Monte-Carlo utility estimates for Bayesian reinforcement learning

Christos Dimitrakakis

Abstract— **This paper introduces a set of algorithms for Monte-Carlo Bayesian reinforcement learning. Firstly, Monte-Carlo estimation of upper bounds on the Bayes-optimal value function is employed to construct an optimistic policy. Secondly, gradient-based algorithms for approximate upper and lower bounds are introduced. Finally, we introduce a new class of gradient algorithms for Bayesian Bellman error minimisation. We theoretically show that the gradient methods are sound. Experimentally, we demonstrate the superiority of the upper bound method in terms of reward obtained. However, we also show that the Bayesian Bellman error method is a close second, despite its significant computational simplicity.**

I. Introduction

Bayesian reinforcement learning [1], [2] is the decision-theoretic approach [3] to solving the reinforcement learning problem. Unfonrtunately, calculating posterior distributions can be computationally expensive. Morever, the Bayes-optimal decision can be intractable [4], [5], [1], and even calculating an optimal solution in a restricted class can be difficult [6]. This paper proposes a set of algorithms that take actions by estimating bounds on the Bayes-optimal utility through sampling. They include a direct Monte-Carlo approach, as well as gradient-based approaches. We demonstrate the effectiveness of the proposed algorithms experimentally.

A. Setting

In the *reinforcement learning problem*, an agent is acting in some unknown Markovian environment $\mu \in \mathcal{M}$, according to some policy $\pi \in \Pi$. The agent's policy is a procedure for selecting actions, with the action at time t being $a_t \in \mathcal{A}$. The environment reacts to this action with a sequence of states $s_t \in \mathcal{S}$ and rewards $r_t \in \mathbb{R}$. Since the agent may be learning from experience, this interaction may depend on the complete history, $h_t \in \mathcal{H}$, where $\mathcal{H} \triangleq (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^*$ is the set of all state action reward sequences.

The complete *Markov decision process* (MDP) is specified as follows. The agent's action at time t depends on the history observed so far:

$$a_t \mid h_t = (s^t, r^t, a^{t-1}) \sim \mathbb{P}^\pi(a_t \mid s^t, r^t, a^{t-1}), \qquad (1)$$

where s^t is a shorthand for the sequence $(s_i)_{i=1}^t$; similarly, we use s_k^t for $(s_i)_{i=k}^t$. We denote the environment's response at time $t+1$ given the history at time t by:

$$s_{t+1}, r_{t+1} \mid h_t = (s^t, r^t, a^t) \sim \mathbb{P}_\mu(s_{t+1}, r_{t+1} \mid s_t, a_t) \quad (2)$$

Finally, the agent's goal is determined through its utility:

$$U = \sum_{t=1}^\infty \gamma^{t-1} r_t, \qquad (3)$$

which is a discounted sum of the total instantaneous rewards obtained, with $\gamma \in [0, 1]$. Without loss of generality, we assume that $U \in [0, U_{\max}]$. The optimal agent policy maximises U in expectation, i.e.

$$\max_{\pi \in \Pi} \mathbb{E}^\pi_\mu U, \qquad (4)$$

where $\mathbb{P}^\pi_\mu, \mathbb{E}^\pi_\mu$ denote probabilities and expectations under the process jointly specified by μ, π. However, as in the reinforcement learning problem the environment μ is unknown, the above maximisation is ill-posed. Intuitively, the agent can increase its expected utility by either: (i) Trying to better estimate μ in order to perform the maximisation later (exploration), or (ii) Use a best-guess estimate of μ to obtain high rewards (exploitation).

In order to solve this trade-off, we can adopt a Bayesian viewpoint [3], [7], where we consider a (potentially infinite) set of environment models $\mathcal{M}$. In particular, we select a *prior probability* measure ξ on $\mathcal{M}$. For an appropriate subset $B \subset \mathcal{M}$, the quantity $\xi(B)$ describes our initial belief that the correct model lies in B. We can now formulate the alternative goal of maximising the expected utility with respect to ξ:

$$\mathbb{E}^*_\xi U \triangleq \max_\pi \mathbb{E}^\pi_\xi U = \max_\pi \int_{\mathcal{M}} (\mathbb{E}^\pi_\mu U) \, \mathrm{d}\xi(\mu). \qquad (5)$$

This makes the problem formally sound. A policy $\pi^*_\xi \in \arg\max_\pi \mathbb{E}^\pi_\xi U$ is called *Bayes-optimal* as it solves the exploration-exploitation problem with respect to out prior belief ξ. However, its computation is generally hard [8] even in restricted classes of policies [6]. On the other hand, simple heuristics such as Thompson sampling [9], [1] provide an efficient trade-off [10], [11] between exploration and exploitation.

