, 2010) and BAMCP is different from evaluating the tree using the posterior mean. This is illustrated empirically in Figures 4 and 5 in the case of simple Bandit problems.

3.4.2 Approximate Inference Case

In Theorem 1, we made the implicit assumption that BAMCP is provided with true samples drawn iid from the posterior. However, most sophisticated priors will require some form of approximate sampling scheme (see, for example, the task in Section 4.2), such as Markov Chain Monte Carlo (MCMC), which generally deliver correlated posterior samples after the chain converges to the stationary distribution (Neal, 1993). Thus, it is necessary to extend the proof of convergence of BAMCP to deal with samples of this nature.

Theorem 2 *When using an approximate sampling procedure based on a MCMC chain with stationary distribution $P(\mathcal{P}\,|h_t)$ (e.g., Metropolis-Hastings or Gibbs sampling) to produce a sample sequence $\mathcal{P}^1, \mathcal{P}^2, \ldots$ at the root node of BAMCP, the value $V(\langle s_t, h_t\rangle)$ found by BAMCP converges in probability to a (near-)optimal value function.*

Proof Let $\epsilon > 0$ be the chosen numerical accuracy of the algorithm. We can choose a finite depth T for the search tree as a function of ϵ, $r_{\max}$, and γ that guarantees the sum total return after depth T amounts to less than ϵ. Now consider any leaf Q-node i of that tree, with mean value $\mu_{in} = \frac{1}{n}\sum_{m=1}^{n} r_m$ after n simulations, where r_m is the reward obtained from this node at the m-th simulation going through that node. Since UCB1 is used throughout the tree, exploration never ceases and this guarantees that $n \to \infty$ (see for example Kocsis & Szepesvári, 2006, Thm. 3).



(a) (b)

Figure 4: Performance comparison of BAMCP (50000 simulations, 100 runs) against the posterior mean decision on an 8-armed Bernoulli bandit with $\gamma = 0.99$ after 300 steps. The arms' success probabilities are all 0.6 except for one arm which has success probability 0.9. The Bayes-optimal result is obtained from 1000 runs with the Gittins indices (Gittins et al., 1989). **a.** Mean sum of rewards after 300 steps. **b.** Mean sum of discounted rewards after 300 steps.

Root sampling filtering (Lemma 1) still holds despite the approximate sampling at the root node; since it is a statement about the distribution of samples, not about the order in which these samples arrive. Therefore, the distribution of dynamics at node i converges to the right stationary distribution $P(\mathcal{P}\,|h_i)$, where h_i is the history corresponding to node i. Asymptotic results on Markov Chains (Law of large numbers for Markov Chains) guarantee us that $\mu_{in} \to \mu_i$ a.s., where μ_i is the true expected reward at leaf node i.

Given convergence at the leaves, we can work our way up the tree by backward induction to show that the values at each node converge to their (near-)optimal values. In particular the value at the root converges to an $\epsilon-$optimal value. $\square$

3.5 Possible Misuse of Latent Variable Information: a Counter-Example

When planning in a BAMDP using a sample-based forward-search algorithm such as BAMCP, it could be tempting to use the knowledge available in the sampler when producing samples (such as the value of latent variables in the model) to take better planning decisions. For example, when generating a sample $\mathcal{P}^i$ of the dynamics according to a posterior distribution $P(\mathcal{P}\,|h)$ which can be written as $\int_\theta P(\mathcal{P}\,|\theta)P(\theta|h)$, $\mathcal{P}^i$ might have been generated by sampling θ^i from $P(\theta|h)$ before sampling $\mathcal{P}^i$ from $P(\mathcal{P}\,|\theta^i)$. Since the value of θ is available and contain high-level information, one natural question is to ask whether the search can be informed by the value of θ.

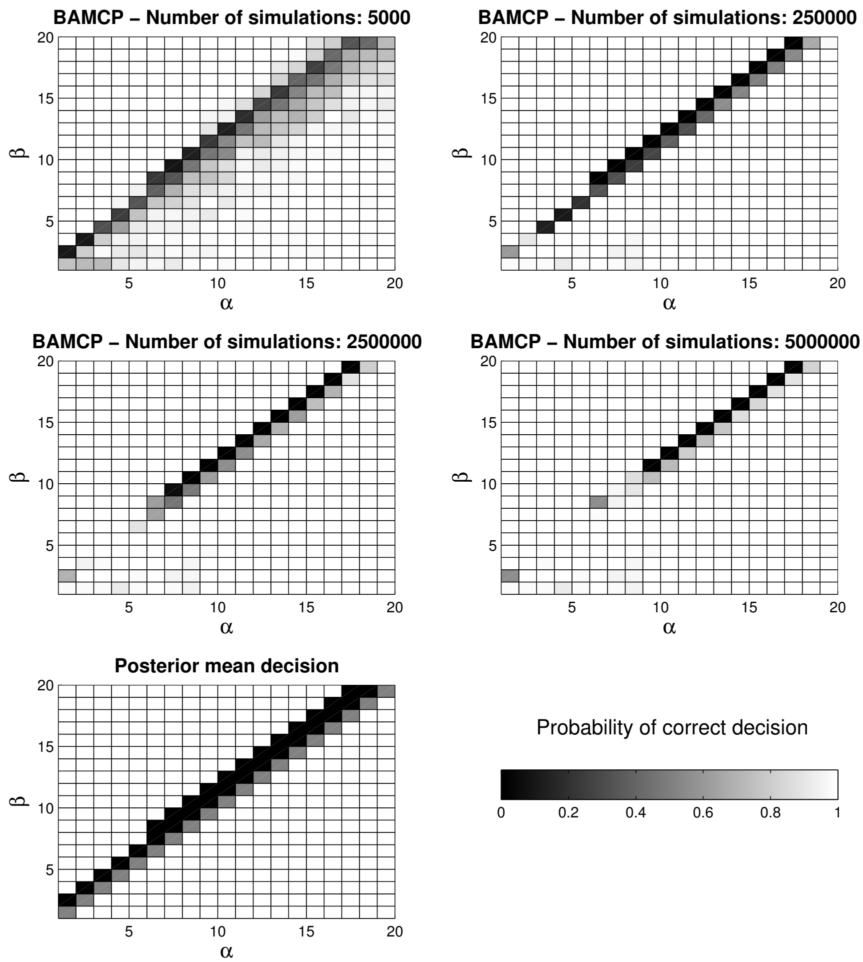
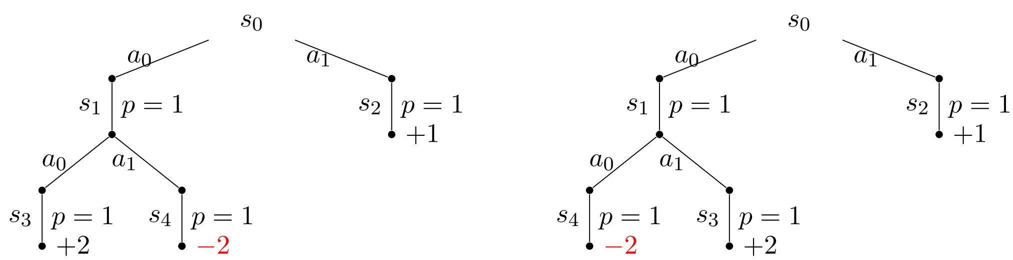


Figure 5: Evaluation of BAMCP against the Bayes-optimal policy, for the case $\gamma = 0.95$, when choosing between a deterministic arm with reward 0.5 and a stochastic arm with reward 1 with posterior probability $p \sim \text{Beta}(\alpha, \beta)$. The result is tabulated for a range of values of α, β, each cell value corresponds to the probability of making the correct decision (computed over 50 runs) when compared to the Gittins indices (Gittins et al., 1989) for the corresponding posterior. The first four tables corresponds to different number of simulations for BAMCP and the last table shows the performance when acting according to the posterior mean. In this range of α, β values, the Gittins indices for the stochastic arm are larger than 0.5 (i.e., selecting the stochastic arm is optimal) for $\beta \leq \alpha + 1$ but also $\beta = \alpha + 2$ for $\alpha \geq 6$. Acting according to the posterior mean is different from the Bayes-optimal decision when $\beta >= \alpha$ and the Gittins index is larger than 0.5. BAMCP is guaranteed to converge to the Bayes-optimal decision in all cases, but convergence is slow for the edge cases where the Gittins index is close to 0.5 (e.g., For $\alpha = 17, \beta = 19$, the Gittins index is 0.5044 which implies a value of at most $0.5044/(1-\gamma) = 10.088$ for the stochastic arm versus a value between 10 and $0.5 + \gamma \times 10.088 = 10.0836$ for the deterministic arm).

