

# *Approximate planning for bayesian hierarchical reinforcement learning*

**Ngo Anh Vien, Hung Ngo, Sungyoung Lee & TaeChoong Chung**

## **Applied Intelligence**

The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies

ISSN 0924-669X

Appl Intell

DOI 10.1007/s10489-014-0565-6

Volume 41, Number 1, July 2014  
ISSN: 0924-669X



## **APPLIED INTELLIGENCE**

*The International Journal of  
Artificial Intelligence,  
Neural Networks, and  
Complex Problem-Solving Technologies*

**Editor-in-Chief:**

**Moonis Ali**

 Springer

 Springer

**Your article is protected by copyright and all rights are held exclusively by Springer Science +Business Media New York. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your article, please use the accepted manuscript version for posting on your own website. You may further deposit the accepted manuscript version in any repository, provided it is only made publicly available 12 months after official publication or later and provided acknowledgement is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at [link.springer.com](http://link.springer.com)".**

# Approximate planning for bayesian hierarchical reinforcement learning

Ngo Anh Vien · Hung Ngo · Sungyoung Lee ·  
TaeChoong Chung

© Springer Science+Business Media New York 2014

**Abstract** In this paper, we propose to use hierarchical action decomposition to make Bayesian model-based reinforcement learning more efficient and feasible for larger problems. We formulate Bayesian hierarchical reinforcement learning as a partially observable semi-Markov decision process (POSMDP). The main POSMDP task is partitioned into a hierarchy of POSMDP subtasks. Each subtask might consist of only primitive actions or hierarchically call other subtasks' policies, since the policies of lower-level subtasks are considered as macro actions in higher-level subtasks. A solution for this hierarchical action decomposition is to solve lower-level subtasks first, then higher-level ones. Because each formulated POSMDP has a continuous state space, we sample from a prior belief to build an approximate model for them, then solve by using a recently introduced Monte Carlo Value Iteration with Macro-Actions solver. We name this method Monte Carlo Bayesian Hierarchical Reinforcement Learning. Simulation results show that our algorithm exploiting the action hierarchy performs

significantly better than that of flat Bayesian reinforcement learning in terms of both reward, and especially solving time, in at least one order of magnitude.

**Keywords** Reinforcement learning · Bayesian model-based RL · Bayesian reinforcement learning · Model-based reinforcement learning · Partially observable Markov decision process (POMDP) · Partially observable semi-MDP (POSDMP)

## 1 Introduction

Reinforcement learning (RL) [40, 42] is a subfield (This is an extension of a conference paper (Vien, N.A. et al (52))) of Machine Learning dealing with learning from delayed reward and sequential decision making in unknown environments. RL has had some remarkable practical successes in various areas, including learning to play checkers [36], backgammon [43–45], job-scheduling [62], chess [7], dynamic channel allocation [37, 48, 49, 55, 57], robotics [1, 4, 27, 28, 50], and other applications [18, 22, 23, 25, 29, 53, 54]. RL algorithms have to deal with the exploration-exploitation trade-off: they need to balance actions that reduce the uncertainty about the environment, with actions that exploit the learned knowledge. A systematic approach to dealing with this issue is to use Bayesian RL.

Bayesian RL has been intensively studied by many approaches such as model-based [3, 12, 33, 38, 58], model-free [10, 13, 14], and policy search [16, 17, 56]. Of these, Bayesian model-based RL (BRL) is a family of methods which use Bayesian inference to update a posterior distribution of the underlying model of the environment. This model is often in the form of a Markov Decision Process (MDP) [40, 42], a mathematical framework for studying

---

N. A. Vien (✉)  
Machine Learning and Robotics Lab, University of Stuttgart,  
Stuttgart, Germany  
e-mail: vien.ngo@ipvs.uni-stuttgart.de

H. Ngo  
Swiss AI Lab IDSIA, USI-SUPSI, Galleria 2, CH-6928 Manno-  
Lugano, Ticino - Switzerland  
e-mail: hung@idsia.ch

S. Lee · T. Chung (✉)  
Department of Computer Engineering, Kyung Hee University,  
Seocheon-dong, Giheung-gu, Yongin-si, Gyeonggi-do, 446-701,  
South Korea  
e-mail: tcchung@khu.ac.kr

S. Lee  
e-mail: sylee@oslab.khu.ac.kr

RL problems with the Markov property (i.e., memoryless) assumption on the environment dynamics. BRL has been known to efficiently utilize the training data and optimally balance the exploration/exploitation trade-off. There has been another effort to use deterministic approximate inference [39] for reinforcement learning such as the work in [15] using variational Bayes. Though their work provided new insight into BRL, it can only be tractable for very small problems, and often converges to a local optimal policy.

Formally, BRL can be reformulated as a partially observable Markov decision process (POMDP) [12, 33, 61]. This formulation is used in many proposals [9, 19, 33–35, 51, 58, 59]. They use either a conjugate belief distribution to perform inference for an approximate policy, or sampling techniques, e.g. particle filtering, to approximately represent beliefs. Being formulated this way, the posterior distribution over models given the observations, called *belief*, is conveniently represented in closed form as a product of Dirichlet distributions. As a result, the optimal value function in BRL can be represented by a set of multivariate polynomials. Unfortunately, the size of the polynomials grows exponentially with the problem horizon, which makes computing Bayes-optimal behavior intractable, severely limiting the applicability of the method.