B. Related work and our contribution

One difficulty that arises when adopting a Bayesian approach to sequential decision making is that in many interesting problems, the posterior calculation itself requires approximations, mainly due to partial observability [12], [4]. The second and more universal problem, which we consider in this paper, is that finding the Bayes-optimal policy is hard, as the set of policies we must consider grows exponentially with the horizon T. However, heuristics exist which, given the current posterior, can obtain a near-optimal policy [13], [14], [1], [15], [6], [16]. In this paper we shall focus on model-based algorithms that use approximate lower and upper bounds on the Bayes-optimal utility to select actions.

The general idea of computing lower and upper bounds via Monte-Carlo sampling in model-based Bayesian reinforcement learning was introduced in [5]. This sampled MDP models from the current belief to estimate stochastic upper and lower bounds. These bounds were then used to perform a stochastic branch and bound search for an optimal policy. In a follow-up paper [6], an attempt was made to obtain tighter lower bounds by finding a good memoryless policy. An earlier class of approaches involving lower bounds is the work of [16], which sampled beliefs rather than MDPs to construct lower bound approximations.

In order to perform the approximations, we also introduce a number of gradient-based algorithms. Relevant work in this domain includes the Gaussian process (GP) based algorithms suggested by [17], [18] and [19]. In particular, [17] performs an incremental temporal-difference fit of the value function using GPs, implicitly using the empirical model of the process. The other two approaches are model-based, with [18] estimating a gradient direction for policy improvement by drawing sample trajectories from the marginal distribution. An analytic solution to the problem of policy improvement with GPs is given by [19], which however relies on the *expected* transition kernel of the process and so does not appear to take the model uncertainty into account.

The approaches suggested in this paper are considerably simpler, as well as more general, in that they are applicable to any Bayesian model of the Markov process and parametrisation of the value function. The fundamental idea stems from the observation that, in order to estimate the Bayes-utility of a policy, we can draw sample MDPs from the posterior, calculate the (either current policy's, or the optimal) utility for each MDP and average. The same effect can be achieved in an iterative procedure, by drawing only one MDP, estimating the utility of our policy, and then adjusting our parameters to approach the sampled utility. This can be achieved with gradient methods. Finally, we use the same sampling idea to minimise the Bellman error of the Bayes-expected value function, in a fully incremental fashion that explicitly takes into account the model uncertainty.

II. Gradient Bayesian reinforcement learning

Imagine that the history $h_t \in \mathcal{H}$ of length t has been generated from $\mathbb{P}^\pi_\mu$, the process defiend by an MDP $\mu \in \mathcal{M}$ controlled with a history-dependent policy π. Now consider a prior belief ξ_0 on $\mathcal{M}$ with the property that $\xi_0(\cdot \mid \pi) = \xi_0(\cdot)$, i.e. that the prior is independent of the policy used. Then the posterior probability, given a history h_t generated by a policy π, that $\mu \in B$ can be written as:

$$\xi_t(B \mid \pi) \triangleq \xi_0(B \mid h_t, \pi) = \frac{\int_B \mathbb{P}^\pi_\mu(h_t) \,\mathrm{d}\xi_0(\mu)}{\int_{\mathcal{M}} \mathbb{P}^\pi_\mu(h_t) \,\mathrm{d}\xi_0(\mu)}. \tag{6}$$

Fortunately, the dependence on the policy can be removed, since the posterior is the same for all policies that put non-zero mass on the observed data. Thus, in the sequel we shall simply write ξ_t for the posterior probability over MDPs at time t.

A. Value functions

Value functions are an important concept in reinforcement learning. Briefly, a value function $V^\pi_\mu : \mathcal{S} \to \mathbb{R}$ gives the expected utility for the policy π acting in an MDP μ, given that we start at state s, i.e.

$$V^\pi_\mu(s) \triangleq \mathbb{E}^\pi_\mu(U \mid s_t = s). \tag{7}$$

A similar notion is expressed by the Q-value function $Q^\pi_\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which is the expected utility for the policy π acting in an MDP μ, given that we start at state s and take action a, i.e.

$$Q^\pi_\mu(s, a) \triangleq \mathbb{E}^\pi_\mu(U \mid s_t = s, a_t = a). \tag{8}$$

Similarly, and with a slight abuse of notation, we define the *Bayesian* value function $V^\pi_\xi : \mathcal{S} \to \mathbb{R}$, and the related Bayesian Q-value function $Q^\pi_\xi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. These are defined for any belief ξ and policy π to be the corresponding expected value functions over all MDPs.