Here, we outline one incorrect way of using the latent variable value during search. Suppose we would want to split our search tree on the value of θ (this would occur implicitly if we were constructing history features based on the value of θ), we provide below a simple counter-example that shows that this is not a valid search approach.

Consider a simple prior distribution on two 5-state MDPs, illustrated in Figure 6, where $P(\theta = 0|h_0) = P(\theta = 1|h_0) = \frac{1}{2}$, and $P(\mathcal{P}\,|\theta)$ is a delta function on the illustrated MDP.



(a) $\theta = 0$ (b) $\theta = 1$

Figure 6: The two possible MDPs corresponding to the two settings of θ.

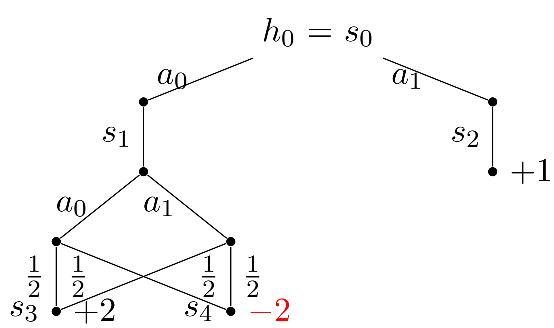


Figure 7: BAMDP, nodes correspond to belief(or history)-states.

There are 2 deterministic actions (a_0, a_1) in each MDP, the episode length is 1 or 2 steps. The only difference between the two MDPs is the outcome of taking action a_0 and a_1 in state s_1, as illustrated in Figure 6, so that a_0 is rewarding when $\theta = 0$ and costly when $\theta = 1$, and vice-versa for a_1. All the rewards are obtained from executing any action at any of the terminal states (s_2, s_3, s_4).

Observing the first transition is not informative, which implies that the posterior distribution is unchanged after the first transition: $P(\mathcal{P}\,|h_0) = P(\mathcal{P}\,|h_0 a_0 s_1) = P(\mathcal{P}\,|h_0 a_1 s_2)$. The BAMDP corresponding to this problem is illustrated in Figure 7.

At history-state $h_0 = s_0$, the Bayes-optimal Q values can easily be computed:

$$
\begin{aligned}
Q^*(h_0, a_1) &= \gamma, && (32)\\
Q^*(h_0a_0s_1, a_0) &= 0 + \gamma\,(2 \cdot P(s_3|h_0a_0s_1a_0) - 2 \cdot P(s_4|h_0a_0s_1a_0)) && (33)\\
&= \gamma(1-1) = 0, && (34)\\
Q^*(h_0a_0s_1, a_1) &= 0 + \gamma\,(2 \cdot P(s_3|h_0a_0s_1a_0) - 2 \cdot P(s_4|h_0a_0s_1a_0)) && (35)\\
&= \gamma(1-1) = 0, && (36)\\
Q^*(h_0, a_0) &= 0 + \gamma \max_a Q^*(h_0a_0s_1, a) = 0, && (37)
\end{aligned}
$$

which implies that $a_1 = \pi^*(h_0)$ for any γ. [We used the fact that $P(s_3|h_0a_0s_1a_0) = P(\theta = 0|h_0a_0s_1a_0) \cdot P(s_3|\theta = 0, s_1a_0) + P(\theta = 1|h_0a_0s_1a_0) \cdot P(s_3|\theta = 1, s_1a_0) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$, and similarly for $P(s_4|h_0a_0s_1a_0)$].

Note that, since belief updates only occur at the terminal states, forward-search with or without root sampling will be equivalent. They both would construct a search tree as in Figure 7 and compute the right value and right decision.

The problem comes in if we decide to split our search tree at chance nodes based on the value of θ in the generated samples going down the tree. For example, after taking action a_0 in state s_0, we would be using either an MDP for which $\theta = 0$ w.p 0.5 or an MDP for which $\theta = 1$ w.p 0.5. Since multiple values of θ go through the node h_0a_0, we would branch the tree as illustrated in Figure 8. This search tree is problematic because the value computed for $Q^*(h_0, a_0)$ becomes $2 \cdot \gamma^2$, which is larger than $Q^*(h_0, a_1) = \gamma$ for any $\gamma > 0.5$. Therefore, the policy that is computed at the root is no longer Bayes-optimal.

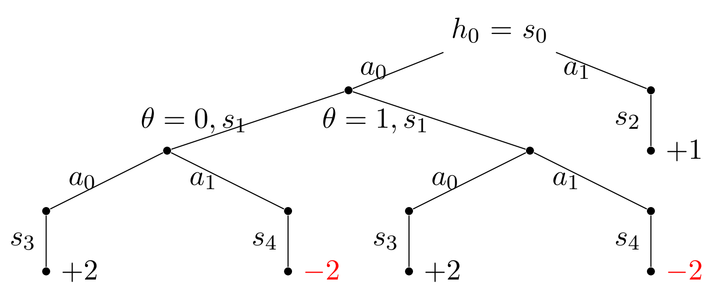


Figure 8: A problematic search tree.

By branching on the latent variable value, we are creating spurious observations: we are implying that the latent variable from the past will be observed in the future, which is not the case.

To summarize, the Bayes-adaptive policy to be optimized must be a function of future histories (i.e., things we'll actually observe in the future), and cannot be a function of future unobserved latent variables. Ignoring this causes problems in simple domains such as the one illustrated above, but similar scenarios would occur in more complex latent variable models for the same reasons.

4. Experiments

We first present empirical results of BAMCP on a set of standard problems with comparisons to other popular algorithms. We then showcase BAMCP's advantages in a large scale task: an infinite 2D grid with complex correlations between reward locations.

4.1 Standard Domains

The following algorithms were run on the standard domains: BAMCP, SBOSS, BEB, BFS3. Details about their implementation and parametrization can be found in Appendix B. In addition, we report results from the work of Strens (2000) for several other algorithms.

For all the following domains, we fix $\gamma = 0.95$.

- The **Double-loop** domain is a 9-state deterministic MDP with 2 actions (Dearden, Friedman, & Russell, 1998), 1000 steps are executed in this domain. It is illustrated in Figure 9(a).
- **Grid5** is a 5×5 grid with a reset state in one corner, and a single reward state diametrically opposite to the reset state. Actions with cardinal directions are executed with small probability of failure ($p_{\text{failure}} = 0.2$) for 1000 steps.
- **Grid10** is a 10×10 grid designed in the same way as Grid5. We collect 2000 steps in this domain.
- **Dearden's Maze** is a 264-states maze with 3 flags to collect (Dearden et al., 1998). A special state provides reward equivalent to the number of flags collected since the last visit. 20000 steps are executed in this domain[5]. It is illustrated in Figure 9(b).

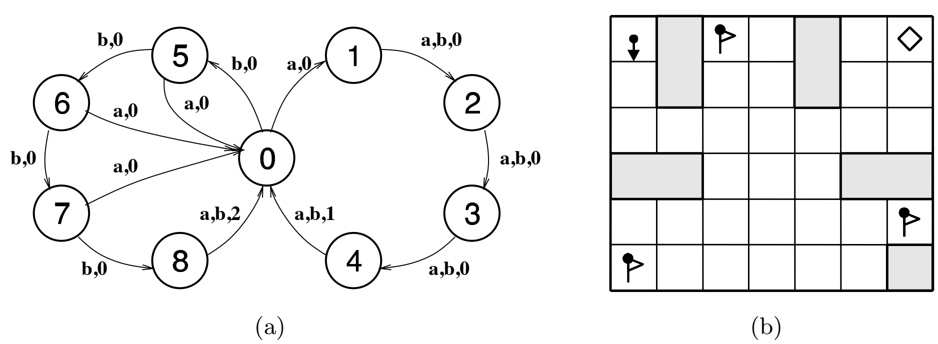


Figure 9: Two of the standard domains described in Section 4.1: a) The Double-loop domain, b) Dearden's maze. Figures from the work of Strens (2000).