Hierarchical action decomposition, which exploits temporal abstraction, has been previously used to accelerate planning and learning in flat MDP [6, 11, 20, 24, 41] and POMDP [21, 26, 31, 46]. It has been shown to effectively improve the performance of traditional RL [2, 6, 41]. However, there has been very little effort in bridging the gap between BRL and temporal abstraction; in fact, there is only one recent work which solves BRL problems with a given action hierarchy, called Bayesian MAXQ [8]. Bayesian MAXQ maintains distributions over primitive actions and pseudo rewards of macro actions [8], then learns a policy online. Even though this method computes a posterior distribution over models, the policy is found using model-free approach. This would not guarantee any Bayesian optimality for exploration/exploitation trade-off. Furthermore, a policy learned online is only optimal with respect to the underlying model at solving time, which is changing over time.

In this paper, we aim to develop an alternative formulation, which not only exploits the hierarchical structure of the problem, but also preserves the Bayesian optimality for the exploration/exploitation trade-off. Besides, we solve for the policy offline, taking into account all possible underlying generative environments. Our formulation, however, will still be used with other online planners (e.i. will still be used with other online planners).

We propose to formulate Bayesian hierarchical reinforcement learning (BHRL) as a partially observable semi-Markov decision process (POSMDP). The main POSMDP

task is partitioned into a hierarchy of smaller subtasks. Each subtask may consist of other subtasks' policies or only primitive actions. The policies of low-level subtasks are considered as macro actions in higher-level subtasks. Therefore, each subtask is again formulated as one POSMDP. To solve this BHRL problem, we develop a Monte Carlo Bayesian Hierarchical Reinforcement Learning (MC-BHRL) algorithm which solves lower-level subtasks first, then higher-level ones. MC-BHRL approximates each subtask's POSMDP by applying a similar idea from [59], which first samples from the prior belief over unknown models, then solves the approximate POSMDP using a recently introduced Macro-MC VI solver [5, 26].

We evaluate MC-BHRL on three domains in both fully and partially observable MDP environments. The simulation results show that MC-BHRL is a promising and effective approach to solving BRL, allowing BRL to be more feasible to larger problems. The computational cost of BRL is reduced significantly when using an action hierarchy in our MC-BHRL framework, lifting the main barrier which has previously limited the applicability of BRL in practical use.

Our main contributions are: 1) A formal formulation of BHRL with an efficient algorithm to solve. This contribution gives new insights into BHRL. 2) The establishment of Bayesian hierarchical optimality for exploration and exploitation trade-off, as in flat BRL. This hierarchical optimality is in the sense that the optimal policy selects an optimal abstract action by using both the current information (i.e., current belief  $b$ —exploration part), and the one-step look-ahead information by observing the outcomes of that abstract action (exploitation part).

The paper is organized as follows. Section 2 reviews background of MDP, semi-MDP (SMDP) formulation for hierarchical RL, POMDP formulation for BRL, and MC-BRL method for BRL. Section 3 introduces POSMDP formulation for BHRL. Section 4 describes a simple and efficient Monte-Carlo method to solve a BHRL problem. Section 5 presents simulation results and analysis for three tasks: Taxi, Cheese-Taxi, and Large Cheese-Taxi. Section 6 concludes the paper.

## 2 Background

We briefly describe the key notions in BRL to set stage for our POSMDP formulation of BHRL in Section 3.

### 2.1 Markov Decision Process

A *discrete* Markov decision process (MDP) is defined by a five tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ , where  $\mathcal{S}$  is a discrete state space,  $\mathcal{A}$  is a discrete action space,  $\mathcal{T}(s, a, s') = P(s'|s, a)$

defines the transition probability to a next state given the current state-action pair,  $\mathcal{R}(s, a, s')$  defines the immediate reward. A control algorithm in RL tries to find an optimal policy  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  with the assumption of unknown environment dynamics, in order to maximize the expected total discounted reward

$$\rho = \mathbb{E} \left( \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s'_t) \right),$$

where  $s_t, a_t, s'_t$ , and  $\gamma \in (0, 1)$  denote a state, an action, a next state, and a discount factor, respectively.

### 2.2 SDMP formulation for hierarchical RL

Semi-Markov decision process (SMDP) framework is an extension of MDP framework by introducing the concept of *options* [41] (or macro actions, composite actions, abstract actions) which is a type of temporal abstraction. An option is considered as a temporally extended action, defined by a three-tuple  $\{\mu, \beta, \mathcal{I}\}$ , where  $\mu$  is an option's policy,  $\beta$  is a termination condition, and  $\mathcal{I} \subseteq \mathcal{S}$  is the set of states where an option initiates. It is proved in [41] that any MDP with a set of defined options is an SMDP defined as a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ , where the state space  $\mathcal{S}$  and action space  $\mathcal{A}$  are primitive or macro actions. The reward function  $\mathcal{R}(s, a) = r_s^a$  and the transit-time function  $\mathcal{T}(s, a, x) = p^a(s, x)$  can be defined by the option's policy  $\mu$  and termination condition  $\beta$  as follows.

The reward of an option  $a$  is

$$r_s^a = E \{ r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{k-1} r_{t+k} | \mu(s_t) = a \},$$

if option  $a$  is taken in state  $s$  at time  $t$ , where  $t + k$  is the termination time of  $a$ . The respective internal transition probability is

$$p^a(s, x) = \sum_{k=1}^{\infty} \gamma^k p(k, x | s, a)$$

where  $p(k, x)$  is the joint probability when taking option  $a$  in state  $s$  and terminating at  $x$  after  $k$  time steps. It is delayed relative to  $\gamma$ .