$$V^\pi_\xi(s) \triangleq \int_{\mathcal{M}} V^\pi_\mu(s) \,\mathrm{d}\xi(\mu) \tag{9}$$

$$Q^\pi_\xi(s, a) \triangleq \int_{\mathcal{M}} Q^\pi_\mu(s, a) \,\mathrm{d}\xi(\mu) \tag{10}$$

Due to the convexity of the Bayes-optimal expected utility [3] with respect to the belief ξ, it can be bounded from above and below also for the Bayesian RL problem [5]:

$$\int_{\mathcal{M}} \max_\pi (\mathbb{E}^\pi_\xi U) \,\mathrm{d}\xi(\mu) \geq \mathbb{E}^*_\xi(U) \geq \mathbb{E}^{\pi'}_\xi(U), \qquad \forall \pi' \in \Pi. \tag{11}$$

Since it hard to find the Bayes-optimal policy [8], [20], [3], [5], we may instead try and estimate upper and lower bounds on the expected utility, and consequently, on the Q-value function. These can then be used to implement a heuristic policy that is either exploratory (when we use upper bounds) or conservative (when we use lower bounds).

To achieve this, we propose a number of simple algorithms. First, we describe the direct upper bound estimation proposed in [5] in the context of tree search. Here, we apply it to select a policy directly, in a manner similar to the lower bound approach in [6]. We then describe gradient-based incremental versions of both algorithms. However, all of these algorithms require estimating the value function of a sampled MDP, a potentially expensive process. For this reason, we also derive a gradient-based algorithm for minimising the Bayes-value function Bellman error. This is shown to perform almost as well as the previous algorithms, with significantly less computational effort.

B. Direct upper bound estimation

The idea of the following algorithm stems directly from the definition of the upper bound (5). In fact, [5] had previously used such upper bounds in order to guide tree search, while [6] had used *lower bounds* directly for taking actions. However, to our knowledge the simple idea of estimating the upper bound (5) and using it to directly take actions has never been tried before in practice.

We can estimate an upper bound value vector[1] q by direct Monte Carlo sampling[2] from our belief ξ:

$$q_{s,a} = \frac{1}{N} \sum_{i=1}^{N} Q^*_{\mu_i}(s,a), \qquad \mu_i \sim \xi. \tag{12}$$

This idea is significantly simpler than that constructing *credible intervals* (see for example [22]). In addition, estimation of $Q^*_{\mu_i}$ for each sampled MDP is easy. This is in contrast with the lower bound approach advocated in [6].

Algorithm 1 U-MCBRL: Upper-bound Monte-Carlo Bayesian RL

```
**Input** prior $\xi_0$, value vector $q$, initial state $s_0$, number of samples $N$.
for $t = 0, \dots$ do
    if switch-policy then
        $\mu_1, \dots, \mu_k \sim \xi_t$                          // Sample $N$ MDPs
        $q = \frac{1}{N} \sum_{i=1}^{N} Q^*_{\mu_i}$                 // Get $Q$ upper bound.
    end if
    $a_t = \arg\max_{a \in \mathcal{A}} q_{s,a}$.                // Act in the real MDP
    $s_{t+1}, r_{t+1} \sim \mu$                      // Observe next state, reward
    $\xi_{t+1}(\cdot) = \xi_t(\cdot \mid s_{t+1}, r_{t+1}, s_t, a_t)$    // Update posterior
end for
```

The algorithm is presented in Alg.1. A hyperparameter of the algorithm is the number of samples N to take at each iteration, as well as the points at which to switch policy[3]. This paper uses the simple strategy of linearly incrementing the switching interval. Let us now see how we can directly approximate both lower bounds such as those in [6], and upper bounds, such as this in Alg. 1, via gradient methods.

C. Direct gradient approximation

We now present a simple class of algorithms for gradient Bayesian reinforcement learning. First, let us consider the estimation for a specific policy π, which will correspond to approximating a lower bound. Define a family of functions $v_\theta : \mathcal{S} \to \mathbb{R}$, $\{ v_\theta \mid \theta \in \Theta \}$. We consider the problem of estimating the expected value function given some belief ξ:

$$\min_{\theta \in \Theta} f(\theta), \qquad f(\theta) \triangleq \int_{\mathcal{S}} g(\theta; s) \,\mathrm{d}\chi(s), \tag{13}$$

$$g(\theta; s) \triangleq \| v_\theta(s) - V^\pi_\xi(s) \| \tag{14}$$

where χ is a measure on $\mathcal{S}$, and $\| \cdot \|$ is the Euclidean norm. Then the derivative of (14) can be written as:

$$\nabla_\theta g(\theta; s) = 2 \left(v_\theta(s) - V^\pi_\xi(s) \right) \nabla_\theta v_\theta(s). \tag{15}$$