To quantify the performance of each algorithm, we measured the total undiscounted reward over many steps. We chose this measure of performance to enable fair comparisons to be drawn with prior work. In fact, we are optimising a different criterion – the discounted

5. The result reported for Dearden's maze with the Bayesian DP alg. by Strens (2000) is for a different version of the task in which the maze layout is given to the agent.

reward from the start state – and so we might expect this evaluation to be unfavourable to our algorithm.

Although one major advantage of Bayesian RL is that one can specify priors about the dynamics, for these domains, we used rather generic priors to enable comparisons with previous work. For the Double-loop domain, the Bayesian RL algorithms were run with a simple Dirichlet-Multinomial model with symmetric Dirichlet parameter $\alpha = \frac{1}{|S|}$. For the grids and the maze domain, the algorithms were run with a sparse Dirichlet-Multinomial model, as described by Friedman and Singer (1999). For both these models, efficient collapsed sampling schemes are available; they are employed for the BA-UCT and BFS3 algorithms in our experiments to compress the posterior parameter sampling and the transition sampling into a single transition sampling step. This considerably reduces the cost of belief updates inside the search tree when using these simple probabilistic models. Unfortunately, efficient collapsed sampling schemes are not available in general (see for example the model in Section 4.2).

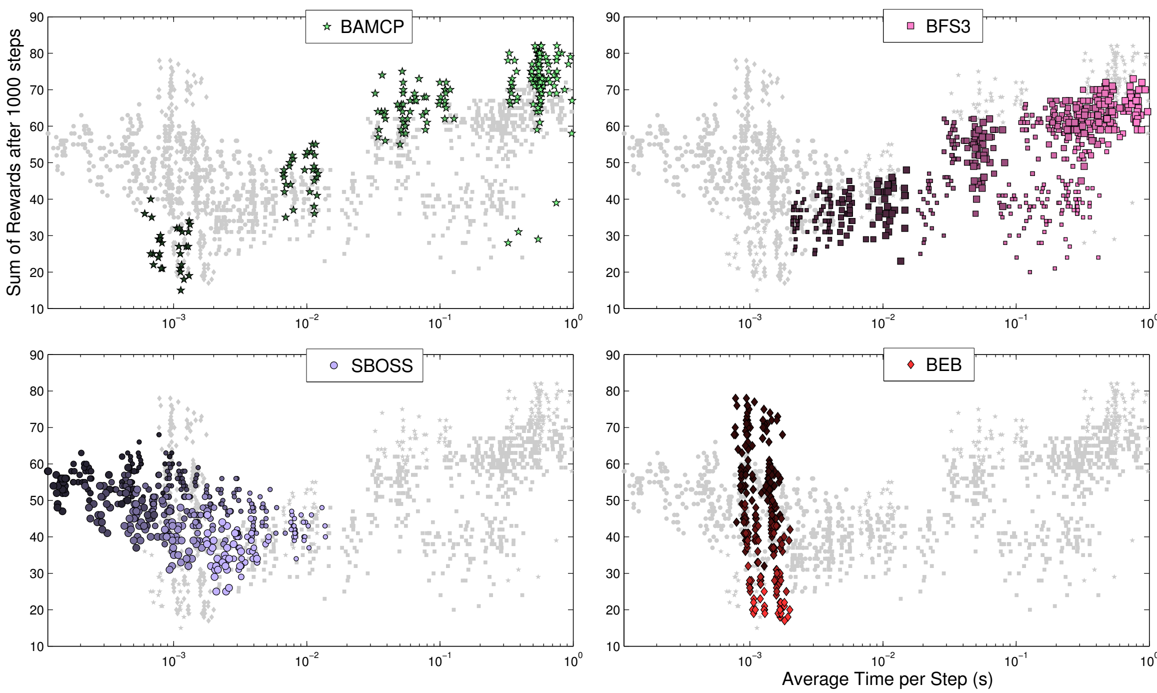


Figure 10: Performance of each algorithm on the Grid5 domain as a function of planning time. Each point corresponds to a single run of an algorithm with an associated setting of the parameters. Increasing brightness inside the points codes for an increasing value of a parameter (BAMCP and BFS3: number of simulations, BEB: bonus parameter β, SBOSS: number of samples K). A second dimension of variation is coded as the size of the points (BFS3: branching factor C, SBOSS: resampling parameter δ). The range of parameters is specified in Appendix B.

A summary of the results is presented in Table 1. Figures 10 and 11 report the planning time/performance trade-off for the different algorithms on the Grid5 and Maze domain.

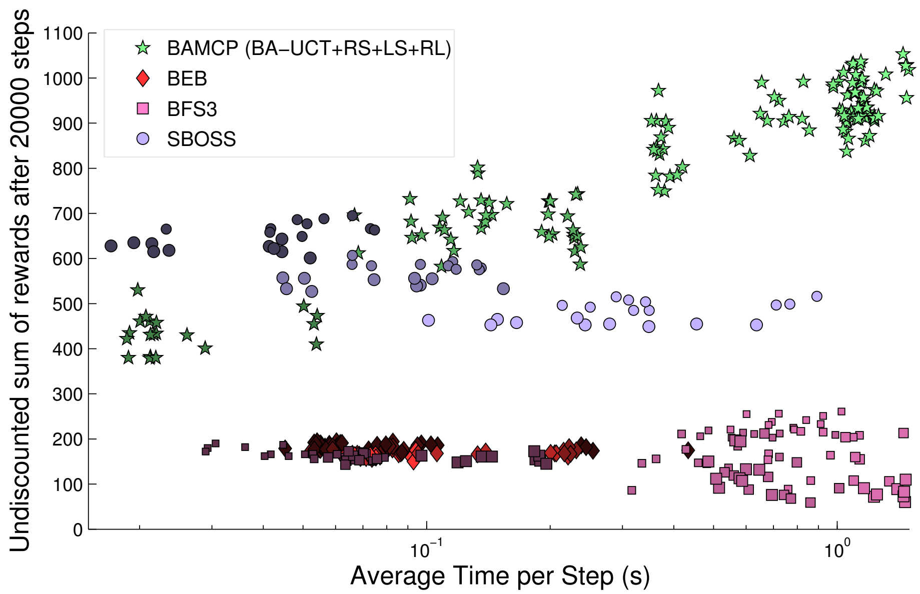


Figure 11: Performance of each algorithm, as in Figure 10 but on Dearden's Maze domain.

	Double-loop	Grid5	Grid10	Dearden's Maze
BAMCP	**387.6 ± 1.5**	**72.9 ± 3**	**32.7 ± 3**	**965.2 ± 73**
BFS3 (Asmuth & Littman, 2011)	382.2 ± 1.5	66 ± 5	10.4 ± 2	240.9 ± 46
SBOSS (Castro & Precup, 2010)	371.5 ± 3	59.3 ± 4	21.8 ± 2	671.3 ± 126
BEB (Kolter & Ng, 2009)	386 ± 0	67.5 ± 3	10 ± 1	184.6 ± 35
Bayesian DP* (Strens, 2000)	377 ± 1	-	-	-
Bayes VPI+MIX* (Dearden et al., 1998)	326 ± 31	-	-	817.6 ± 29
IEQL+* (Meuleau & Bourgine, 1999)	264 ± 1	-	-	269.4 ± 1
QL Boltzmann*	186 ± 1	-	-	195.2 ± 20

Table 1: Experiment results summary. For each algorithm, we report the mean sum of rewards and confidence interval for the best performing parameter within a reasonable planning time limit (0.25 s/step for Double-loop, 1 s/step for Grid5 and Grid10, 1.5 s/step for the Maze). For BAMCP, this simply corresponds to the number of simulations that achieve a planning time just under the imposed limit. * Results by Strens (2000) reported without timing information.

On all the domains tested, BAMCP performed best. Other algorithms came close on some tasks, but only when their parameters were tuned to that specific domain. This is particularly evident for BEB, which required a different value of exploration bonus to achieve maximum performance in each domain. BAMCP's performance is stable with respect to the choice of its exploration constant ($c = 3$) and it did not require fine tuning to obtain the results.