The Bellman equations for the general policy  $\pi$  are

$$\begin{aligned} V^\pi(s) &= E \left\{ r_{t+1} + \dots + \gamma^{k-1} r_{t+k} + \gamma^k V^\pi(s_{t+k}) \right\} \\ &= \sum_{a \in \mathcal{A}_s} \pi(s, a) \left[ r_s^a + \sum_{s'} p^a(s, s') V^\pi(s') \right] \end{aligned}$$

and the option-value function is

$$\begin{aligned} Q^\pi(s, a) &= E \left\{ r_{t+1} + \dots + \gamma^{k-1} r_{t+k} \right. \\ &\quad \left. + \gamma^k V^\pi(s_{t+k}) | \pi(s_t) = a \right\} \\ &= E \left\{ r_{t+1} + \dots + \gamma^{k-1} r_{t+k} \right. \end{aligned}$$

$$\begin{aligned} &\left. + \gamma^k \sum_{a' \in \mathcal{A}_{s'}} \pi(s, a') Q^\mu(s_{t+k}, a') | \pi(s_t) = a \right\} \\ &= r_s^a + \sum_{s'} p^a(s, s') \sum_{a' \in \mathcal{A}_{s'}} \pi(s', a') Q^\pi(s', a') \end{aligned}$$

It is easy to show that  $Q^\pi(s, a) = V^{a\pi}(s)$ . If the macro actions are decomposed hierarchically, the model  $\{r_s^a, p^a(s, s')\}$  of an option  $a$  can also be re-written as Bellman-like equations

$$\begin{aligned} r_s^a &= \sum_{a' \in \mathcal{A}} \pi(s, a') E \left\{ r + \gamma(1 - \beta(s')) r_{s'}^a \right\} \quad (1) \\ &= \sum_{a' \in \mathcal{A}} \mu(s, a') \left[ r(s, a') + \sum_{s'} p^{a'}(s, s') (1 - \beta(s')) r_{s'}^a \right] \end{aligned}$$

and

$$\begin{aligned} p^a(s, x) &= \sum_{a' \in \mathcal{A}_s} \mu(s, a') \gamma E \left\{ (1 - \beta(s')) p^{a'}(s', x) + \beta(s') \delta_{s'x} \right\} \\ &= \sum_{a' \in \mathcal{A}_s} \mu(s, a') \sum_{s'} p^{a'}(s, s') \left[ (1 - \beta(s')) p^{a'}(s', x) + \beta(s') \delta_{s'x} \right] \quad (2) \end{aligned}$$

### 2.3 POMDP formulation for BRL

A BRL problem can be formulated into a POMDP,  $\mathcal{P} = \langle \mathcal{S}_P, \mathcal{A}, \mathcal{T}_P, \mathcal{R}_P, \mathcal{O}, \mathcal{Z} \rangle$ , whose state space  $\mathcal{S}_P$  consists of the underlying MDP's state space  $\mathcal{S}$  as well as the parameter space modeling the unknown transition dynamics. Specifically, if each unknown transition probability is parameterized by a parameter  $\theta_{s,s'}^a \in [0, 1]$ , then the new state space is  $\mathcal{S}_P = \mathcal{S} \times \left\{ \theta_{s,s'}^a \right\}$ . The action space  $\mathcal{A}$  is kept similarly to that of the original MDP. The transition model  $T_P(s, \theta, a, s', \theta') = \Pr(s', \theta' | s, \theta, a)$  is factored into two parts  $\Pr(s' | s, \theta_{s,s'}^a, a) = \theta_{s,s'}^a$  and  $\Pr(\theta' | \theta) = \delta_{\theta\theta'}$  (the Kronecker delta). The reward function  $R_P(s, \theta, a, s', \theta') = R(s, a, s')$  is as in the original MDP's reward function. The observation space  $\mathcal{O}$  is the observable state space  $\mathcal{S}$  of the underlying MDP [12, 33]. The observation function  $\mathcal{Z}(s', a, o) = \Pr(o | s', a)$  is defined as  $\Pr(o | s', a) = \delta_{s'o}$ . Thus, we obtained a *continuous* state, discrete observation POMDP problem.

Let the prior belief over all unknown parameters  $\theta_a^s$  be  $b(\theta) = \Pr(\theta)$ . Assuming that the prior belief is a product of Dirichlets, then the posterior is also a product of Dirichlets

$$b(\theta) = \prod_{s,a} \mathcal{D}(\theta_a^s; n_a^s), \quad (3)$$

where each unknown distribution  $\theta_a^s$  per one pair  $(s, a)$  is represented by one Dirichlet,  $\mathcal{D}(\theta_a^s; n_a^s) = k \prod_{s'} \theta_{s,a,s'}^{n_{s,a,s'}^s - 1}$ , with  $n_a^s$  a vector of parameters  $\left\{ n_{s,a,s'}^s \right\}$ .

In POMDP formulation,  $o$  is an observation in the state space of the original MDP. Thus we can write the Bellman's update as

$$V_s^{t+1}(b) = \max_a \sum_{s'} \Pr(s'|s, b, a) \left[ R(s, a, s') + \gamma V_{s'}^t(b_a^{s, s'}) \right]. \tag{4}$$

By solving this POMDP we can optimally balance the trade-off between uncertainty in the dynamics and the uncertainty in the unknown model parameter. Thus, optimal trade-off between exploration and exploitation can be achieved [33].