Let $\omega_k(s) = V^\pi_{\mu_k}(s)$, be the value function of an MDP sampled from the belief, i.e. $\mu_k \sim \xi$. Then, due to the linearity of expectations, it is easy to see that:

$$\nabla_\theta g(\theta; s) = \mathbb{E}_\xi \left[2 \left(v_\theta(s) - \omega_k(s) \right) \nabla_\theta v_\theta(s) \right]. \tag{16}$$

Consequently, ω_k can be used to obtain the following stochastic approximation [23], [24] algorithm

$$\theta_{k+1} = \theta_k - \eta_k \left(v_\theta(s) - \omega_k(s) \right) \nabla_\theta v_\theta(s), \tag{17}$$

where η_k must be a step-size parameter satisfying $\sum_k \eta_k = \infty$, $\sum_k \eta_k^2 < \infty$. A similar approach can be used to estimate the Q-value function with an approximation $q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

$$\theta_{k+1} = \theta_k - \eta_k \left(q_\theta(s,a) - \omega_k(s,a) \right) \nabla_\theta q_\theta(s,a), \tag{18}$$

where $\omega_k(s,a) = Q^\pi_{\mu_k}(s,a)$. This update can also be performed over the complete state-action space

$$\theta_{k+1} = \theta_k - \eta_k \sum_{s,a} D_k(s,a), \tag{19}$$

$$D_k(s,a) = \left(q_\theta(s,a) - \omega_k(s,a) \right) \nabla_\theta q_\theta(s,a). \tag{20}$$

The same procedure can be applied to approximate the upper bound (11). This only requires a trivial modification to the above algorithms, by setting $\omega_k(s) = V^*_{\mu_k}(s)$ or $\omega_k(s,a) = Q^*_{\mu_k}(s,a)$ in either case. It is easy to see that the above approximation still holds.

Algorithm 2 DGBRL: Direct gradient Bayesian RL.

```
**Input** prior $\xi_0$, parameters $\theta_0$, initial state $s_0$
for $t = 0, \dots$ do
    $\mu_t \sim \xi_t$                                   // Sample an MDP
    $\omega_t = Q^\pi_{\mu_t}$ (or $Q^*_\mu$)                      // Get value of sample
    $\theta_{t+1} = \theta_t - \eta_k \sum_{s,a} D_k(s,a)$      // Update parameters
    $a_t = \arg\max_{a \in \mathcal{A}} q_{\theta_t}(s,a)$.      // Act in the real MDP
    $s_{t+1}, r_{t+1} \sim \mu$                     // Observe next state, reward
    $\xi_{t+1}(\cdot) = \xi_t(\cdot \mid s_{t+1}, r_{t+1}, s_t, a_t)$    // Update posterior
end for
```

To make the approximation faster, we can take a single MDP sample at every step, take an action, and then use the previous approximation for the next step. If the belief ξ_t changes sufficiently slowly then this will be almost as good as taking multiple samples and finding the best approximation at every step. The complete algorithm is shown in Alg.2. The advantage of this idea over the upper and lower bound approach advocated in [5], [6], is that we can reuse information from previous steps without needing to take multiple MDP samples.

In either case, the computational difficulty is the calculation of V_{μ_k}, which we still need to do once at every step. The next section discusses another idea where the complete estimation of a value function for each sampled MDP is not required.

D. Temporal difference-like error minimisation

One alternative idea is to simply estimate a consistent value function approximation, similar to those used in temporal-difference (TD) methods (in particular the gradient-based view of TD-like methods in [24]). The general idea is

[1] For continuous spaces, this can be defined on a set of representative states.

[2] Due to the Hoeffding bound [21] and the boundedness of the value function, it is easy to see that this estimate is $O(1/\sqrt{N})$-close to the upper bound (11) with high probability.

[3] since re-sampling and calculating new value functions is expensive

to form the following minimisation problem:

$$\min_{\theta \in \Theta} f(\theta), \quad f(\theta) \triangleq \int_{\mathcal{S}} g(\theta; s) \, \mathrm{d}\chi(s), \tag{21}$$

$$g(\theta; s) \triangleq \int_{\mathcal{M}} \|h(\theta; \mu, s)\| \, \mathrm{d}\xi(\mu) \tag{22}$$

$$h(\theta; \mu, s) \triangleq v_\theta(s) - \rho(s) - \gamma \int_{\mathcal{S}} v_\theta(s') \, \mathrm{d}\mathbb{P}^{\pi}_{\mu}(s' \mid s). \tag{23}$$