BAMCP's performance scaled well as a function of planning time, as is evident in Figures 10 and 11. In contrast, SBOSS follows the opposite trend. If more samples are employed to build the merged model, SBOSS actually becomes too optimistic and over-explores, de-

(a)

(b)

(c)

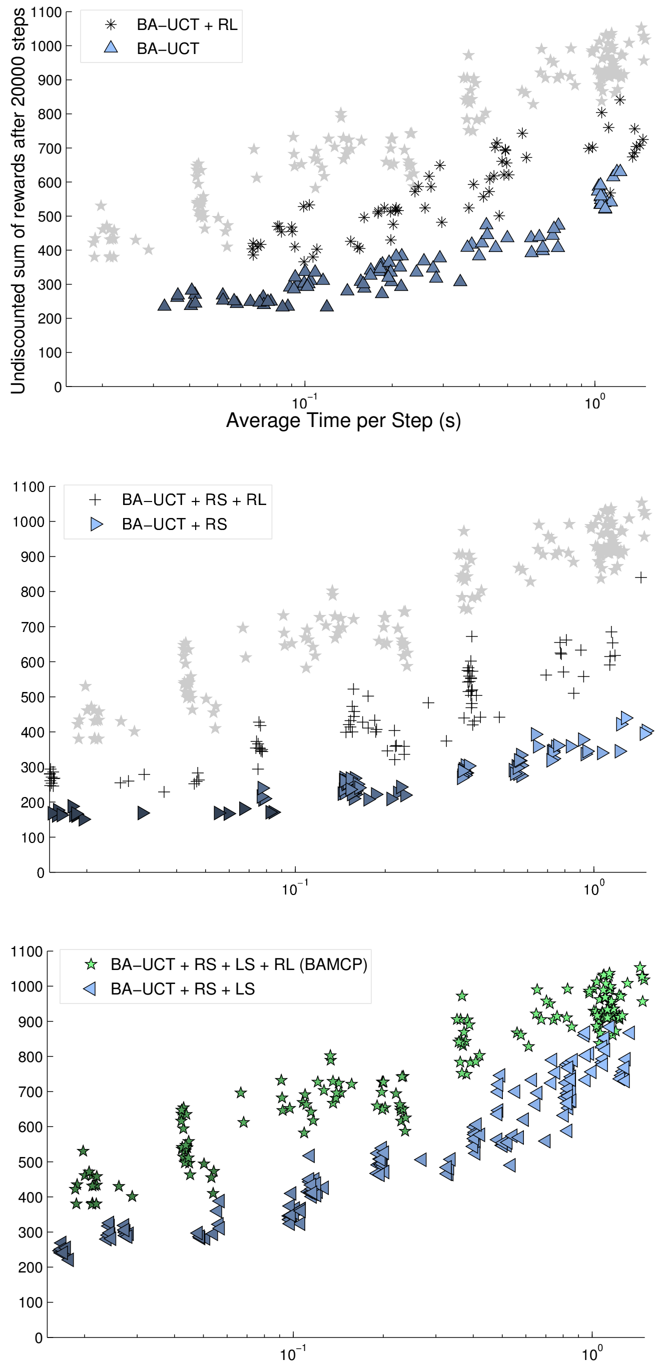


Figure 12: Evolution of performance from BA-UCT to BAMCP on Dearden's Maze domain. BAMCP is present on all plots for comparison, as also displayed in Figure 11. **a.** Performance of vanilla BA-UCT with and without rollout policy learning (RL) presented in Section 3.3. **b.** Performance of BA-UCT with Root Sampling (RS), as presented in Section 3.1, and with and without rollout learning. **c.** Performance of BA-UCT with Root Sampling and Lazy Sampling (LS), as presented in Section 3.2. In addition with rollout policy learning, this is the BAMCP algorithm.

grading its performance. BEB cannot take advantage of prolonged planning time at all. The performance of BFS3 generally improves with more planning time, given an appropriate choice of parameters, but it is not obvious how to trade-off the branching factor, depth, and number of simulations in each domain. BAMCP greatly benefited from our lazy sampling scheme in the experiments, providing a 35× speed improvement over the naive approach in the maze domain for example; this is illustrated in Figure 12.

Dearden's maze aptly illustrates a major drawback of forward search sparse sampling algorithms such as BFS3. Like many maze problems, all rewards are zero for at least k steps, where k is the solution length. Without prior knowledge of the optimal solution length, all upper bounds will be higher than the true optimal value until the tree has been fully expanded up to depth k – even if a simulation happens to solve the maze. In contrast, once BAMCP discovers a successful simulation, its Monte-Carlo evaluation will immediately bias the search tree towards the successful trajectory.

Figure 12 confirms that, even on a moderate-sized domain with a simple prior (Independent Sparse Dirichlet-Multinomial), BAMCP amply benefits from root sampling, lazy sampling, and rollout learning. For more complex priors, as in the following section, BA-UCT becomes computationally intractable. Root sampling and lazy sampling are then mandatory components.

4.2 Infinite 2D Grid Task

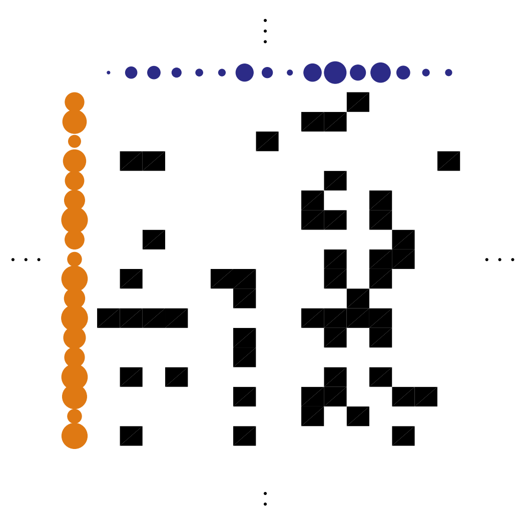


Figure 13: A portion of an infinite 2D grid task generated with Beta distribution parameters $\alpha_1 = 1, \beta_1 = 2$ (columns) and $\alpha_2 = 2, \beta_2 = 1$ (rows). Black squares at location (i,j) indicates a reward of 1, the circles represent the corresponding parameters p_i (blue) and q_j (orange) for each row and column (area of the circle is proportional to the parameter value). One way to interpret these parameters is that following column i implies a collection of $2p_i/3$ reward on average (2/3 is the mean of a Beta(2, 1) distribution) whereas following any row j implies a collection of $q_j/3$ reward on average; but high values of parameters p_i are less likely than high values parameters q_j. These parameters are employed for the results presented in Figure 14-c).

It is perhaps not unfair to characterize all the domains in the previous section as being of very limited scale. Indeed, this can be seen as a correct reflection of the state of the art of Bayesian RL. However, BAMCP, because of its root-based lazy sampling, can be applied to considerably larger and more challenging domains. We therefore designed a new problem that is well beyond the capabilities of prior algorithms since it has an infinite and combinatorially structured state space, and an even more challenging belief space. Although still abstract, this new task illustrates something of BAMCP's power.

4.2.1 Problem Description

The new problem is a class of complex MDPs over an infinite grid. In a draw of a particular MDP, each column i has an associated latent parameter $p_i \sim \text{Beta}(\alpha_1, \beta_1)$ and each row j has an associated latent parameter $q_j \sim \text{Beta}(\alpha_2, \beta_2)$. The probability of grid cell ij having a reward of 1 is $p_i q_j$, otherwise the reward is 0. The agent knows it is on a grid and is always free to move in any of the four cardinal directions. Rewards are consumed when visited; returning to the same location subsequently results in a reward of 0. As opposed to the independent Dirichlet priors employed in standard domains, here, dynamics are tightly correlated across states (i.e., observing a state transition provides information about other state transitions).

The domain is illustrated in Figure 13. Although the uncertainty appears to concern the reward function of the MDP rather than the dynamics, it can be viewed formally as uncertainty in the dynamics when the state is augmented with a binary variable that indicates whether a reward is present.[6]

Formally, since rewards disappear after one visit, the description of the state in the MDP needs to include information about the state of all the rewards (for example in the form of a set of grid locations previously visited) in addition to the position of the agent on the infinite grid. A state s is therefore the combination of the current agent's location (i, j), the unordered set of previously visited locations V, and the binary variable $R = r_{ij}$. The dynamics $\mathcal{P}$ then deterministically updates the position of the agent and the visited locations based on the agent's action, and updates R according to the reward map. The known reward function is then simply $\mathcal{R}(s, a) = s(R)$ for all a (i.e., as described before, the agent gets a reward in position ij if $r_{ij} = 1$).

