Reward Shaping for Model-Based Bayesian Reinforcement Learning

Hyeoneun Kim, Woosang Lim, Kanghoon Lee, Yung-Kyun Noh and **Kee-Eung Kim**
Department of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon 305-701, Korea
hekim@ai.kaist.ac.kr, quasar17@kaist.ac.kr, khlee@ai.kaist.ac.kr,
nohyung@kaist.ac.kr and kekim@cs.kaist.ac.kr

Abstract

Bayesian reinforcement learning (BRL) provides a formal framework for optimal exploration-exploitation tradeoff in reinforcement learning. Unfortunately, it is generally intractable to find the Bayes-optimal behavior except for restricted cases. As a consequence, many BRL algorithms, model-based approaches in particular, rely on approximated models or real-time search methods. In this paper, we present potential-based shaping for improving the learning performance in model-based BRL. We propose a number of potential functions that are particularly well suited for BRL, and are domain-independent in the sense that they do not require any prior knowledge about the actual environment. By incorporating the potential function into real-time heuristic search, we show that we can significantly improve the learning performance in standard benchmark domains.

Introduction

A reinforcement learning (RL) agent interacts with an unknown environment to maximize the total reward. One of the unique challenges for the agent is the well-known exploration-exploitation tradeoff: without complete knowledge about the environment, the agent has to explore untried actions that may lead to better long-term rewards, but at the same time, it also needs to execute actions that are known to yield the largest rewards given the current knowledge about the environment. Bayesian reinforcement learning (BRL) provides a principled mathematical framework for computing Bayes-optimal actions that achieve an ideal balance between exploration and exploitation.

Although the Bayes-optimal action has a succinct formulation in model-based BRL, where the agent attempts to build an explicit model of the environment for learning, it is computationally intractable except for restricted cases. As a consequence, most model-based BRL algorithms rely on constructing approximated models that are tractable to solve, or real-time heuristic search methods that build search trees on-the-fly.

The main focus of this paper is on shaping rewards for improving the learning performance of BRL algorithms. In shaping, we can transform rewards by a shaping function in order to mitigate the sparsity and delay in rewards. A popular approach to shaping is potential-based shaping, where the domain knowledge is leveraged to encode the desirability of states into the potential function while the optimal policy remains unchanged. We address two main challenges in using shaping for model-based BRL. First, instead of using a static or fixed shaping function defined from a-priori domain knowledge, can we make the shaping function adapt to experiences (i.e. observed transitions)? Second, when shaping is used with heuristic search, can we preserve the soundness and completeness of the search heuristic?

We present a set of domain-independent shaping functions that (1) does not use a-priori knowledge of the true underlying environment, (2) adapts to experiences, and (3) preserves the soundness and completeness of the heuristic used for real-time search. We experimentally show that these shaping functions significantly improve learning performance in standard benchmark domains.

Background

BAMDP: A model-based BRL framework

We start with the model of the underlying environment, which is assumed to be a discrete-state Markov decision process (MDP) defined as a 5-tuple $M = \langle S, A, T, R, \gamma \rangle$, where S is the set of environment states, A is the set of agent actions, $T(s, a, s')$ is the transition probability $\Pr(s'|s, a)$ of making transition to state s' from state s by executing action a, $R(s, a, s') \in [R_{\min}, R_{\max}]$ is the nonnegative reward (*i.e.*, $R_{\min} \geq 0$) earned from the transition $\langle s, a, s' \rangle$, and $\gamma \in [0, 1)$ is the discount factor. The agent that interacts with the environment executes actions by following a policy $\pi : S \rightarrow A$, which prescribes the action to be executed in each state.

The performance criterion for π we use in this paper is the expected discounted return V^π for state s_t at timestep t,

$$V^\pi(s_t) = E[\textstyle\sum_{\tau=t}^{\infty} \gamma^\tau R(s_\tau, \pi(s_\tau), s_{\tau+1})],$$

defined as the *state value function*. The Bellman optimality equation states that the state value function for the optimal policy π^* satisfies

$$V^*(s) = \max_a \textstyle\sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \quad (1)$$

which can be computed by classical dynamic programming algorithms such as value iteration or policy iteration when the MDP model M is completely known. The optimal policy can also be recovered from the optimal *action value function*

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \quad (2)$$

since $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$. On the other hand, when the model M is not known, computing an optimal policy becomes an RL problem.

The uncertainty in the model can be represented using a probability distribution b, also known as the *belief* over the models. Throughout the paper, we shall assume that the reward function is known to the agent, while the transition probability is unknown. Since the transition for each state-action pair is essentially a multinomial distribution, a straightforward way to represent the belief b is to define the parameter $\theta_a^{s,s'}$ for each unknown transition probability $T(s,a,s')$ and use the product of independent Dirichlet distributions as the conjugate prior. When the transition $\langle s,a,s' \rangle$ is observed, the belief b is updated by the Bayes rule

$$\begin{aligned} b_a^{s,s'}(\theta) &\propto \theta_a^{s,s'} b(\theta) \\ &= \theta_a^{s,s'} \textstyle\prod_{\hat{s},\hat{a}} \mathrm{Dir}(\theta_{\hat{a}}^{\hat{s}}(\cdot); n_{\hat{a}}^{\hat{s}}(\cdot)) \\ &= \textstyle\prod_{\hat{s},\hat{a}} \mathrm{Dir}(\theta_{\hat{a}}^{\hat{s}}(\cdot); n_{\hat{a}}^{\hat{s}}(\cdot) + \delta_{\hat{s},\hat{a},\hat{s'}}(s,a,s')), \end{aligned}$$

which is equivalent to incrementing the single parameter that corresponds to the observed transition, *i.e.*, $n_a^s(s') \leftarrow n_a^s(s') + 1$.