Now let us sample a state $s_k \sim \chi$ from the state distribution, an MDP $\mu_k \sim \xi$ from the belief and a next state $s'_k \sim \mathbb{P}^{\pi}_{\mu_k}(s' \mid s_k)$ from the transition kernel of the sampled MDP given the sampled current state. Using the euclidean norm for $\| \cdot \|$ and taking the gradient with respect to θ we obtain:

$$D_k = 2h(\theta_k; \mu_k, s_k) \left(\nabla_\theta v_{\theta_k}(s_k) - \gamma \nabla_\theta v_{\theta_k}(s'_k) \right) \tag{24}$$

$$\theta_{k+1} = \theta_k - \eta_k D_k. \tag{25}$$

By choosing an appropriate approximation architecture, e.g. a linear approximation with bounded bases, the following corollary holds:

Corollary 1 *If $\|\nabla_\theta v_\theta\| \leq c$ and $\|\nabla^2_\theta v_\theta\| \leq c'$ with $c, c' < \infty$, then $f(\theta_k)$ converges, with $\lim_{t \to \infty} \nabla_\theta f(\theta_k) = 0$.*

Proof: This results follows from Proposition 4.1 in [24], since the the sequence satisfies the four conditions in Assumption 4.2. (a) $f \geq 0$. (b) f is twice differentiable and its second derivative is bounded, as $\left\| \int_{\mathcal{S}} \nabla^2_\theta v_\theta(s') \, \mathrm{d}\mathbb{P}^{\pi}_{\mu}(s' \mid s) \right\| \leq \int_{\mathcal{S}} \|\nabla^2_\theta v_\theta(s')\| \, \mathrm{d}\mathbb{P}^{\pi}_{\mu}(s' \mid s) \leq c$. (c) By taking expectations over the sample, it is easy to see that $\mathbb{E}\, D_k = \nabla_\theta f(\theta_k)$. (d) follows from the boundedness of the first derivative. ■

E. Bellman error minimisation

An alternative formulation is Bellman error minimisation ([24], Sec. 6.10), where instead of minimising the error with respect to the current policy, we minimise the error over the Bellman operator applied to the current value function. This is simplest to do when we are working with Q-value functions. Then the problem can be written as:

$$\min_{\theta \in \Theta} f(\theta), \quad f(\theta) \triangleq \sum_{a \in \mathcal{A}} \int_{\mathcal{S}} g(\theta; s, a) \, \mathrm{d}\chi(s), \tag{26}$$

$$g(\theta; s, a) \triangleq \int_{\mathcal{M}} \|h(\theta; \mu, s, a)\| \, \mathrm{d}\xi(\mu) \tag{27}$$

$$h(\theta; \mu, s, a) \triangleq q_\theta(s, a) - \rho(s) - \gamma \int_{\mathcal{S}} q_\theta(s', a^*(s')) \, \mathrm{d}\mathbb{P}^{\pi}_{\mu}(s' \mid s) \tag{28}$$

$$a^*(s') \triangleq \arg\max_{a' \in \mathcal{A}} q_\theta(s', a'). \tag{29}$$

Using the same reasoning as in Sec. II-D, we sample $s_k \sim \chi$, $\mu_k \sim \xi$ from the belief and a next state $s'_k \sim \mathbb{P}^{\pi}_{\mu_k}(s' \mid s_k)$ from the transition kernel of the sampled MDP given the sampled current state. Using the euclidean norm for $\| \cdot \|$ and taking the gradient with respect to θ we obtain the algorithm:

$$D_k = 2h(\theta_k; \mu_k, s_k, a_k)[\nabla_{\theta_k} q_\theta(s_k, a_k) \tag{30}$$

$$- \gamma \nabla_\theta q_{\theta_k}(s'_k, a^*(s'_k))]$$

$$\theta_{k+1} = \theta_k - \eta_k D_k. \tag{31}$$

It easy to see that Corollary 1 is also applicable to this update sequence. When the state sequence is generated from a particular policy, rather than being drawn from some distribution χ, we obtain Alg.3.

Algorithm 3 BGBRL: Bellman gradient Bayesian RL

```
Input prior $\xi_0$, parameters $\theta_0$, initial state $s_0$
for $t = 0, \dots$ do
    $\mu_t \sim \xi_t$                                   // Sample an MDP
    $s'_t \sim \mathbb{P}^{\pi}_{\mu_t}(s_{t+1} \mid s_t)$            // Sample a next state
    $\theta_{t+1} = \theta_t - \eta_t D_t$            // Update parameters using (30)
    $a_t = \arg\max_{a \in \mathcal{A}} q_{\theta_t}(s, a)$.    // Act in the real MDP
    $s_{t+1}, r_{t+1} \sim \mu$                  // Observe next state, reward
    $\xi_{t+1}(\cdot) = \xi_t(\cdot \mid s_{t+1}, r_{t+1}, s_t, a_t)$    // Update posterior
end for
```

III. Experiments

We present experiments illustrating the performance of U-MCBRL and BGRL and compare them with other algorithms. In particular we also examine the lower-bound algorithm presented in [6], the well known UCRL [25] algorithm,[4] and $Q(\lambda)$, for completeness.