4.2.2 Inference

Posterior inference (of the dynamics $\mathcal{P}$) in this model requires approximation because of the non-conjugate coupling of the variables. To see this, consider the posterior probability of a particular grid cell kl having a reward of 1 (denote this event $r_{kl} = 1$), then

$$P(r_{kl} = 1|O) = \int_{p_k, q_l} p_k q_l \; P(p_k, q_l|O) \; \mathrm{d}p_k \mathrm{d}q_l, \tag{38}$$

where $O = \{(i, j)\}$ is the set of observed reward locations, each associated with an observed reward $r_{ij} \in \{0, 1\}$. Sampling r_{kl} is straightforward given access to posterior samples of

6. In fact, the BAMDP framework can be straightforwardly extended to deal with more general, partially-observed, reward functions (Duff, 2002).

p_k and q_l. However, the posterior distribution on p_k and q_l, $P(p_k, q_l|O)$, cannot be easily sampled from, it is given by:

$$P(p_k, q_l|O) \propto P(O|p_k, q_l)P(p_k)P(q_l) \tag{39}$$

$$= \int_{P_O \backslash p_k, Q_O \backslash q_l} P(O|P_O, Q_O) \prod_{p \in P_O} P(p) \prod_{q \in Q_O} P(q) \tag{40}$$

$$= \int_{P_O \backslash p_k, Q_O \backslash q_l} \prod_{(i,j) \in O} (p_i q_j)^{r_{ij}} (1 - p_i q_j)^{1 - r_{ij}} \prod_{p \in P_O} \mathrm{Beta}(p; \alpha_1, \beta_1) \prod_{q \in Q_O} \mathrm{Beta}(q; \alpha_2, \beta_2), \tag{41}$$

where P_O denotes the set of parameters p_i for all observed columns i (columns where at least one observation exists) and similarly for Q_O with rows. This posterior suffers from non-conjugacy (because of the multiplicative interaction between the two Beta distribution) but also from a complicated dependence structure (p_k and q_l depend on observations outside of column k and row l). For these reasons, the inference is done approximately via Metropolis-Hastings (details in Appendix C).

4.2.3 Results

Planning algorithms that attempt to solve an MDP based on sample(s) (or the mean) of the posterior (e.g., BOSS, BEB, Bayesian DP) cannot directly handle this large combinatorial state space. Previous forward-search methods (e.g., BA-UCT, BFS3) can deal with the state space, but not the complex belief space: at every node of the search tree they must solve an approximate inference problem to estimate the posterior beliefs. By contrast, BAMCP limits the posterior inference to the root of the search tree and is not directly affected by the size of the state space or belief space, which allows the algorithm to perform well even with a limited planning time. Note that lazy sampling is required in this setup since a full sample of the dynamics involves infinitely many parameters.

Figure 14 demonstrates the planning performance of BAMCP in this complex domain. Performance improves with additional planning time. The quality of the prior clearly affects the agent's performance, BAMCP can take advantage of correct prior information to gain more rewards. In addition, the behavior of the agent is qualitatively different depending on the prior parameters employed.

For example, for the case of Figure 14-a, rewards are often found in relatively dense blocks on the map and the agent exploits this fact when exploring; this explains the high frequency of short dwell times. For Figure 14-b, good rewards rates can be obtained by following the rare rows that have high q_j parameters, but finding good rows can be expensive for at least two reasons: 1) good rows can be far from the agent's current position and 2) it takes longer to decide the value of a row if most observations lack rewards; this is because the entropy of the posterior is larger given observations of no rewards (which can be explained by either rows or columns being poor, or both at the same time) than given observations of rewards (which can be explained with high probability by both rows and columns being good, since $r_{ij} \sim \mathrm{Bernoulli}(p_i q_j)$). Hence, the agent might settle on sub-optimal rows for large periods of time, for example until it gains enough confidence that a better row is likely to be found nearby (as in Bandit problems where the Bayes-optimal agent might settle on

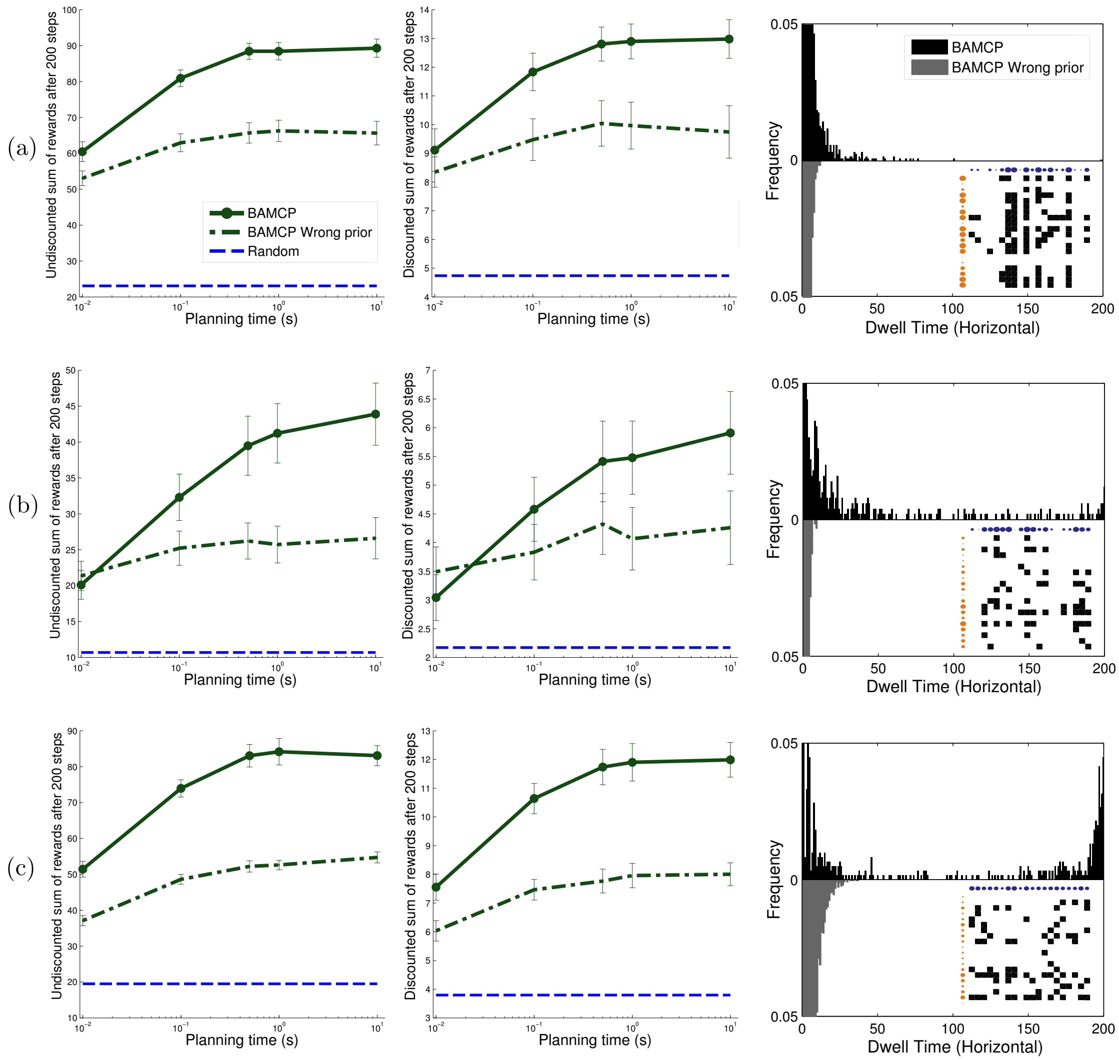


Figure 14: Performance of BAMCP as a function of planning time on the Infinite 2D grid task, for $\gamma = 0.97$, where each row corresponds to a different set of parameters generating the grid. The performance during the first 200 steps in the environment is averaged over 50 sampled environments (5 runs for each sample) and is reported both in terms of undiscounted (left) and discounted (center) sum of rewards. BAMCP is run either with the correct generative model as prior (solid green) or with an incorrect prior (dotted green). The performance of a uniform random policy is also reported (blue). A small sample portion of a grid generated with these parameters is displayed on each row, presented as in Figure 13. The frequency histogram of dwell times — the number of consecutive steps the agent stays on a row before switching — is reported for each scenario. The grids are generated with Beta parameters **a)** α_1=0.5, β_1=0.5, α_2=0.5, β_2=0.5, **b)** α_1=0.5, β_1=0.5, α_2=1, β_2=3, and **c)** α_1=2, β_1=1, α_2=1, β_2=2. For the case of wrong priors (dot-dashed lines), BAMCP is given the parameters **a)** α_1=4, β_1=1, α_2=0.5, β_2=0.5, **b)** α_1=1, β_1=3, α_2=0.5, β_2=0.5, and **c)** α_1=1, β_1=2, α_2=2, β_2=1.

a sub-optimal arm if it believes it likely is the best arm given past data). The heavier-tail distribution of dwell times for this scenario, in Figure 14-b, reflects this behavior.