Bayes-Adaptive MDP (BAMDP) provides a succinct planning formulation of the Bayes-optimal action under uncertainty in the model (Duff 2002). By augmenting the state with the belief b, the Bayes-optimal policy of the BAMDP should satisfy the optimality equation

$$V^*(s,b) = \max_a \sum_{s'} T_b(s,a,s')(R(s,a,s') + \gamma V^*(s', b_a^{s,s'})),$$

where $T_b(s,a,s') = E[\Pr(s'|s,a,b)] = n_a^s(s') / \sum_{s''} n_a^s(s'')$. Unfortunately, computing an optimal policy of the BAMDP is known to be computationally intractable in general cases.

One of the popular approaches to mitigate the intractability result is the real-time search, such as Bayes-Adaptive Monte-Carlo Planning (BAMCP) (Guez, Silver, and Dayan 2012), Bayesian Optimistic Planning (BOP) (Fonteneau, Busoniu, and Munos 2013), and Bayesian Forward Search Sparse Sampling (BFS3) (Asmuth and Littman 2011). These approaches leverage the computation time allowed between consecutive action executions to construct a lookahead search tree to find the best action.

On the other hand, there are other approaches that do not involve lookahead search. They often use an approximation of BAMDP that is tractable to solve, while guaranteeing sufficient approximation to the Bayes-optimal policy with high probability. PAC-BAMDP algorithms such as Best of Sampled Set (BOSS) (Asmuth et al. 2009), Smart BOSS (Castro and Precup 2010), Bayesian Exploration Bonus (BEB) (Kolter and Ng 2009), and Bayesian Optimistic Local Transitions (BOLT) (Araya-López, Thomas, and Buffet 2012) belong to these approaches.

Finally, we should remark that BAMDP is essentially a hybrid-state Partially Observable MDP (POMDP) where the (hidden) state is the pair $\langle s, \theta \rangle$, the observation is the environment state, while the action and the reward remain the same as in the environment MDP (Poupart et al. 2006). In this formulation, the transition probability is defined as $T(\langle s,\theta \rangle, a, \langle s', \theta' \rangle) = \theta_a^{s,s'} \delta_\theta(\theta')$ and the observation probability is defined as $Z(\langle s', \theta' \rangle, a, o) = \Pr(o|s',a) = \delta_{s'}(o)$. In fact, real-time search algorithms in the above can be viewed as extensions of POMDP planning algorithms to the hybrid-state POMDP.

Real-Time Heuristic Search

Anytime Error Minimizing Search (AEMS2) (Ross et al. 2008) is one of the successful online POMDP solvers. The algorithm performs the best-first search using the heuristic based on the error bound of the value function for a given amount of time. Upon timeout, the algorithm selects and executes the best action at the root node of the search tree. The search process is essentially a real-time version of the AO* search algorithm (Nilsson 1982). Specifically, the search tree is represented as an AND-OR tree, in which AND-nodes correspond to belief transitions and OR-nodes correspond to action selections. AEMS2 maintains the upper and lower bounds on the optimal value at each node and use them for the search heuristic. In each iteration, a fringe node of the tree is chosen by navigating down the search tree using the following heuristic: the action with the maximum upper bound value at the OR-node and the belief with the maximum error contribution to the root node at the AND-node. One of the strengths of AEMS2 is its completeness and ϵ-optimality of the search (Ross, Pineau, and Chaib-draa 2007): the gap between the upper and lower bounds asymptotically converges to 0 as the size of the search tree grows larger.

Given that BRL can be seen as a BAMDP (i.e. a hybrid-state POMDP) planning problem, AEMS2 can be readily extended to BRL. In fact, various online search-based POMDP solvers (Ross et al. 2008) can be readily extended to BRL. As an example, it is interesting to note that BOP (Fonteneau, Busoniu, and Munos 2013) is coincidentally a special case of real-time AO* on the hybrid-state POMDP with naive initial bounds, i.e. constant upper and lower bounds $\frac{R_{\max}}{1-\gamma}$ and $\frac{R_{\min}}{1-\gamma}$.

Potential-based Shaping

One of the most challenging aspects in RL is the sparsity and delay in reward signals. Suppose that the agent has to navigate in a large environment to reach the goal location. If the reward is zero everywhere except at the goal, the initial exploration of the agent would be nothing better than a random walk with a very small chance of reaching the goal. On the other hand, if we change the reward function so that we give a small positive reward whenever the agent makes a progress towards the goal, the initial exploration can be made very efficient.

Potential-based shaping (Ng, Harada, and Russell 1999) introduces additive bonus (or penalty) to rewards to make

the learning potentially more efficient, while guaranteeing invariance in the optimal behavior. This is achieved by defining a potential function on states $\Phi(s)$ and transforming rewards to

$$R_\Phi(s,a,s') = R(s,a,s') + F_\Phi(s,s'),$$

where $F_\Phi(s,s') = \gamma\Phi(s') - \Phi(s)$ is the shaping function. If the rewards are set to R_Φ instead of R, the optimal value functions satisfy

$$\begin{aligned} V^*_\Phi(s) &= V^*(s) - \Phi(s) \\ Q^*_\Phi(s,a) &= Q^*(s,a) - \Phi(s) \end{aligned} \tag{3}$$

which can be shown using Eq. (1) and Eq. (2). The second equation above gives invariance in the optimal policy.