A. Experiment design

Since all algorithms have hyperparameters, we followed a principled experiment design methodology. Firstly, we selected a set of possible hyperparameter values for each algorithm. For each evaluation domain, we performed 10 runs for each hyperparameter choice and chose the one with the highest total reward over these runs. We then measured the performance of the algorithms over 10^3 runs. This ensures an unbiased evaluation.

Methods	parameter	function
$Q(\lambda)$	ε_0	exploration
UCRL	δ	confidence interval
MCBRL, U-MCBRL	N	number of samples
BGBRL, $Q(\lambda)$	η_0	step size

TABLE I: Automatically tuned hyperparameters

The set of hyper-parameters that were automatically tuned for each method are listed in Table I. For $Q(\lambda)$, we fixed $\lambda = 0.9$ and used an ε-greedy strategy with a decaying rate and tuned initial value ϵ_0. For UCRL, we tune the interval error probability δ. Gradient algorithms require tuning the initial step-size parameter η_0. Monte-Carlo algorithms require tuning the number of samples N. UCRL, MCBRL and U-MCBRL all used the same policy-switching heuristic.

B. Domains

We employed standard domains from discrete-state problems in exploration in reinforcement learning. Thus, Bayesian inference is closed-form, as we can use a Dirichlet-product prior for the transitions and a Normal-Gamma prior

[4] Although UCRL is defined for undiscounted problems, it is trivial to apply to discounted problems by adding replacing average value iteration with discounted value iteration.

for the reward. Value function parametrisation is tabular, i.e. there is one parameter per state-action pair. These domains are the Chain problem [1], River-Swim [26], Double-Loop [1]. In addition, we consider the mountain car domain of [27], using a uniform 5×5 grid as features. All domains employed a discount factor $\gamma = 0.99$.

Chain				
$Q(\lambda)$	1993.9	1999.7	2005.4	3
UCRL	3543.5	3547.5	3551.3	1613
MCBRL	3610.5	3616.1	3621.7	464
U-MCBRL	3617.8	3623.4	3629.1	1560
BGBRL	3593.6	3598.3	3602.7	48
Double Loop				
$Q(\lambda)$	2053.7	2058.1	2062.1	5
UCRL	3841.0	3841.0	3841.0	369
MCBRL	3949.5	3950.2	3951.0	2343
U-MCBRL	3946.7	3947.5	3948.3	5135
BGBRL	3925.3	3926.2	3927.0	96
River Swim				
$Q(\lambda)$	5.0	5.0	5.0	5
UCRL	312.4	313.8	315.3	240
MCBRL	624.0	625.4	626.8	1187
U-MCBRL	626.3	627.6	629.0	2329
BGBRL	600.3	601.7	603.2	69
Mountain Car 5×5				
$Q(\lambda)$	-9957.6	-9957.0	-9956.3	15
UCRL	-9952.9	-9951.6	-9950.3	1908
MCBRL	-9829.1	-9827.2	-9825.5	35733
U-MCBRL	-9811.8	-9810.2	-9808.6	66252
BGBRL	-9883.2	-9881.9	-9880.6	886
Method	95% lower	mean	95% upper	CPU (s)

TABLE II: Total reward and CPU time

C. Results

From the online performance results shown in Fig 1, it is clear that apart from $Q(\lambda)$, all algorithms are performing relatively similarly in the simpler environments. However, UCRL converges somewhat more slowly and is particularly unstable in the Mountain Car domain. [5]

A clearer view of the performance of each algorithm is can be seen in Table II, in terms of the average total reward obtained. It additionally shows the 95% lower and upper confidence bound calculated on the mean (shown in the middle column) via 10^4 bootstrap samples. The best-performing methods in each environment (taking into account the bootstrap intervals) are shown in bold. One immediately notices that MCBRL and U-MCBRL are usually tied for best. This is perhaps not surprising, as they have the same structure: in fact, for $N = 1$, they are equivalent to Thompson sampling [1], as mentioned in [6]. However, MCBRL uses a *lower bound* on the value function, while U-MCBRL an *upper bound*, which makes it more optimistic.[6]

The most significant finding, however is that BGBRL is a relatively close second most of the time, performing better than all the remaining algorithms. This is despite its computational simplicity.