The case of Figure 14-c consists of a mixture of rich and poor rows. The agent can determine moderately quickly if a row is not good enough, given what it expects to find, and then switches to a nearby row. Once a good enough row is found, the agent can stick to it for large periods of time. This is reflected in the bimodal nature of the distribution of dwell times in Figure 14-c. In many cases, the agent is satisfied with one of the first rows he visits, since it is likely that the agent starts on a good row. He then decides to stay on it for the entire duration of the episode, which explains the peak towards 200.

When BAMCP's prior belief about the dynamics is not the same as the generative model's distribution (*Wrong prior* dot-dashed lines in Figure 14), then maladaptive behavior can be observed. For instance, in Figure 14-a, the deluded agent expects most columns to be rich, and some rows to be rich and others to be poor. Hence, a good strategy given this prior belief is to find one of the good rows and exploit it by travelling horizontally. However, since a lot of columns are actually poor in this generative model, the agent never encounters the continuous sequence of rewards it expects to find on good rows. Given its wrong prior, even if on what is actually a good row, it explains the observation by the row being poor — rather than the column — and switches to a different row. This behavior is reflected in the shorter horizontal dwell times plotted in Figure 14-a. Similar effects can be observed in the *Wrong prior* cases of Figure 14-b,c.

It should be pointed out that the actual Bayes-optimal strategy in this domain is not known — the behavior of BAMCP for finite planning time might not qualitatively match the Bayes-optimal strategy. Nevertheless, we speculate that some of the behavior we observe with BAMCP, including the apparently maladaptive behaviors, would also be found in the Bayes-optimal solution.

5. Discussion

Bayesian model-based reinforcement learning addresses the problem of optimizing the discounted return of an agent when the dynamics are uncertain. By solving an augmented MDP called the BAMDP, the agent can optimally trade-off exploration and exploitation and maximize its expected discounted return according to its prior beliefs. While formally attractive, this framework suffers from a major drawback: it is computationally intractable to solve the BAMDP exactly. While we are not aware of any formal complexity analysis for solving BAMDPs, BAMDPs can be mapped to continuous-state POMDPs with discrete observations (Duff, 2002). In general, solving discrete POMDPs is known to be challenging (Mundhenk, Goldsmith, Lusena, & Allender, 2000; Madani, Hanks, & Condon, 2003).

To approximate the Bayes-optimal solution efficiently, we suggested a sample-based algorithm for Bayesian RL called BAMCP that significantly surpassed the performance of existing algorithms on several standard tasks. We showed that BAMCP can tackle larger and more complex tasks generated from a structured prior, where existing approaches scale poorly. In addition, BAMCP provably converges to the Bayes-optimal solution, even when MCMC-based posterior sampling is employed.

The main idea is to employ Monte-Carlo tree search to explore the augmented Bayes-adaptive search space efficiently. The naive implementation of that idea is an algorithm that we called BA-UCT. However, BA-UCT cannot cope with most priors because it employs expensive belief updates inside the search tree. We therefore introduced three modifications to obtain a computationally tractable sampled-based algorithm: root sampling, which only requires beliefs to be sampled at the start of each simulation (as in Silver & Veness, 2010); a model-free RL algorithm that learns a rollout policy; and a lazy sampling scheme which enables the posterior beliefs to be sampled cheaply.

5.1 Future Work: Algorithms

Despite its excellent empirical performance in many domains (Gelly et al., 2012), the UCT algorithm is known to suffer several drawbacks. First, there is no finite-time regret bound. It is possible to construct malicious environments, for example in which the optimal policy is hidden in a generally low reward region of the tree, where UCT can be misled for long periods (Coquelin & Munos, 2007). Of course, in our setting, appropriate prior distributions might help structure search more effectively. But, the issue of convergence of the MCMC chain in approximate inference settings may hinder any effort to get finite-time guarantees. Second, the UCT algorithm treats every action node as a multi-armed bandit problem. However, there is no actual benefit to accruing reward during planning, and so it is in theory more appropriate to use *pure exploration* bandits (Bubeck, Munos, & Stoltz, 2009).

We focused on learning the dynamics (and implicitly the rewards in the infinite grid task) of a fully observable MDP. If the states are not observed directly, then BAMCP could be extended to maintain beliefs over both the dynamics and the state. Both state and dynamics would then be sampled from the posterior distribution, at the start of each simulation. This setting is known as a Bayes-Adaptive Partially Observable MDP (BAPOMDP) (Ross, Pineau, Chaib-draa, & Kreitmann, 2011).

In this work, we limited ourselves to the case of discrete-state MDPs, since they already present significant challenges for Bayesian exploration. BAMCP cannot be straightforwardly converted to deal with continuous-state MDPs, but it remains to see whether the ingredients that make BAMCP successful in the discrete setting could be reassembled into a continuous-state solution, for example using some form of value function approximation during simulation-based search (Silver, Sutton, & Müller, 2008).

5.2 Future Work: Priors

BAMCP is able to exploit prior knowledge about the dynamics in a principled manner. It is possible to encode many aspects of domain knowledge into the prior distribution, and so an important avenue for future work is to explore rich, structured priors about the dynamics of the MDP. As we showed, if this prior knowledge matches the class of environments that the agent will encounter, then exploration can be significantly accelerated. It is therefore important to understand how to select or learn appropriate priors so that large real-world tasks can be tackled with Bayesian RL. One promising category of rich priors in this context are non-parametric priors. For example, Doshi-Velez, Wingate, Roy, and Tenenbaum (2010) and Wingate, Goodman, Roy, Kaelbling, and Tenenbaum (2011) have investigated this direction, but as yet only in combination with myopic planning algorithms, rather than Bayes-Adaptive planning.

5.3 Evaluation of Bayesian RL Algorithms

Bayesian RL algorithms have traditionally been tested on small, hand-crafted, domains. Even though these domains can contain substantial structure, the priors given to the agent are usually independent Dirichlet distributions that only generate unstructured random worlds. This mismatch between the prior distribution and the domains is problematic to evaluate, since a perfect Bayesian RL algorithm is not guaranteed to perform well given incorrect priors. Since it is not tractable to compute the Bayes-optimal solution exactly, it becomes impossible to decide whether an algorithm that obtains a low return compared to other algorithms in some hand-crafted domain is a better or a worse approximation to the Bayes-optimal solution.

For the purpose of algorithmic evaluation, the obvious solution is to design priors that actually generate the tasks that we solve. When a Bayesian RL algorithm is given this generative model as prior and is also tested on many of these generated tasks, then the Bayes-optimal solution is guaranteed to obtain the best discounted return on average. In this case, a higher mean return becomes synonymous with a better approximation — given the goal of matching the Bayes-optimal solution's performance. We employed this method of averaging across generated domains in Section 4.2 to evaluate the BAMCP algorithm on the Infinite Grid task. To understand the exploration performance of proposed algorithmic solutions truly, future comparisons between algorithms would likely benefit from such evaluation schemes.

5.4 Conclusion

Bayes-adaptive planning is conceptually appealing but computationally very challenging. The enormous computation required by prior approaches is largely due to the fact that root values are computed from expectations and/or maximisations over the complete tree of possible actions and states that follows on from the current history. In addition, these values must integrate over the distribution of transition and potentially reward models at each state in the search tree. As a result, computation typically grows exponentially with search depth, at a rate determined by the action space, successor state space, and model space.