In the ideal case where we set $\Phi(s) = V^*(s)$, the agent that myopically acts solely based on the immediate shaped reward follows an optimal policy, since

$$\begin{aligned} &\operatorname{argmax}_a \textstyle\sum_{s'} T(s,a,s') R_\Phi(s,a,s') \\ &= \operatorname{argmax}_a \textstyle\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s') - V^*(s)] \\ &= \operatorname{argmax}_a \textstyle\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \\ &= \operatorname{argmax}_a Q^*(s,a) = \pi^*(s). \end{aligned}$$

In a recent work by (Eck et al. 2013), shaping was shown to produce more efficient solutions for POMDP planning using potential functions that encode domain-specific knowledge. It was also used with classical RL algorithms such as SARSA and RMAX, showing promising results (Asmuth, Littman, and Zinkov 2008; Grześ and Kudenko 2010).

Heuristic Search with Shaping for BRL

Shaping for BAMDP

Although AEMS2 and other online POMDP solvers naturally extend to BAMDP, one of the main factors that critically impact the performance is the sparsity and delay in rewards. Combined with the uncertainty in the transition model, we end up with a very loose initial bound of the value function. Ideally, we would like to have a fast yet tight bound of the Bayes-optimal value function, but they are hard to obtain, just as in POMDPs. We address this issue by shaping rewards to focus the search effort on more promising outcomes using inferred knowledge about the true underlying environment.

We can naturally extend shaping in MDPs to BAMDPs by noticing that BAMDPs are essentially MDPs with beliefs as states. Specifically, define a nonnegative potential function for reward shaping to be of the form $\Phi : S \times B \to [0, +\infty)$ where S is the set of environment states and B is the set of all possible beliefs. Hence, rewards are shaped by

$$\begin{aligned} &R_\Phi(\langle s,b\rangle, a, \langle s', b_a^{s,s'}\rangle) \\ &\quad = R(\langle s,b\rangle, a, \langle s', b_a^{s,s'}\rangle) + F_\Phi(\langle s,b\rangle, \langle s', b_a^{s,s'}\rangle) \\ &\quad = R(s,a,s') + F_\Phi(\langle s,b\rangle, \langle s', b_a^{s,s'}\rangle), \end{aligned}$$

where $F_\Phi(\langle s,b\rangle, \langle s', b_a^{s,s'}\rangle) = \gamma\Phi(s', b_a^{s,s'}) - \Phi(s,b)$, and we shall use R_Φ as the reward function.

The resulting real-time heuristic search algorithm mostly follows the structure of an online POMDP solver. For the sake of comprehensiveness, we provide the pseudo-code of AEMS2 extended to BAMDPs. Algorithm 1 is the main loop that controls the search in each timestep. In each iteration of the search, one of the fringe nodes is chosen and expanded in the best-first manner. The objective of the expansion is to close the gap in the bounds, and hence we select the fringe node that is likely to maximally reduce the gap at the root node if expanded.

Algorithm 1 The real-time heuristic search BRL algorithm

Input: $\langle s_0, b_0\rangle$: initial state s_0 and model prior b_0
Static: $\langle s, b\rangle$: the current state of the agent
$\mathcal{T}$: the current AND-OR search tree

```
⟨s, b⟩ ← ⟨s_0, b_0⟩
Initialize T to single root node ⟨s, b⟩
while not ExecutionTerminated() do
    while not SearchTimedOut() do
        ⟨s*, b*⟩ ← ChooseNextNodeToExpand()
        Expand(⟨s*, b*⟩)
        UpdateAncestor(⟨s*, b*⟩)
    end while
    Execute best action a* for ⟨s, b⟩
    Observe new state s'
    Update tree T so that ⟨s', b_{a*}^{s,s'}⟩ is the new root
end while
```

Specifically, we consider fringe nodes that are reachable by actions with the maximum upper bound value at each intermediate node

$$\Pr(a|\langle s,b\rangle) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a' \in A} U_\mathcal{T}(\langle s,b\rangle, a') \\ 0 & \text{otherwise} \end{cases}$$

and compute the probability of reaching a fringe node $\langle s_d, b_d\rangle$ at depth d by taking the path h from the root node $\langle s_0, b_0\rangle$ to the fringe node, i.e. $h(\langle s_d, b_d\rangle) = \langle s_0, b_0\rangle, a_0, \langle s_1, b_1\rangle, a_1, \ldots, a_{d-1}, \langle s_d, b_d\rangle$,

$$\begin{aligned} \Pr(h(\langle s_d, b_d\rangle)) &= \prod_{i=0}^{d-1} \Pr(s_{i+1}|\langle s_i, b_i\rangle, a_i) \Pr(a_i|\langle s_i, b_i\rangle) \\ &= \prod_{i=0}^{d-1} T_{b_i}(s_i, a_i, s_{i+1}) \Pr(a_i|\langle s_i, b_i\rangle). \end{aligned}$$

Using the probability of path h, we compute the error contribution of each fringe node $\langle s_d, b_d\rangle$ on the root node

$$e(\langle s_d, b_d\rangle) = \gamma^d \Pr(h(\langle s_d, b_d\rangle))[U_\mathcal{T}(s_d, b_d) - L_\mathcal{T}(s_d, b_d)]$$

and choose the fringe node with the maximum error contribution. The best fringe node can be identified efficiently without exhaustive enumeration in each iteration.

We then expand the chosen fringe node and update its lower and upper bounds using Algorithm 2. These updated bounds are then propagated up to the root node using Algorithm 3. Note that we use shaped reward R_Φ throughout the algorithm, instead of the actual reward R.