### 2.4 MC-BRL method

MC-BRL builds an approximate  $\hat{\mathcal{P}}$  of a POMDP  $\mathcal{P}$  by sampling directly from a prior belief  $b_0$ . For a brief summary, MC-BRL consists of three steps. First, it samples  $M$  primitive models  $\{\hat{\theta}^0, \dots, \hat{\theta}^{M-1}\}$  from a prior distribution  $b_0(\theta)$  of  $\mathcal{P}$ . Second, it creates a new approximate discrete POMDP  $\hat{\mathcal{P}} = \langle \mathcal{S}_{\hat{\mathcal{P}}}, \mathcal{A}_{\hat{\mathcal{P}}}, \mathcal{O}_{\hat{\mathcal{P}}}, T_{\hat{\mathcal{P}}}, R_{\hat{\mathcal{P}}}, \gamma, b_{\hat{\mathcal{P}}}^0 \rangle$ .  $\hat{\mathcal{P}}$  has state space  $\mathcal{S}_{\hat{\mathcal{P}}} = \mathcal{S} \times \{1, \dots, M\}$ , action space  $\mathcal{A}_{\hat{\mathcal{P}}} = \mathcal{A}$ , observation space  $\mathcal{O}_{\hat{\mathcal{P}}} = \mathcal{S}$  the MDP state space, and reward function  $R_{\hat{\mathcal{P}}}(s, m, a, s', m') = R(s, a, s')$ . The transition and observation functions of  $\hat{\mathcal{P}}$  are

$$T_{\hat{\mathcal{P}}}(s, m, a, m', s', o) = \hat{\theta}_{sas'}^k \delta_{mm'},$$

$$Z_{\hat{\mathcal{P}}}(s', m', a, o) = \delta_{s'o}.$$

Finally, it uses an existing POMDP solver to solve  $\hat{\mathcal{P}}$  for a policy  $\hat{\pi}$ . Since our main focus is on a new formulation for BHRL, which in principle guarantees the Bayesian optimality as in BRL, we assume some general POMDP solver

is given. Specifically, in this paper we use Monte Carlo Bayesian RL algorithm (MC-BRL), which is recently introduced and has been shown to be an efficient and general method for BRL [59].

### 3 POSMDP formulation for BHRL

This section presents our main contribution and gives insight into how BHRL is in principle formulated as a POSMDP. We adopt a similar method of POMDP formulation for BRL, extended to incorporate macro actions. The new formulation, a transformed POMDP consisting of macro actions, is a partially observable semi-Markov decision process (POSMDP) [60].

Consider a hierarchical model-based RL problem with an action set  $\mathcal{A}$  consisting of both primitive and abstract actions. Assume that the problem's action hierarchy defined by a tree is given [11, 24] (e.g., see Fig. 1 and its description in Section 5). Tree nodes are either abstract actions (internal nodes) or primitive actions (leaf nodes). For simplicity, we assume that macro actions are built from primitive actions (as defined in a hierarchy), and the primitive transition function is unknown and parameterized as  $\theta_{ss'}^a \in \Theta$ . The formulation can also be straightforwardly extended when the reward function is unknown; one simple method is to discretize possible reward values as described in [33].

We now formulate BHRL with an extended set of macro actions as a POSMDP, assuming that an action hierarchy and its macro action's termination condition  $\beta$  are given. As shown in [26], a POMDP with macro actions is reformulated as a POSMDP. Therefore, we could formulate BHRL as a hierarchy of multiple sub-POSMDPs, where a higher-level POSMDP consists of macro actions

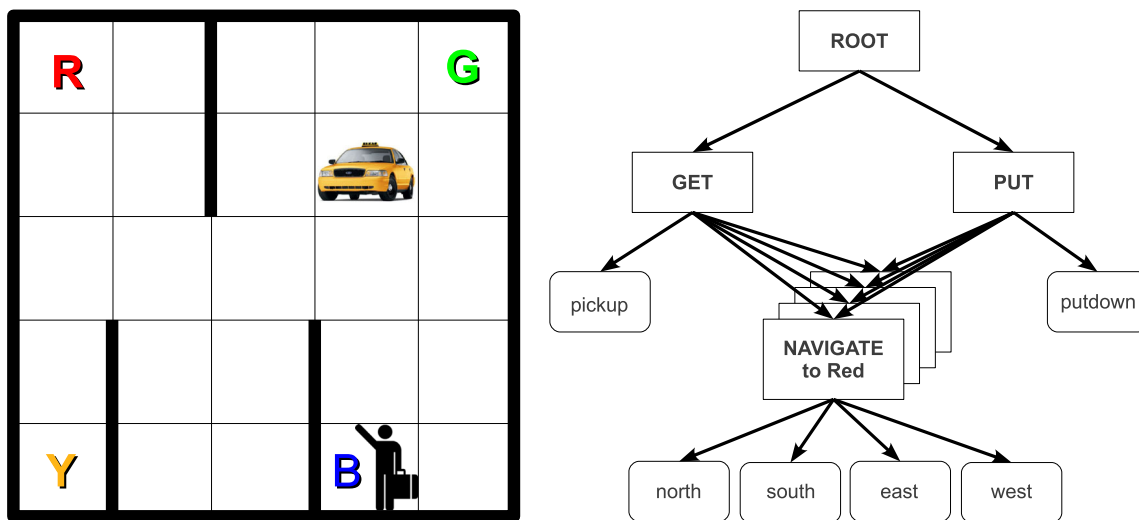


Fig. 1 Taxi problem and an action hierarchy

as policies of lower-level POSMDP. A POSMDP  $\mathcal{P}$  is formally defined as a tuple  $\langle \mathcal{S}_P, \mathcal{A}_P, \mathcal{O}_P, T_P, R_P, \gamma \rangle$ . The state space  $\mathcal{S}_P = \{\mathcal{S}, \Theta\}$  consists of the MDP state space  $\mathcal{S}$  and the unknown dynamics' parameter space  $\Theta$ .  $\mathcal{A}_P$  is an action space consisting of either primitive or macro actions. The observation space  $\mathcal{O}_P = \mathcal{S}$  is the MDP state space.  $T_P$  is a joint distribution function of the state transition, the number of time steps  $k$  taken, and the observation, i.e.,  $T_P(s, \theta, a, \theta', s', o, k) = p(\theta', s', k, o|s, \theta, a)$ . It can be factorized into three probabilities:  $p(s', k|s, \theta, a)$ ,  $p(\theta'|\theta) = \delta_{\theta\theta'}$ , and observation  $p(o|s', \theta', a) = \delta_{s'o}$ . The reward function is  $R_P(s, \theta, a, s', \theta') = R(s, a, s')$ , and  $\gamma$  is a discount factor.