Our new algorithm, BAMCP, builds on previous work (Kearns et al., 1999; Kocsis & Szepesvári, 2006; Silver & Veness, 2010) that solves these problems systematically by *sampling* these expectations, and notably on the POMCP algorithm of Silver and Veness (2010). Only a tiny fraction of the future tree is actually explored, chosen by sampled actions according to their likely worth; and by sampling transitions from the dynamics. Additionally, BAMCP also solves the requirement of integrating over models: by sampling models from the belief distribution; but only only at the root node, so as to avoid the need to compute posteriors throughout the tree; and by lazily avoiding realizing random choices until the last possible moment.

The result is an efficient algorithm that outperforms previous Bayesian model-based reinforcement learning algorithms by a significant margin on several well-known benchmark problems, and that can scale to problems with an infinite state space and a complex prior structure.

Acknowledgments

We acknowledge support for this project from the Gatsby Charitable Foundation (AG, PD), the Natural Sciences and Engineering Research Council of Canada (AG), the Royal Society (DS), and from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 270327 (DS).

Appendix A: List of Acronyms

BAMCP	Bayes-Adaptive Monte-Carlo Planner (Algorithm name)
BAMDP	Bayes-Adaptive Markov Decision Process
BA-UCT	Bayes-Adaptive UCT (Algorithm name)
BEB	Bayesian Exploration Bonus (Algorithm name)
BEETLE	Bayesian Exploration Exploitation Tradeoff in LEarning (Algorithm name)
BFS3	Bayesian Forward Search Sparse Sampling (Algorithm name)
BOLT	Bayesian Optimistic Local Transitions (Algorithm name)
BOSS	Best Of Sampled Set (Algorithm name)
FSSS	Forward Search Sparse Sampling (Algorithm name)
IEQL	Interval Estimation Q-learning (Algorithm name)
MCMC	Monte-Carlo Markov Chain
MCTS	Monte-Carlo Tree Search (Algorithm name)
MDP	Markov Decision Process
PAC	Probably Approximately Correct
POMCP	Partially-Observable Monte-Carlo Planner (Algorithm name)
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
SBOSS	Smarter Best Of Sampled Set (Algorithm name)
UCB1	Upper Confidence Bound 1 (Algorithm name)
UCT	Upper Confidence bounds applied to Trees (Algorithm name)

Appendix B: Algorithms' Implementation

All algorithms below were implemented in C++ (code components were shared across algorithms as much as possible):

- **BAMCP** - The algorithm presented in Section 3, implemented with root sampling, lazy sampling, and rollout learning. The algorithm was run for different number of simulations (10 to 10000) to span different planning times. In all experiments, we set π_{ro} to be an ϵ-greedy policy with $\epsilon = 0.5$. The UCT exploration constant was left unchanged for all experiments ($c = 3$), we experimented with other values of $c \in \{0.5, 1, 5\}$ with similar results.

- **SBOSS** (Castro & Precup, 2010): for each domain, we varied the number of samples $K \in \{2, 4, 8, 16, 32\}$ and the resampling threshold parameter $\delta \in \{3, 5, 7\}$.

- **BEB** (Kolter & Ng, 2009): for each domain, we varied the bonus parameter $\beta \in \{0.5, 1, 1.5, 2, 2.5, 3, 5, 10, 15, 20\}$.

- **BFS3** (Asmuth & Littman, 2011) for each domain, we varied the branching factor $C \in \{2, 5, 10, 15\}$ and the number of simulations (10 to 2000). The depth of search was set to 15 in all domains except for the larger grid and maze domain where it was set to 50. We also tuned the $V_{\max}$ parameter for each domain — $V_{\min}$ was always set to 0.

Code for this paper can be found online on the first author's website, or directly by following this GitHub link `https://github.com/acguez/bamcp`.

Appendix C: Inference Details for the Infinite 2D Grid Task

We construct a Markov Chain using the Metropolis-Hastings algorithm to sample from the posterior distribution of row and column parameters given observed transitions, following the notation introduced in Section 4.2. Let $O = \{(i, j)\}$ be the set of observed reward locations, each associated with an observed reward $r_{ij} \in \{0, 1\}$. The proposal distribution chooses a row-column pair (i_p, j_p) from O uniformly at random, and samples $\tilde{\mathrm{p}}_{i_p} \sim \mathrm{Beta}(\alpha_1 + m_1, \beta_1 + n_1)$ and $\tilde{\mathrm{q}}_{j_p} \sim \mathrm{Beta}(\alpha_2 + m_2, \beta_2 + n_2)$, where $m_1 = \sum_{(i,j)\in O} \mathbf{1}_{i=i_p} r_{ij}$ (i.e., the sum of rewards observed on that column) and $n_1 = (1 - {}^{\beta_2}/_{2(\alpha_2+\beta_2)}) \sum_{(i,j)\in O} \mathbf{1}_{i=i_p}(1 - r_{ij})$, and similarly for m_2, n_2 (mutatis mutandis). The n_1 term for the proposed column parameter $\tilde{\mathrm{p}}_i$ has this rough correction term, based on the prior mean failure of the row parameters, to account for observed 0 rewards on the column due to potentially low row parameters. Since the proposal is biased with respect to the true conditional distribution (from which we cannot sample), we also prevent the proposal distribution from getting too peaked. Better proposals (e.g., taking into account the sampled row parameters) could be devised, but they would likely introduce additional computational cost and the proposal above generated large enough acceptance probabilities (generally above 0.5 for our experiments). All other parameters p_i, q_j such that i or j is present in O are kept from the last accepted samples (i.e., $\tilde{\mathrm{p}}_i = p_i$ and $\tilde{\mathrm{q}}_j = p_j$ for these is and js), and all parameters p_i, q_j that are not linked to observations are (lazily) resampled from the prior — they do not influence the acceptance probability. We denote by $Q(\mathbf{p}, \mathbf{q} \to \tilde{\mathbf{p}}, \tilde{\mathbf{q}})$ the probability of proposing the set of parameters $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ from the last accepted sample of column/row parameters $\mathbf{p}$ and $\mathbf{q}$. The acceptance probability A can then be computed as $A = \min(1, A')$ where:

$$
\begin{aligned}
A' &= \frac{P(\tilde{\mathbf{p}}, \tilde{\mathbf{q}}\,|h)Q(\tilde{\mathbf{p}}, \tilde{\mathbf{q}} \to \mathbf{p}, \mathbf{q})}{P(\mathbf{p}, \mathbf{q}\,|h)Q(\mathbf{p}, \mathbf{q} \to \tilde{\mathbf{p}}, \tilde{\mathbf{q}})} \\
&= \frac{P(\tilde{\mathbf{p}}, \tilde{\mathbf{q}})Q(\tilde{\mathbf{p}}, \tilde{\mathbf{q}} \to \mathbf{p}, \mathbf{q})P(h|\,\tilde{\mathbf{p}}, \tilde{\mathbf{q}})}{P(\mathbf{p}, \mathbf{q})Q(\mathbf{p}, \mathbf{q} \to \tilde{\mathbf{p}}, \tilde{\mathbf{q}})P(h|\,\mathbf{p}, \mathbf{q})} \\
&= \frac{p_{i_p}^{m_1}(1-p_{i_p})^{n_1} q_{j_p}^{m_2}(1-q_{j_p})^{n_2} \prod_{(i,j)\in O} \mathbb{1}[i = i_p \text{ or } j = j_p](\tilde{\mathrm{p}}_i\,\tilde{\mathrm{q}}_j)^{r_{ij}}(1-\tilde{\mathrm{p}}_i\,\tilde{\mathrm{q}}_j)^{1-r_{ij}}}{\tilde{\mathrm{p}}_{i_p}^{m_1}(1-\tilde{\mathrm{p}}_{i_p})^{n_1}\,\tilde{\mathrm{q}}_{j_p}^{m_2}(1-\tilde{\mathrm{q}}_{j_p})^{n_2} \prod_{(i,j)\in O} \mathbb{1}[i = i_p \text{ or } j = j_p](p_i q_j)^{r_{ij}}(1-p_i q_j)^{1-r_{ij}}}.
\end{aligned}
$$

The last accepted sampled is employed whenever a sample is rejected. Finally, reward values R_{ij} are resampled lazily based on the last accepted sample of the parameters p_i, q_j, when they have not been observed already. We omit the implicit deterministic mapping to obtain the dynamics $\mathcal{P}$ from these parameters.

Appendix D: On the Existence of the Bayes-Optimal Policy

As described in Definition 1, Martin (1967) proves the following statement for MDPs with finite state spaces.