We remark that the bound initialization is a subtle but a crucial step for search. Given the initial upper and lower

Algorithm 2 Expand($\langle s, b\rangle$)

Input: $\langle s, b\rangle$: an OR-Node chosen to expand
Static: U : an upper bound on V^*
L : a lower bound on V^*
$\mathcal{T}$: the current AND-OR search tree

for $a \in A$ **do**
  for $s' \in S$ **do**
    Create child node $\langle s', b_a^{s,s'}\rangle$
    $U_{\mathcal{T}}(s', b_a^{s,s'}) \leftarrow U^0_\Phi(s', b_a^{s,s'})$
    $L_{\mathcal{T}}(s', b_a^{s,s'}) \leftarrow L^0_\Phi(s', b_a^{s,s'})$
  end for
  $U_{\mathcal{T}}(\langle s, b\rangle, a) \leftarrow \sum_{s' \in S} T_b(s, a, s')$
    $[R_\Phi(\langle s, b\rangle, a, \langle s', b_a^{s,s'}\rangle) + \gamma U_{\mathcal{T}}(s', b_a^{s,s'})]$
  $L_{\mathcal{T}}(\langle s, b\rangle, a) \leftarrow \sum_{s' \in S} T_b(s, a, s')$
    $[R_\Phi(\langle s, b\rangle, a, \langle s', b_a^{s,s'}\rangle) + \gamma L_{\mathcal{T}}(s', b_a^{s,s'})]$
end for
$U_{\mathcal{T}}(s, b) \leftarrow \min\left(U_{\mathcal{T}}(s, b), \max_a U_{\mathcal{T}}(\langle s, b\rangle, a)\right)$
$L_{\mathcal{T}}(s, b) \leftarrow \max\left(L_{\mathcal{T}}(s, b), \max_a L_{\mathcal{T}}(\langle s, b\rangle, a)\right)$

Algorithm 3 UpdateAncestor($\langle s', b'\rangle$)

Input: $\langle s', b'\rangle$: an OR-Node chosen to update its ancestors
Static: U : an upper bound
L : a lower bound
$\mathcal{T}$: the current AND-OR search tree

while $\langle s', b'\rangle$ is not root of $\mathcal{T}$ **do**
  Set $\langle s, b\rangle$ to be the parent of $\langle s', b'\rangle$ and a to be the corresponding action
  $U_{\mathcal{T}}(\langle s, b\rangle, a) \leftarrow \sum_{s' \in S} T_b(s, a, s')$
    $[R_\Phi(\langle s, b\rangle, a, \langle s', b_a^{s,s'}\rangle) + \gamma U_{\mathcal{T}}(s', b_a^{s,s'})]$
  $L_{\mathcal{T}}(\langle s, b\rangle, a) \leftarrow \sum_{s' \in S} T_b(s, a, s')$
    $[R_\Phi(\langle s, b\rangle, a, \langle s', b_a^{s,s'}\rangle) + \gamma L_{\mathcal{T}}(s', b_a^{s,s'})]$
  $U_{\mathcal{T}}(s, b) \leftarrow \min\left(U_{\mathcal{T}}(s, b), \max_a U_{\mathcal{T}}(\langle s, b\rangle, a)\right)$
  $L_{\mathcal{T}}(s, b) \leftarrow \max\left(L_{\mathcal{T}}(s, b), \max_a L_{\mathcal{T}}(\langle s, b\rangle, a)\right)$
  $\langle s', b'\rangle \leftarrow \langle s, b\rangle$
end while

bounds $U^0(s, b)$ and $L^0(s, b)$ using the original reward function, it may seem natural to use initial bounds $U^0_\Phi(s, b) = U^0(s, b) - \Phi(s, b)$ and $L^0_\Phi(s, b) = L^0(s, b) - \Phi(s, b)$, based on the relationship in Eq. (3). However, we can show that this is not a good idea:

Theorem 1. *If the bounds using the shaped reward are initialized $U^0_\Phi(s, b) = U^0(s, b) - \Phi(s, b)$ and $L^0_\Phi(s, b) = L^0(s, b) - \Phi(s, b)$, the algorithm will expand the same fringe node as using the original reward.*

Proof. Given the path h from the root node to the fringe node $\langle s, b\rangle$ at depth d, the error contribution of the fringe node under the shaped rewards is

$$\begin{aligned} e_\Phi(\langle s, b\rangle) &= \gamma^d \Pr(h(\langle s, b\rangle))[U^0_\Phi(s, b) - L^0_\Phi(s, b)] \\ &= \gamma^d \Pr(h(\langle s, b\rangle))[U^0(s, b) - L^0(s, b)], \end{aligned}$$

since $\Phi(s, b)$ cancels out. In addition, the upper bound value at its parent node $\langle s'', b''\rangle$ under the shaped rewards is updated by

$$U_{\mathcal{T}}(\langle s'', b''\rangle, a) = \textstyle\sum_{s'} T_{b''}(s'', a, s')\mathcal{U}_\Phi(s'', b'', a, s'),$$

where

$$\begin{aligned} &\mathcal{U}_\Phi(s'', b'', a, s') \\ &\quad = [R_\Phi(\langle s'', b''\rangle, a, \langle s', (b'')_a^{s'',s'}\rangle) + \gamma U_{\mathcal{T}}(s', (b'')_a^{s'',s'})] \\ &\quad = [R(s'', a, s') + \gamma U^0(s', (b'')_a^{s'',s'}) - \Phi(s'', b'')], \end{aligned}$$

which makes the maximum upper bound action $\operatorname{argmax}_a U_{\mathcal{T}}(\langle s'', b''\rangle, a)$ at the parent node unchanged compared to the one using the original rewards, since $\Phi(s'', b'')$ is invariant over actions. By induction, the maximum upper bound actions at all intermediate nodes on the path h do not change, which makes $\Pr(h(\langle s, b\rangle))$ unchanged. Thus, we have $e_\Phi(\langle s, b\rangle) = e(\langle s, b\rangle)$, which implies that the algorithm will expand the same fringe node as with the original rewards. □