IV. Conclusion

This paper introduced a set of Monte-Carlo algorithms for Bayesian reinforcement learning. The first, U-MCBRL is a modification of a lower-bound algorithm to an upper-bound setting, which has very good performance but has relatively high computational complexity. The second, DGBRL, is a type of gradient-based algorithm for approximating either the lower or the upper bound, but nevertheless does not necessarily alleviate the problem of complexity. Finally, BGBRL defines a novel type of Bellman error minimisation, on the Bayes-expected value function. By performing gradient descent to reduce this error through sampling possible MDPs, we obtain an efficient and highly competitive algorithm.

The algorithms were tested using an unbiased experimental methodology, whereby hyperparameters were automatically selected from a small number of runs. This ensures that algorithmic brittleness is not an issue. In all of those experiments, U-MCBRL and its sibling, MCBRL outperformed all alternatives. However, BGBRL was a close runner-up, even though it is computationally much simpler, as it does not require performing value iteration.

A subject that this paper has not touched upon is the theoretical performance of U-MCBRL and BGBRL. For the first, the results for MCBRL [6] should be applicable with few modifications. The performance analysis of BGBRL-like algorithms, on the other hand, is a completely open question at the moment.

[5] Due to the discretisation, this domain is no longer fully observable.

[6] Although we did not explicitly consider Thompson sampling, we note that the hyperparameter $N = 1$ corresponding to Thompson sampling was never chosen by the automatic procedure as it always had worse performance than taking more samples. Nevertheless, its performance over the 10^3 runs was always significantly worse than those of MCBRL and U-MCBRL.

References

[1] M. Strens, “A Bayesian framework for reinforcement learning,” in *ICML 2000*, 2000, pp. 943–950.

[2] N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart, *Reinforcement Learning*. Springer, 2012, ch. Bayesian Reinforcement Learning, pp. 359–386.

[3] M. H. DeGroot, *Optimal Statistical Decisions*. John Wiley & Sons, 1970.

[4] S. Ross, B. Chaib-draa, and J. Pineau, “Bayes-adaptive POMDPs,” in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008.

[5] C. Dimitrakakis, “Complexity of stochastic branch and bound methods for belief tree search in Bayesian reinforcement learning,” in *2nd international conference on agents and artificial intelligence (ICAART 2010)*, ISNTICC. Valencia, Spain: Springer, 2010, pp. 259–264.

[6] ——, “Robust bayesian reinforcement learning through tight lower bounds,” in *European Workshop on Reinforcement Learning (EWRL 2011)*, ser. LNCS, no. 7188, 2011, pp. 177–188.

[7] L. J. Savage, *The Foundations of Statistics*. Dover Publications, 1972.

[8] M. O. Duff, “Optimal learning computational procedures for Bayes-adaptive Markov decision processes,” Ph.D. dissertation, University of Massachusetts at Amherst, 2002.

[9] W. Thompson, “On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of two Samples,” *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