Theorem 3 *(Martin, 1967, Thm. 3.2.1) Let $v(s, h, \tilde{\pi})$ be the expected discounted return in an* MDP *(with $|S|$ and $|A|$ finite) when the process starts from the augmented state $\langle s, h \rangle$ and the EE policy $\tilde{\pi}$ is used. Let*

$$v^*(s, h) = \sup_{\tilde{\pi} \in \tilde{\Pi}} v(s, h, \tilde{\pi}). \tag{42}$$

Then there is a policy $\tilde{\pi}^* \in \tilde{\Pi}$ such that $v^*(s, h) = v(s, h, \tilde{\pi}^*)$.

The proof of Theorem 3 consists in proving that the set of EE policies $\tilde{\Pi}$ can be mapped into the a compact subset of the real line, and that the mapping of the function $v(s, h, \cdot)$ is continuous on this set. The proof requires an ordering of histories that relies on the finiteness of the state space. Let $N = |S|$, then the history ordering employed by Martin (1967) is:

$$z(h_{t'}) \equiv \sum_{t=1}^{t'} s_t N^{-t+1}, \tag{43}$$

where $z(h)$ is the number corresponding to history h in the order. In general $|S|$ is not finite, but in some scenarios we may bound by N_t the number of states the agent can be in at time t (for example in the Infinite Grid scenario). For these scenarios we consider the following ordering of histories:

$$w(h_{t'}) \equiv \sum_{t=1}^{t'} s_t \prod_{k=0}^{t} N_k^{-1}, \tag{44}$$

which reduces to $w(h) = z(h)$ if $N_t = N \;\; \forall t$. With this ordering, the proof of Theorem 3 can then be carried out as by Martin (1967) (with minimal modifications) to prove the statement for these more general MDPs - their state space is infinite but the possible states of the agent grows in a controlled manner over time.

Although it is reassuring to know that the Bayes-optimal policy exists for these additional cases, in practice we are satisfied with ϵ approximation to the Bayes-optimal policy and the existence of ϵ-Bayes-optimal policies is likely to be guaranteed in even more general scenarios.

References

Agrawal, S., & Goyal, N. (2011). Analysis of Thompson sampling for the multi-armed bandit problem. Arxiv preprint.

Araya-López, M., Buffet, O., & Thomas, V. (2012). Near-optimal BRL using optimistic local transitions. In *Proceedings of the 29th International Conference on Machine Learning*.

Asmuth, J., Li, L., Littman, M., Nouri, A., & Wingate, D. (2009). A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 19–26.

Asmuth, J., & Littman, M. (2011). Approaching Bayes-optimality using Monte-Carlo tree search. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pp. 19–26.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, *47*(2), 235–256.

Bellman, R. (1954). The theory of dynamic programming. *Bull. Amer. Math. Soc*, *60*(6), 503–515.

Brafman, R., & Tennenholtz, M. (2003). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, *3*, 213–231.

Bubeck, S., Munos, R., & Stoltz, G. (2009). Pure exploration in multi-armed bandits problems. In *Proceedings of the 20th international conference on Algorithmic learning theory*, pp. 23–37. Springer-Verlag.

Castro, P., & Precup, D. (2010). Smarter sampling in model-based Bayesian reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pp. 200–214. Springer.

Castro, P. (2007). *Bayesian exploration in Markov decision processes*. Ph.D. thesis, McGill University.

Castro, P., & Precup, D. (2007). Using linear programming for Bayesian exploration in Markov decision processes. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 2437–2442.

Coquelin, P., & Munos, R. (2007). Bandit algorithms for tree search. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pp. 67–74.

Cozzolino, J., Gonzalez-Zubieta, R., & Miller, R. (1965). Markov decision processes with uncertain transition probabilities. Tech. rep., 11, Operations Research Center, MIT.

Davies, S., Ng, A., & Moore, A. (1998). Applying online search techniques to reinforcement learning. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 753–760.

Dayan, P., & Sejnowski, T. (1996). Exploration bonuses and dual control. *Machine Learning*, *25*(1), 5–22.

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 761–768.

Doshi-Velez, F., Wingate, D., Roy, N., & Tenenbaum, J. (2010). Nonparametric bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Duff, M. (2003). Design for an optimal probe. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 131–138.

Duff, M. (2002). *Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Processes*. Ph.D. thesis, University of Massachusetts Amherst.

Fonteneau, R., Busoniu, L., & Munos, R. (2013). Optimistic planning for belief-augmented Markov decision processes. In *IEEE International Symposium on Adaptive Dynamic Programming and reinforcement Learning (ADPRL 2013)*.

Friedman, N., & Singer, Y. (1999). Efficient Bayesian parameter estimation in large discrete domains. *Advances in Neural Information Processing Systems (NIPS)*, *1*(1), 417–423.

Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., & Teytaud, O. (2012). The grand challenge of computer Go: Monte Carlo tree search and extensions. *Communications of the ACM*, *55*(3), 106–113.

Gelly, S., & Silver, D. (2007). Combining online and offline knowledge in UCT. In *Proceedings of the 24th International Conference on Machine learning*, pp. 273–280.

Gittins, J., Weber, R., & Glazebrook, K. (1989). *Multi-armed bandit allocation indices*. Wiley Online Library.

Guez, A., Silver, D., & Dayan, P. (2012). Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1034–1042.

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, *4*(2), 100–107.

Jaksch, T., Ortner, R., & Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *The Journal of Machine Learning Research*, *99*, 1563–1600.

Kearns, M., Mansour, Y., & Ng, A. (1999). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pp. 1324–1331.

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Machine Learning: ECML*, pp. 282–293. Springer.

Kolter, J., & Ng, A. (2009). Near-Bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 513–520.

Madani, O., Hanks, S., & Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, *147*(1), 5–34.

Martin, J. (1967). *Bayesian decision problems and Markov chains*. Wiley.

Meuleau, N., & Bourgine, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, *35*(2), 117–154.

Mundhenk, M., Goldsmith, J., Lusena, C., & Allender, E. (2000). Complexity of finite-horizon markov decision process problems. *Journal of the ACM (JACM)*, *47*(4), 681–720.

Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods. Tech. rep., University of Toronto.

Poupart, P., Vlassis, N., Hoey, J., & Regan, K. (2006). An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 697–704. ACM.

Ross, S., Pineau, J., Chaib-draa, B., & Kreitmann, P. (2011). A Bayesian approach for learning and planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, *12*, 1729–1770.

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

Ross, S. (1983). *Introduction to stochastic dynamic programming: Probability and mathematical*. Academic Press, Inc.

Schmidhuber, J. (1991). Curious model-building control systems. In *IEEE International Joint Conference on Neural Networks*, pp. 1458–1463.

Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2164–2172.

Silver, D., Sutton, R. S., & Müller, M. (2008). Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, pp. 968–975. ACM.

Silver, E. (1963). Markovian decision processes with uncertain transition probabilities or rewards. Tech. rep., DTIC Document.

Strehl, A., Li, L., & Littman, M. (2009). Reinforcement learning in finite MDPs: PAC analysis. *The Journal of Machine Learning Research*, *10*, 2413–2444.

Strens, M. (2000). A Bayesian framework for reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 943–950.

Sutton, R. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, Vol. 216, p. 224. Citeseer.

Sutton, R., & Barto, A. (1998). *Reinforcement learning*. MIT Press.

Szepesvári, C. (2010). *Algorithms for reinforcement learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Thompson, W. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3/4), 285–294.

Tonk, S., & Kappen, H. (2010). Optimal exploration as a symmetry breaking phenomenon. Tech. rep., Radboud University Nijmegen.

Vien, N. A., & Ertel, W. (2012). Monte carlo tree search for bayesian reinforcement learning. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, Vol. 1, pp. 138–143. IEEE.

Walsh, T., Goschin, S., & Littman, M. (2010). Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*.

Wang, T., Lizotte, D., Bowling, M., & Schuurmans, D. (2005). Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the 22nd International Conference on Machine learning*, pp. 956–963.

Wang, Y., Won, K., Hsu, D., & Lee, W. (2012). Monte Carlo Bayesian reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*.

Watkins, C. (1989). *Learning from delayed rewards*. Ph.D. thesis, Cambridge.

Wingate, D., Goodman, N., Roy, D., Kaelbling, L., & Tenenbaum, J. (2011). Bayesian policy search with policy priors. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.