This theorem states that translating both bounds by $-\Phi$ essentially nullifies the effect of shaping, making the algorithm build the exact same search tree as with the original reward. In our implementation, we used

$$\begin{aligned} U^0_\Phi(s, b) &= U^0(s, b) - \Phi_{\min} \\ L^0_\Phi(s, b) &= L^0(s, b) - \Phi(s, b), \end{aligned}$$

where $\Phi_{\min} = \min_{s,b} \Phi(s, b)$. This is to make the reward shaping affect the construction of search tree (since the AEMS2 heuristic chooses the action with the maximum upper bound value), while not affecting the final choice of the action for execution (since the algorithm chooses the action with the maximum lower bound value). We can show the latter by the following corollary:

Corollary 1. *Given a search tree $\mathcal{T}$, the best lower bound action computed from $\mathcal{T}$ using the shaped rewards is identical to the one using the original rewards.*

Proof. In the proof of Theorem 1, simply replace the upper bound U by the lower bound L. □

As a final remark, since we have not modified the search heuristic, the completeness and ϵ-optimality guarantee of the search in (Ross, Pineau, and Chaib-draa 2007) is preserved.

Potential Functions

It would seem that we need to define potential function a-priori. However, due to the nature of real-time search algorithms, the number of belief states at which the potential function is evaluated is bounded by the number of nodes expanded in the search tree. Hence, we can perform evaluations on-the-fly, even incorporating experiences observed from the environment. One thing that we should keep in mind is that if we decide to use some value for the potential function at $\langle s, b\rangle$, we should use the same value when the search later creates a node with the same $\langle s, b\rangle$. This is to have a consistent specification of the potential function.

Value Functions of MDP Samples An ideal potential function would be the optimal value function of the actual underlying environment model. However, since this is not available, we could use a set of models sampled from the current belief. In fact, a number of BRL algorithms took the sampling approach albeit for different purposes: MC-BRL (Wang et al. 2012) and BA-POMDP (Ross, Chaib-draa, and Pineau 2007) used the set of sampled models for approximate belief tracking (i.e. particle filtering), and BOSS (Asmuth et al. 2009) used the samples for computing a probabilistic upper bound of the Bayes-optimal value.

In our approach, we periodically sample a set of K MDPs (i.e. transition probabilities) $\{\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_K\}$ from the current belief b and solve each MDP to obtain optimal value functions $\{V^*_{\hat{\theta}_1}, V^*_{\hat{\theta}_2}, \ldots, V^*_{\hat{\theta}_K}\}$. The potential function is defined as

$$\Phi_{\text{KMDP}}(s, b) = \sum_{k=1}^{K} w_k(b) V^*_{\hat{\theta}_k}(s),$$

where the weight w_k is set to the posterior probability $\Pr(k|b)$ of the k-th MDP. The weights are initially set to $1/K$ and updated by the Bayes rule upon transition $\langle s, a, s' \rangle$

$$w_k \leftarrow \eta \cdot w_k \cdot \hat{\theta}_k^{s,a,s'} \quad \forall k,$$

where η is the normalization constant. The MDP resampling period controls the tradeoff between reducing the overall running time and obtaining a more accurate value function estimate. In addition, since solving MDPs can take a considerable amount of time, we perform resampling only at the root node and reuse them in node expansions, as in sample-based tree search algorithms (Silver and Veness 2010).

Note that the MDP resampling and the weight update lead to the change in the potential function. In order to make the optimal policy invariant, we need the potential function to be consistent by making the same $\langle s, b \rangle$ yield the same potential value. This is achieved by a hashtable so that the updated potential function is only evaluated for novel state-belief pairs that are encountered during search.

Value Function of Optimistic MDP Inspired by PAC-BAMDP algorithms, we can periodically build an optimistic MDP approximation of the BAMDP based on the current belief, and use its optimal value function as the potential function. As an example, we can use the optimistic MDP constructed in BEB (Kolter and Ng 2009), of which the optimal value function is defined by

$$V^*_{\text{BEB}}(s, b) = \max_a \sum_{s'} T_b(s, a, s') \cdot \left[R(s, a, s') + \frac{\beta}{1 + n_b(s, a)} + \gamma V^*_{\text{BEB}}(s', b) \right],$$

where $\beta \geq 0$ is the exploration bonus parameter, and $n_b(s, a)$ is the number of visits to the state-action pair $\langle s, a \rangle$ in the current belief b. The value function V^*_{BEB} can be efficiently computed by the standard value iteration. We denote this potential function as Φ_{BEB}. Our method can be seen as the multi-step lookahead search extension to BEB.

As in Φ_{KMDP}, we recompute Φ_{BEB} only at the root node at regular intervals, not necessarily at every timestep. We also use a hashtable so that the updated potential function is only evaluated for novel state-belief pairs.