Similar to the POMDP formulation for BRL, the formulated POSMDP has partially observable continuous state, and discrete action and observation spaces. All unknown parameters embedded in the state space are monitored as a belief  $\theta_a^s: b(\theta) = \Pr(\theta)$ . Following [26], we use reweighted beliefs to represent the belief. The belief update,  $b' = \tau(b, a, o)$ , given an abstract action  $a$  and an observation  $o$  (or  $s'$ ), is given by,

$$b'(\theta') = Z_c \sum_{k=1}^{\infty} \gamma^{k-1} \int p(\theta', s', k, o|s, \theta, a) b(\theta) d\theta, \quad (5)$$

where  $Z_c$  is a normalization constant,

$$\begin{aligned} Z_c &= \sum_{k=1}^{\infty} \gamma^{k-1} \int \int p(\theta'', s', k, o|s, \theta, a) b(\theta) d\theta d\theta'' \\ &= \sum_{k=1}^{\infty} \gamma^{k-1} \int \int p(s', k|s, \theta, a) \delta_{\theta''\theta} \delta_{s'o} b(\theta) d\theta d\theta'' \\ &= \sum_{k=1}^{\infty} \gamma^{k-1} \int p(s', k|s, \theta, a) b(\theta) d\theta. \end{aligned}$$

The normalization  $Z_c$  is also interpreted as the discounted probability  $p(o|b, a)$  of observing  $o = s'$ , which is an MDP next state.

A policy  $\pi : \mathcal{B}_P \rightarrow \mathcal{A}_P$  is defined as a mapping from the belief space to the action space. The value function  $V^\pi$  measures the expected discounted reward while following  $\pi$ . The Bellman's equations recursively updating  $V^\pi$  are

$$V_s^\pi(b) = r(b, s, a) + \gamma \sum_{o \in \mathcal{O}_P} p(o|\pi(b), b) V_{s'}^\pi(b'). \quad (6)$$

The optimal policy  $\pi^*$  is defined to have the best value  $V_s^*(b) \geq V_s^\pi(b), \forall \pi, b, s$ , and can be found under the backup operator  $H$ :

$$HV_s(b) = \max_a \left\{ r(b, s, a) + \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b) V_{s'}(b') \right\}. \quad (7)$$

The Bellman equation optimizes the value functions by taking in account not only the current belief  $b$ , but also the next belief  $b'$  computed by observing the outcomes of the macro actions. This guarantees that the optimal policy maximizes the cumulative reward.

We now present some desired properties of the new POSMDP formulation, which is convex and piecewise linear, similar to continuous POMDP [32] and discrete POSMDP [26].

**Lemma 1** *The backup operator is a contraction mapping,*

$$\|HU - HV\|_\infty \leq \gamma \|U - V\|_\infty$$

where  $U$  and  $V$  are two value functions.

*Proof* Assuming that  $\|HU - HV\|$  obtains maximum at belief point  $b$  and state  $s$ , and  $HV_s(b) \leq HU_s(b)$ , then according to the definition of the mapping  $H$  we can write

$$\begin{aligned} \|HU - HV\| &= HU_s(b) - HV_s(b) \\ &= \max_a \{r(b, s, a) + \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)U(b')\} \\ &\quad - \max_{a'} \{r(b, s, a') + \gamma \sum_{o \in \mathcal{O}_P} p(o|a', b)V(b')\} \\ &\leq \max_a \{r(b, s, a) + \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)U_{s'}(b')\} \\ &\quad - \max_a \{r(b, s, a) + \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)V_{s'}(b')\} \\ &\leq \max_a \left\{ \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)U_{s'}(b') \right. \\ &\quad \left. - \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)V_{s'}(b') \right\} \\ &\leq \max_a \left\{ \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)|U_{s'}(b') - V_{s'}(b')| \right\} \\ &\leq \max_a \left\{ \gamma \sum_{o \in \mathcal{O}_P} p(o|a, b)\|U - V\| \right\} \\ &\leq \gamma \|U - V\| \end{aligned}$$

□

The following theorem is the result of the Banach fixed-point theorem and Lemma 1.

**Theorem 1** *The optimal value function  $V^*$  is a single fixed point of the backup operator  $H: V^* = HV^*$ .*

Finally, we provide an important property of the  $t$ -step policy that its value function can be represented by a set of linear functions.

**Theorem 2** *The  $t$ -step optimal value function is convex and piecewise linear, which is represented as*

$$V_t(b) = \sup_{\alpha_t^i \in \Gamma_t} \langle \alpha_t^i, b \rangle$$

where  $\Gamma_t$  is a continuous set of  $\alpha$ -functions  $\alpha_t^i : \mathcal{S}_P \rightarrow \mathfrak{R}$ .