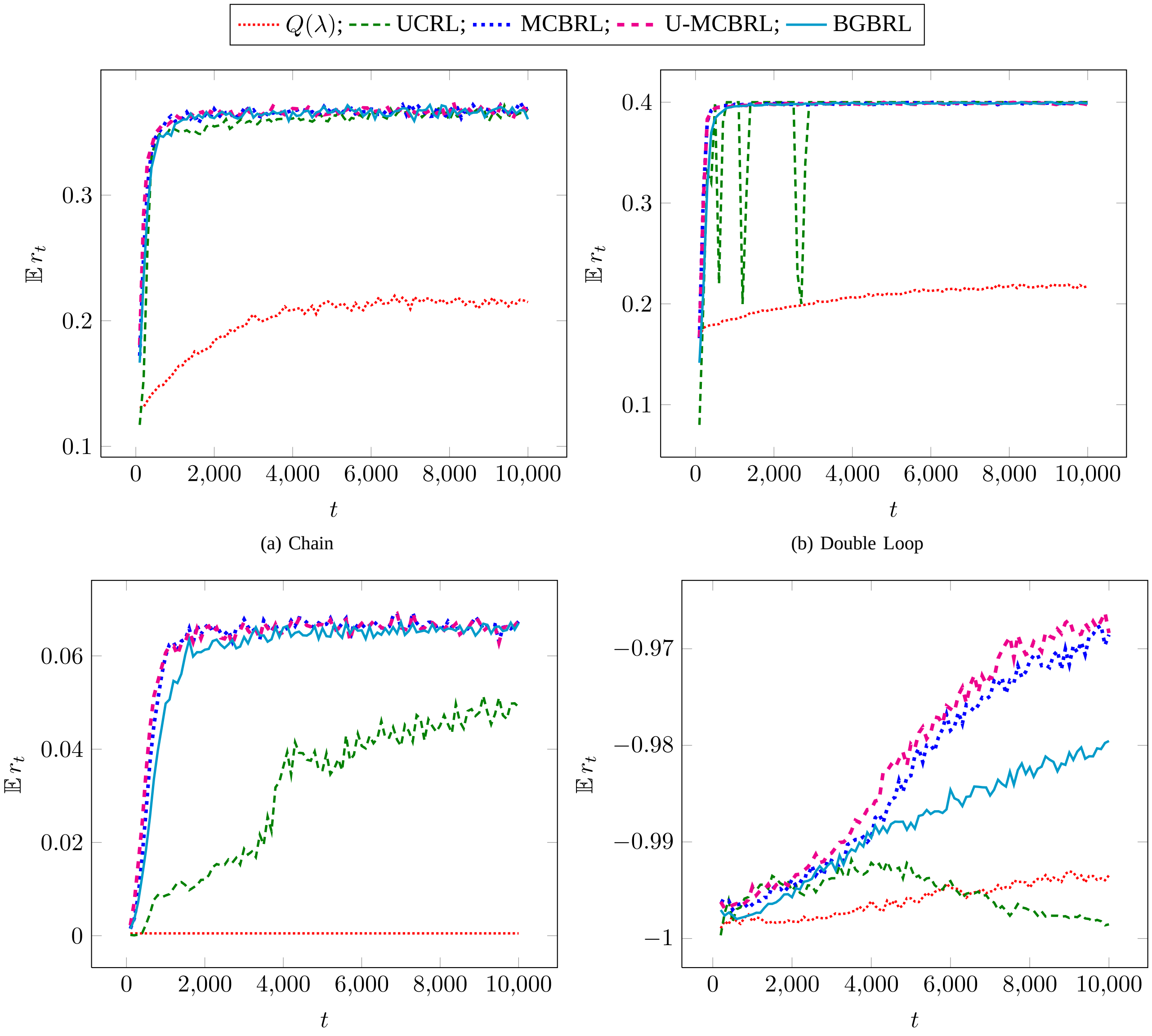
[10] S. Agrawal and N. Goyal, “Analysis of thompson sampling for the multi-armed bandit problem,” in *COLT 2012*, 2012.

[11] E. Kaufmanna, N. Korda, and R. Munos, “Thompson sampling: An optimal finite time analysis,” in *ALT-2012*, 2012.

[12] P. Poupart and N. Vlassis, “Model-based Bayesian reinforcement learning in partially observable domains,” in *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

[13] J. Z. Kolter and A. Y. Ng, “Near-Bayesian exploration in polynomial time,” in *ICML 2009*, 2009.

[14] P. S. Castro and D. Precup, “Using linear programming for Bayesian exploration in Markov decision processes,” in *IJCAI*, M. M. Veloso, Ed., 2007, pp. 2437–2442.



(a) Chain

(b) Double Loop

(c) River Swim

(d) Mountain Car 5×5

Fig. 1: Reward per step, smoothed over 100 steps and averaged over 10^3 runs.

[15] M. Araya, V. Thomas, O. Buffet, *et al.*, “Near-optimal BRL using optimistic local transitions,” in *ICML*, 2012.

[16] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, “An analytic solution to discrete Bayesian reinforcement learning,” in *ICML 2006*. ACM Press New York, NY, USA, 2006, pp. 697–704.

[17] Y. Engel, S. Mannor, and R. Meir, “Bayes meets bellman: The gaussian process approach to temporal difference learning,” in *ICML 2003*, 2003.

[18] M. Ghavamzadeh and Y. Engel, “Bayesian policy gradient algorithms,” in *NIPS 2006*, 2006.

[19] M. P. Deisenroth and C. E. Rasmussen, “Pilco: A model-based and data-efficient approach to policy search,” in *International conference on Machine Learning (ICML)*, Bellevue, WA, USA, July 2011.

[20] R. Dearden, N. Friedman, and D. Andre, “Model based Bayesian exploration,” in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, K. B. Laskey and H. Prade, Eds. San Francisco, CA: Morgan Kaufmann, San Francisco, CA, July 30–Aug. 1 1999, pp. 150–159.

[21] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, March 1963.

[22] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, “Gaussian process optimization in the bandit setting: No regret and experimental design,” in *ICML 2010*, 2010.

[23] H. Robbins and S. Monro, “A stochastic approximation method,” *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[24] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[25] T. Jacksh, R. Ortner, and P. Auer, “Near-optimal regret bounds for reinforcement learning,” *Journal of Machine Learning Research*, vol. 11, pp. 1563–1600, 2010.

[26] A. Strehl and M. Littman, “An analysis of model-based interval estimation for Markov decision processes,” *Journal of Computer and System Sciences*, vol. 74, no. 8, pp. 1309–1331, 2008.

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.