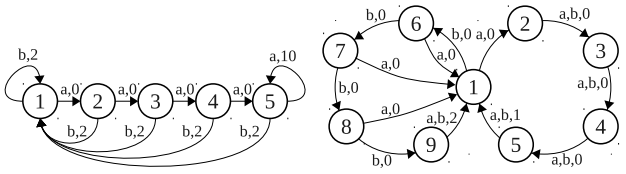

Figure 1: (a) CHAIN (left) and (b) DOUBLE-LOOP (right)

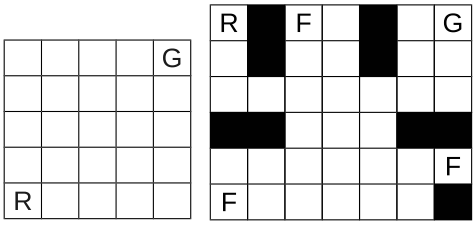

Figure 2: (a) GRID5 (left) and (b) MAZE (right)

Experiments

We conducted experiments on the following five benchmark BRL domains:

- CHAIN (Strens 2000) consists of a linear chain of 5 states and 2 actions $\{a, b\}$, as shown in Figure 1 (a). The rewards are shown as edge labels for each transition. The transitions are stochastic: the agent "slips" and perform the other action with probability 0.2.
- DOUBLE-LOOP (Dearden, Friedman, and Russell 1998) consists of 9 states and 2 actions, as shown in Figure 1 (b). The transitions are deterministic. optimal behavior is to complete the traversal of the left loop with a reward of 2 by executing action b all the time, while the right loop is easier to complete yielding a reward of 1.
- GRID5 (Guez, Silver, and Dayan 2012) consists of 5×5 states with no reward anywhere except at the goal location (G) which is at the opposite to the reset location (R) (Figure 2 (a)). Once the agent reaches G, it is sent back to R with a reward of 1. There are 4 actions for moving in each cardinal direction, of which the transitions are stochastic: the agent moves in random directions with probability 0.2.
- GRID10 (Guez, Silver, and Dayan 2012) is a larger version of GRID5 with 10×10 states.
- MAZE (Dearden, Friedman, and Russell 1998) consists of 264 states and 4 actions, where the agent has to collect flags at certain locations (F) and arrive at the goal location (G), as shown in Figure 2 (b). Once the agent reaches G, it is sent back to the reset location (R) with the reward equal to the number of flags (F) collected. The stochasticity in transition is same as GRID5.

In Table 1, we compare the total undiscounted rewards gathered from the following 5 algorithms: BAMCP[1] (Guez,

[1] https://github.com/acguez/bamcp

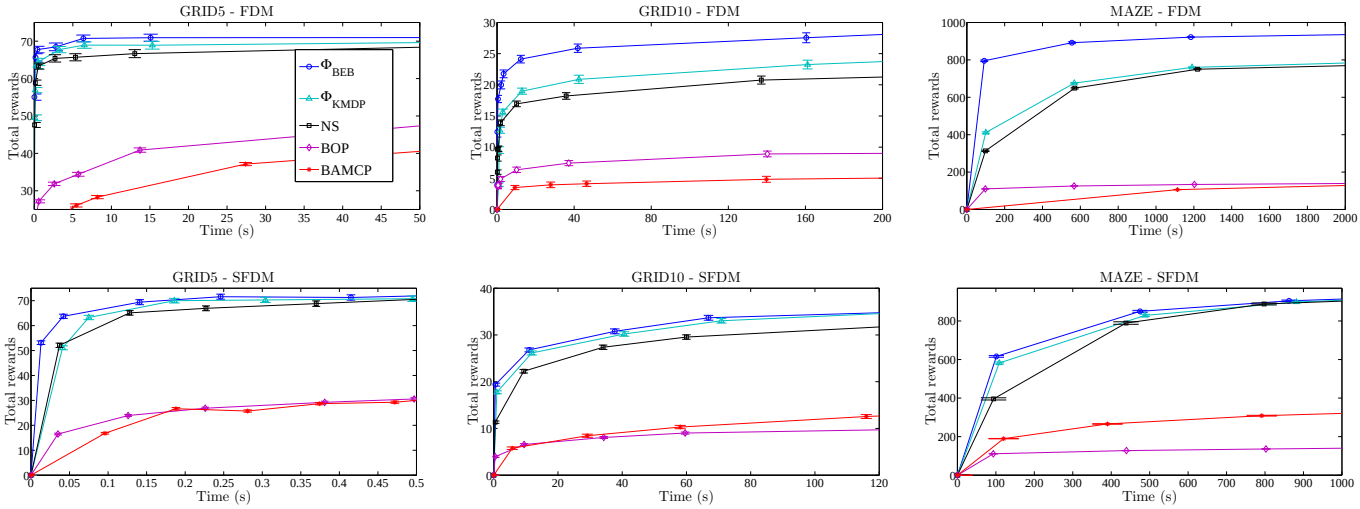

Figure 3: Total rewards vs. search CPU times in larger domains (GRID5, GRID10, and MAZE)