*Proof* We denote an inner product of two continuous functions as

$$\langle f, g \rangle = \int f(x).g(x)dx. \tag{8}$$

We use induction to prove this theorem. Assuming that it holds when  $t = 1$

$$V_1(b) = \max_a \left\{ r(a, b) \right\} = \max_a \langle r_a, b \rangle, \tag{9}$$

where  $r(a, b) = \int r(a, s, \theta)b(\theta)d\theta$  (we have simplified the MDP state  $s$ ). This satisfies the convex and piecewise linear property if we represent one  $\alpha$ -function for each action. The continuous set of these  $\alpha$ -functions is

$$\Gamma_0 = \{r_a\}_{a \in \mathcal{A}}.$$

Assuming that the induction hypothesis holds till  $t - 1$ , thus the value function  $V_{t-1}$  is represented by a continuous set of  $\alpha$ -functions  $\Gamma_{t-1}$  as

$$V_{t-1}(b) = \sup_{\alpha \in \Gamma_{t-1}} \langle \alpha, b \rangle.$$

For the step  $t$  case, we have

$$V_t(b) = \max_a \left\{ r(b, a) + \gamma \sum_{o \in \mathcal{O}} p(o|a, b)V_{t-1}(b') \right\}. \tag{10}$$

By the induction hypothesis and the definition of  $b^{a,o}$  in (5) in the paper, we have

$$\begin{aligned} V_{t-1}(b') &= \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \langle \alpha_{t-1}, b' \rangle = \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \langle \alpha_{t-1}, b^{a,o} \rangle \\ &= \frac{1}{p(o|a, b)} \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \int \alpha_{t-1}(\theta') \sum_{k=1}^{\infty} \gamma^{k-1} \\ &\quad \times \int p(\theta', s', k, o|s, \theta, a)b(\theta)d\theta d\theta'. \end{aligned} \tag{11}$$

Substitute (11) into 10, we obtain

$$\begin{aligned} V_t(b) &= \max_a \left\{ \langle r_a, b \rangle + \gamma \sum_{o \in \mathcal{O}} \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \int \alpha_{t-1}(\theta') \sum_{k=1}^{\infty} \gamma^{k-1} \right. \\ &\quad \left. \times \int p(\theta', s', k, o|s, \theta, a)b(\theta)d\theta d\theta' \right\} \\ &= \max_a \left\{ \langle r_a, b \rangle + \gamma \sum_{o \in \mathcal{O}} \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \int \alpha_{t-1}(\theta') \sum_{k=1}^{\infty} \gamma^{k-1} \right. \\ &\quad \left. \times \int p(\theta', s', k, o|s, \theta, a)b(\theta)d\theta d\theta' \right\} \end{aligned}$$

$$\begin{aligned} &= \max_a \left\{ \langle r_a, b \rangle + \gamma \sum_{o \in \mathcal{O}} \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \int b(\theta) \sum_{k=1}^{\infty} \gamma^{k-1} \right. \\ &\quad \left. \times \int \alpha_{t-1}(\theta')p(\theta', s', k, o|s, \theta, a)d\theta' d\theta \right\} \\ &= \max_a \left\{ \langle r_a, b \rangle + \gamma \sum_{o \in \mathcal{O}} \sup_{\alpha_{t-1} \in \Gamma_{t-1}} \left\langle b, \sum_{k=1}^{\infty} \gamma^{k-1} \right. \right. \\ &\quad \left. \left. \times \int \alpha_{t-1}(\theta')p(\theta', s', k, o|s, \theta, a)d\theta' \right\rangle \right\} \end{aligned}$$

We denote

$$\alpha_{a,o}^j(\theta) = \sum_{k=1}^{\infty} \gamma^{k-1} \int \alpha_{t-1}^j(\theta')p(\theta', s', k, o|s, \theta, a)d\theta',$$

and,

$$\alpha_{a,o,b} = \arg \sup_{\{\alpha_{a,o}^j\}_j} \langle \alpha_{a,o}^j, b \rangle.$$

Therefore, we can form a continuous set of  $\alpha$ -functions at  $t$ -step by

$$\Gamma_t = \bigcup_{\forall b,a \in \mathcal{A}} \left\{ r_a + \gamma \sum_{o \in \mathcal{O}} \alpha_{a,o,b} \right\}.$$

Finally, the  $t$ -step optimal value function  $V_t$  can be represented

$$V_t(b) = \sup_{\alpha \in \Gamma_t} \langle \alpha, b \rangle,$$

in a continuous set of  $\alpha$ -functions. □

#### 4 Monte carlo BHRL

We now describe a recursive method, called Monte Carlo Bayesian Hierarchical Reinforcement Learning (MC-BHRL), for solving the BHRL formulated as POSMDP with a given (i.e., predefined) hierarchical action decomposition  $\mathcal{H}$ . We adopt the similar idea from MC-BRL method [59], to sample from a prior to approximate the exact continuous POSMDP, then use the Macro-MCVI solver [26]. Because Macro-MCVI uses Monte Carlo simulations to approximate the value iteration backup, it ignores the number of steps taken by the macro actions. It only needs to know the primitive actions' estimate models and macro actions' policies to run Monte Carlo simulations. This means we do not need to explicitly model and estimate macro actions' reward functions and transition probabilities like in [8, 30]. Therefore, MC-BHRL needs only to maintain the distribution over the primitive actions' models, as described in Section 2.3.

One may also use the same method from BEETLE algorithm [33] to solve each POSMDP by sampling a set of



reachable beliefs. With such a method, BEETLE failed to find any *good* policies even with a small problem, like the Chain domain, due to the exponentially increasing complexity in the value function's representation. In POSMDP, the sampled belief set is required to be exponentially larger than that of POMDP since terminal time of macro actions is taken into account.

#### 4.1 MC-BHRL algorithm

Each internal node in the task hierarchy  $\mathcal{H}$  is a well-defined POSMDP subtask, which can belong to one of the following categories of formulation:

- Node has only primitive actions as child nodes: This subtask is a standard BRL which can be formulated as one POMDP, a special case of POSMDP.
- Node has both primitive and abstract actions, or only abstract actions: This subtask is a BRL with macro actions which can be formulated as one POSMDP.