Methods		CHAIN	DOUBLE-LOOP	GRID5	GRID10	MAZE
Φ_{BEB}	FDM	**2556.16** ($\pm$75.66)	299.59 ($\pm$8.52)	**71.88** ($\pm$0.96)	**28.19** ($\pm$0.51)	**936.80** ($\pm$1.53)
Φ_{KMDP}	FDM	**2583.86** ($\pm$73.62)	305.27 ($\pm$8.34)	**71.29** ($\pm$0.97)	24.47 ($\pm$0.59)	786.48 ($\pm$2.54)
NS	FDM	**2515.71** ($\pm$75.40)	297.47 ($\pm$8.58)	**69.78** ($\pm$1.02)	21.40 ($\pm$0.50)	772.24 ($\pm$2.61)
BOP	FDM	**2540.52** ($\pm$76.56)	286.47 ($\pm$8.54)	54.72 ($\pm$0.82)	9.04 ($\pm$0.28)	139.46 ($\pm$0.87)
BAMCP	FDM	1849.72 ($\pm$25.10)	**387.17** ($\pm$1.17)	48.65 ($\pm$0.50)	5.14 ($\pm$0.41)	133.59 ($\pm$0.84)
Φ_{BEB}	SFDM	**2924.03** ($\pm$74.53)	248.79 ($\pm$7.53)	**73.05** ($\pm$1.08)	**36.33** ($\pm$0.60)	**966.73** ($\pm$4.02)
Φ_{KMDP}	SFDM	**2942.92** ($\pm$72.11)	250.16 ($\pm$7.58)	**72.42** ($\pm$1.20)	**36.11** ($\pm$0.59)	**961.74** ($\pm$4.78)
NS	SFDM	**2961.97** ($\pm$72.23)	247.19 ($\pm$7.45)	**72.27** ($\pm$1.17)	33.92 ($\pm$0.55)	951.61 ($\pm$4.12)
BOP	SFDM	**2985.16** ($\pm$72.45)	248.71 ($\pm$7.52)	65.15 ($\pm$1.13)	10.36 ($\pm$0.25)	151.57 ($\pm$0.86)
BAMCP	SFDM	1934.57 ($\pm$27.14)	**282.70** ($\pm$8.10)	70.97 ($\pm$0.79)	16.72 ($\pm$0.42)	377.09 ($\pm$3.44)

Table 1: The averages of total undiscounted rewards and their 95% confidence intervals. For all domains except GRID10 and MAZE, the results are from 500 runs of 1000 timesteps. For GRID10 and MAZE, we used 2000 timesteps and 20000 timesteps, respectively. We set $\gamma = 0.95$ for all domains. Top performance results are highlighted in bold face.

Silver, and Dayan 2012) is one of the most efficient algorithms that uses Monte-Carlo Tree Search; BOP (Fonteneau, Busoniu, and Munos 2013) is a real-time heuristic search algorithm that uses the naive upper and lower bounds $\frac{R_{\max}}{1-\gamma}$ and $\frac{R_{\min}}{1-\gamma}$; NS is the real-time heuristic search algorithm presented in the previous section without shaping (No Shaping), but using more sophisticated bounds calculated by optimistic and pessimistic value iteration (Givan, Leach, and Dean 2000); Φ_{KMDP} and Φ_{BEB} are the same search algorithms using the corresponding potential functions for shaping. In addition, we experimented with two different priors: flat Dirichlet multinomial (FDM) with $\alpha_0 = 1/|S|$ (Guez, Silver, and Dayan 2012) and sparse factored Dirichlet multinomial (SFDM) (Friedman and Singer 1999).

Each algorithm was given the CPU time of 0.1s per timestep by adjusting the number of node expansions. This time limit was sufficient for all the algorithms to reach their highest levels of performance, except in larger domains GRID10 and MAZE. The parameter settings for each algorithm were as follows: for BAMCP, we followed the exact settings in (Guez, Silver, and Dayan 2012), which were $c = 3$ and $\epsilon = 0.5$ for the exploration constants in the tree search and the rollout simulation, and the maximum depth of the search tree was set to 15 in all domains except GRID10 and MAZE, in which the depth was increased to 50; for Φ_{KMDP}, we set the number of MDP samples $K = 10$; for Φ_{BEB}, β was chosen from $\{0.5, 1, 10, 20, 30, 50\}$ that performed the best; the recomputation of the potential function was set to happen 10 times during a run. In addition, upon noticing that the algorithms with shaping performed well even without the hashtable, we decided removed the bookkeeping in the experiments for further speedup. In fact, the hit rate of the cache was less than 10% in all experiments.

Real-time heuristic search with reward shaping yielded the best results in all domains except DOUBLE-LOOP, and showed significant improvement in learning performance on larger domains such as GRID5, GRID10, and MAZE. In CHAIN, which was the smallest domain, shaping had almost no effect on search since good actions could be readily found with small search trees. Finally, Figure 3 shows the improvement in the total reward as we increase the search time for three larger domains. It clearly shows the effectiveness of shaping for real-time heuristic search.

It is interesting to note the singularity in the DOUBLE-LOOP results. BAMCP performed far better than other al-

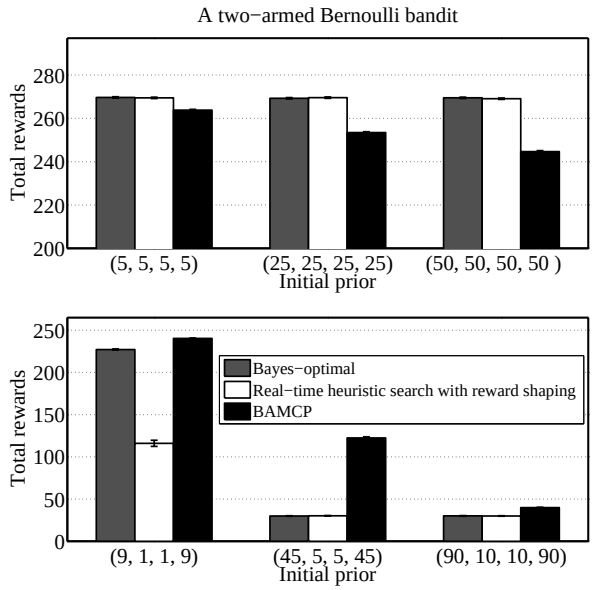

Figure 4: Performance comparison of real-time heuristic search with reward shaping and BAMCP against the Bayes-optimal policy on a two-armed Bernoulli bandit $\mu_1 = 0.1$ and $\mu_2 = 0.9$ with $\gamma = 0.99$.

gorithms in this particular domain. In order to further analyze the results, we conducted another set of experiments on a bandit problem where we can obtain Bayes-optimal policies with a high accuracy via computing Gittins indices (Gittins 1979). In this experiment, we consider a two-armed Bernoulli bandit, where the two arms have 0.1 and 0.9 success probabilities. In Figure 4, we compare the total rewards obtained by the Bayes-optimal policy, our real-time heuristic search algorithm with reward shaping, and BAMCP. The averages were obtained from 1000 runs of 300 time steps. Again, as for BAMCP, the exploration constant c was chosen from $\{0.5, 1, 1.5, 2, 2.5, 3\}$ that performed the best.