The subtasks are solved in bottom-up ordering, as in Algorithm 1. MC-BHRL consists of two stages: The offline stage outputs a policy with respect to uncertainty of all unknown models which have been considered as unobservable factors in the state space. The online stage uses the policy for BHRL online learning. To solve each POSMDP  $\mathcal{P}$  in the offline stage, we sample  $M$  primitive models from a prior distribution  $b_0(\theta)$ , then create a new approximate POSMDP  $\hat{\mathcal{P}} = \langle \mathcal{S}_{\hat{\mathcal{P}}}, \mathcal{A}_{\hat{\mathcal{P}}}, \mathcal{O}_{\hat{\mathcal{P}}}, \mathbb{T}_{\hat{\mathcal{P}}}, \mathbb{R}_{\hat{\mathcal{P}}}, \gamma, b_{\hat{\mathcal{P}}}^0 \rangle$ . This approximate POSMDP has state space  $\mathcal{S}_{\hat{\mathcal{P}}} = \mathcal{S} \times \{1, \dots, M\}$ , action space  $\mathcal{A}_{\hat{\mathcal{P}}} = \mathcal{A}$ , observation space  $\mathcal{O}_{\hat{\mathcal{P}}} = \mathcal{S}$ , reward function  $\mathbb{R}_{\hat{\mathcal{P}}}(s, m, a, s', m') = R(s, a, s')$ , the subsumed transition and observation function

$$\mathbb{T}_{\hat{\mathcal{P}}}(s, m, a, m', s', o, k) = \mathbb{T}_{\mathcal{P}}(s, \hat{\theta}^m, a, s', \hat{\theta}^{m'}, o, k) \delta_{mm'},$$

and the initial belief  $b_{\hat{\mathcal{P}}}^0$  uniformly distributed over  $\{1, \dots, M\}$ .

---

#### Algorithm 1 MC-BHRL Planning

---

- 1: **Require:** A hierarchy  $\mathcal{H}$ .
  - 2: **for** each subtask  $a \in \mathcal{H}$  **do** solve in bottom-up ordering
  - 3: Formulate a POSMDP  $\mathcal{P}$  with lower-level macro actions.
  - 4: Sample  $M$  primitive models  $\{\hat{\theta}^1, \dots, \hat{\theta}^M\} \sim b_0(\theta, \mathcal{P})$ .
  - 5: Form a new approximate POSMDP  $\hat{\mathcal{P}}$ .
  - 6: Use Macro-MCVI to solve POSMDP  $\hat{\mathcal{P}}$  for a policy  $\hat{\pi}_a^*$ .
  - 7: **end for**
  - 8: **return**  $\hat{\pi}_a^*$ .
- 

#### 4.2 Theoretical analysis

We compute the error bound of one step solving a subtask of MC-BHRL. The total error bound (at the root node in hierarchy) could be computed similarly by adapting the  $R_{max}$  value to a total maximum reward of macro actions.

MC-BHRL uses Macro-MCVI to solve the approximate POSMDP  $\hat{\mathcal{P}}$ , and represents its policy as a policy graph. Thus, there is a correspondence between policies of  $\hat{\mathcal{P}}$  and  $\mathcal{P}$  if the policy of  $\mathcal{P}$  is also represented in a policy graph, because they have the same action space  $\mathcal{A}$  and observation space  $\mathcal{O}$ . We denote that  $\pi^*$  is an optimal policy of  $\mathcal{P}$ ,  $\hat{\pi}^*$  is an optimal policy of  $\hat{\mathcal{P}}$ , and  $\hat{\pi}_t$  is a computed policy of  $\hat{\mathcal{P}}$  after  $t$ -step approximate backup of Macro-MCVI.

First, we bound the error for solving  $\hat{\mathcal{P}}$ , i.e., the difference between  $\hat{V}_t$  of  $\hat{\pi}_t$  and  $\hat{V}^*$  of  $\hat{\pi}^*$ . Let  $\delta_B = \sup_b \min_{b'} \|b - b'\|_1$  be the maximum  $L_1$  distance from any point in the belief space  $\mathcal{B}$  to the closest point in  $B$ , where  $B$  is a set of belief points sampled from  $\mathcal{B}$ . Macro-MCVI uses Monte-Carlo simulations to approximate the exact value iteration backup by sampling  $N$  times from a belief point  $b$ . This results in the approximate backup operator  $\hat{H}_B$ .

**Theorem 3** For every  $b \in \mathcal{B}$ ,

$$|\hat{V}^*(b) - \hat{V}_t(b)| \leq \frac{2R_{max}}{(1-\gamma)^2} \sqrt{\frac{2(|\mathcal{S}_{\hat{\mathcal{P}}}| \ln(|B|t) + \ln(2|\mathcal{A}|) + \ln(|B|t/\tau))}{N}} + \frac{2R_{max}}{(1-\gamma)^2} \delta_B + \frac{2\gamma^t R_{max}}{(1-\gamma)} \tag{12}$$

with probability at least  $1 - \tau$

Our proof uses the same reasoning as the proof in [5], and thus omitted here. The difference is in the contraction property derived in Lemma 1 and Lipschitz condition obtained by using the convex and piecewise linear properties established in Theorem 2.

The theorem shows that the error is in order of  $O(1/\sqrt{N})$ , and can be reduced by increasing the number of samples  $N$ . The following theorem uses the previous result to establish the error between an optimal policy  $\pi^*$  and  $\hat{\pi}_t$  for the original POSMDP  $\mathcal{P}$ . Recall that the policy  $\hat{\pi}_t$  is computed by solving  $\hat{\mathcal{P}}$ . We denote that  $|\pi|$  is the number of nodes in policy graph used to represent  $\pi$ .