The top graph in Figure 4 shows the results with three different initial priors, $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (5, 5, 5, 5), (25, 25, 25, 25)$, and $(50, 50, 50, 50)$. Note that while our search algorithm performed very close to the Bayes-optimal policy, BAMCP was quite susceptible to the prior and was not able to overcome the strong incorrect prior. The bottom graph shows the same comparison with a different set of incorrect initial priors: $(9, 1, 1, 9), (45, 5, 5, 45)$, and $(90, 10, 10, 90)$. In this case, the priors had less effect on the BAMCP performance, and in fact BAMCP was performing better than Bayes-optimal policy. Our search algorithm on the other hand, matched the performance of the Bayes-optimal policy in two out of three cases.

The reason why we obtained these result is because the behavior of BAMCP arises from the combination of two parameters: the prior and the exploration constant. Hence, by changing the exploration constant, the prior can be ignored and make the algorithm be tuned to the specific problem at hand. We believe that this is what is happening behind the DOUBLE-LOOP experiments.

Conclusion and Future Work

In this paper, we presented shaping for significantly improving the learning performance of a model-based BRL method. Our main insight comes from the BAMDP formulation of BRL, which is a hybrid-state POMDP. We showed how shaping can be used for real-time AO* search as an efficient BRL method.

Shaping mitigates the sparsity and delay of rewards, helping the search algorithm to find good actions without the necessity to build large search trees for long-horizon planning. We proposed two approaches to defining the potential function for shaping, which do not depend on a-priori knowledge about the true underlying environment - they only leverage the structural regularity in the POMDP that arises from BAMDP. They are also adaptive in the sense that they use past experiences from the underlying model to estimate the Bayes-optimal value.

Extending our approach to larger or continuous state spaces, and integrating shaping with Monte-Carlo Tree Search algorithms are promising directions for the future work.

Acknowledgments

This work was partly supported by the ICT R&D program of MSIP/IITP [14-824-09-014, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)], National Research Foundation of Korea (Grant# 2012-007881), and Defense Acquisition Program Administration and Agency for Defense Development under the contract UD140022PD, Korea

References

Araya-López, M.; Thomas, V.; and Buffet, O. 2012. Near-optimal BRL using optimistic local transition. In *Proceedings of the 29th International Conference on Machine Learning*, 97–104.

Asmuth, J., and Littman, M. 2011. Learning is planning: Near Bayes-optimal reinforcement learning via Monte-Carlo tree search. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 19–26.

Asmuth, J.; Li, L.; Littman, M. L.; Nouri, A.; and Wingate, D. 2009. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 19–26.

Asmuth, J.; Littman, M. L.; and Zinkov, R. 2008. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*.

Castro, P. S., and Precup, D. 2010. Smarter sampling in model-based Bayesian reinforcement learning. In *Machine Learning and Knowldege Discovery in Database*. Springer. 200–214.

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

Duff, M. O. 2002. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. Ph.D. Dissertation, University of Massachusetts Amherst.

Eck, A.; Soh, L.-K.; Devlin, S.; and Kudenko, D. 2013. Potential-based reward shaping for POMDPs. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, 1123–1124.

Fonteneau, R.; Busoniu, L.; and Munos, R. 2013. Optimistic planning for belief-augmented Markov decision processes. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 77–84.

Friedman, N., and Singer, Y. 1999. Efficient Bayesian parameter estimation in large discrete domains. In *Advances in Neural Information Processing Systems*, 417–423.

Gittins, J. C. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)* 41(2):pp. 148–177.

Givan, R.; Leach, S.; and Dean, T. 2000. Bounded-parameter Markov decision processes. *Artificial Intelligence* 122:71–109.

Grześ, M., and Kudenko, D. 2010. Online learning of shaping rewards in reinforcement learning. *Neural Networks*.

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

Kolter, J. Z., and Ng, A. Y. 2009. Near-Bayesian exploration in polynomial time. In *Proceedings of the 26th International Conference on Machine Learning*, 513–520.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of 16th International Conference on Machine Learning*, 278–287.

Nilsson, N. J. 1982. *Principles of Artificial Intelligence*. Symbolic Computation / Aritificial Intelligence. Springer.

Poupart, P.; Vlassis, N.; Hoey, J.; and Regan, K. 2006. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 697–704.

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

Ross, S.; Chaib-draa, B.; and Pineau, J. 2007. Bayes-adaptive POMDPs. In *Advances in Neural Information Processing Systems*, 1225–1232.

Ross, S.; Pineau, J.; and Chaib-draa, B. 2007. Theoretical analysis of heuristic search methods for online POMDPs. In *Advances in Neural Information Processing Systems*, 1216–1225.

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

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

Wang, Y.; Won, K. S.; Hsu, D.; and Lee, W. S. 2012. Monte Carlo Bayesian reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*, 1135–1142.