**Theorem 4** If  $\hat{\pi}_t$  is the policy received by solving  $\hat{\mathcal{P}}$ , then the the difference between it and an optimal policy  $\pi^*$  can be bounded as

$$V_{\pi^*} - V_{\hat{\pi}_t} \leq \frac{2R_{max}}{1-\gamma} \sqrt{\frac{2((|\hat{\pi}_t| |\mathcal{O}| + 2) \ln |\hat{\pi}_t| + |\hat{\pi}_t| \ln \mathcal{A} + \ln(4/\tau))}{M}}$$

$$\begin{aligned}
 &+ \frac{2R_{max}}{(1-\gamma)^2} \sqrt{\frac{2(|\mathcal{S}_{\hat{P}}| \ln(|B|t) + \ln(2|A|) + \ln(|B|t/\tau))}{N}} \\
 &+ \frac{2R_{max}}{(1-\gamma)^2} \delta_B + \frac{2\gamma^t R_{max}}{(1-\gamma)}
 \end{aligned}
 \tag{13}$$

The proof is similar to the proof in [59]. The first component on the right-hand side is the error between the value functions of any policy for  $\hat{\mathcal{P}}$  and  $\mathcal{P}$ . The remaining is from solving  $\hat{\mathcal{P}}$  approximately, which has been derived in Theorem 3.

The first term, decaying at a rate of  $O(1/\sqrt{M})$ , can be reduced if we increase the number of samples  $M$  from the initial belief  $b_0(\mathcal{P})$ . Similarly, the errors from solving  $\hat{\mathcal{P}}$  in the remaining terms are reduced if we sample a larger number  $N$  of beliefs, reduce  $\delta_B$  to cover better  $\mathcal{B}$ , and do more MC-Backup iterations  $t$ . However, if  $M$  is large, then we obtain a larger approximate POSMDP  $\hat{\mathcal{P}}$  problem (a larger state space) which would yield a larger policy graph  $\hat{\pi}_t$ . Similarly, decreasing  $\delta_B$  by running Macro-MCVI longer makes the cover  $|B|$  larger. So, there is a trade-off between training error and generalization error. This is similar to the problem of overfitting in statistical learning theory as also discussed in [59].

### 4.3 RL in POMDP environments

We briefly describe how our proposed approach is easily extended to apply for RL setting in POMDP environments. Assuming that a BHRL problem in a POMDP environment is defined with a tuple  $\{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mathcal{R}, \gamma\}$ . This BHRL is also given an action decomposition hierarchy, and with the assumption of unknown transition function  $\mathcal{T}$ , and unknown observation function  $\mathcal{Z}$ . The unknown transition and observation functions are parameterized by  $\theta$  and  $\psi$ , respectively, as  $\theta_a^{s's'}$  =  $\mathcal{T}(s'|s, a)$  and  $\psi_a^{s'o}$  =  $\mathcal{Z}(o|s', a)$ . Similar to the method in [59], MC-BHRL can be generalized to solve these problems. We first sample  $M$  hypotheses  $(\theta, \psi)$  from the joint prior distribution  $b_0(\theta, \psi)$ , then form the approximate POSMDP  $\hat{\mathcal{P}}$ . Solving for an approximate policy  $\hat{\pi}$  for  $\hat{\mathcal{P}}$  is like before. The policy should reason over uncertainty of both unknown transition and observation functions.

## 5 Evaluation and analysis

We evaluate the performance of MC-BHRL on three simulated RL domains. First, we use a familiar variant of Taxi domain from [11] which is a fully observable MDP task. The other two are RL tasks in POMDP: Cheese-Taxi [30]

and large Cheese-Taxi [47]. For the Taxi task, we compare with flat Q-learning, MAXQ [11], Bayesian MAXQ [8], and MC-BRL [59] algorithms. For two partially observable tasks, we make comparisons mainly with MC-BRL algorithm [59]. MC-BRL is a flat Bayesian model-based RL algorithm which does not use any action decomposition hierarchy. To make fair comparisons, both MC-BHRL and MC-BRL use Macro-MCVI [26] as POSMDP/POMDP solver.

### 5.1 Taxi domain

The Taxi domain as described in Fig. 1 was first introduced in [11]. The agent is a taxi whose task is to pickup a passenger, then deliver him to his desired destination. The passenger location and his destination, which is different from his initial location, are one of four landmarks (*Red, Blue, Green, and Yellow*). The taxi is randomly placed in the grid-world after each episode. There are six actions: 4 movement actions (*North, South, East, West*), *Pickup*, and *Putdown*. For each movement action, the agent moves with probability of 0.8 to its intended direction, and probability of 0.1 for each perpendicular direction. A movement cost is  $-1$ , each wrong pickup or putdown cost is  $-10$ , and a correct putdown is 20 reward. In this task, we use a similar task hierarchy in [11] as described in Fig. 1.

In this task, we assume that the dynamics of the movement actions are not given. We evaluate MC-BRL and MC-BHRL with values of  $M = 1000, 2000$ , Macro-MCVI uses  $N = 5000$ . All performance results for comparison are reported in Table 1. The online time is for running 500 episodes. The reported rewards (cumulative reward per episode) and success rates of Q-learning, MAXQ, Bayesian MAXQ are the average performance of episodes from 450 to 500 over 10 runs. The performance results of MC-BRL and MC-BHRL are averaged over 10 offline simulations. Each simulation is online evaluated with 500 trials

**Table 1** Comparisons on average total rewards for taxi domain. For each algorithm, we report the average rewards, success rates, and the offline and online running time in seconds

Algorithm	Rew.	Suc.	Offline	Online
MC-BHRL ( $M = 1K$ )	-14.42	74%	2000	4
MC-BHRL ( $M = 2K$ )	-10.29	85%	4000	4
MC-BRL ( $M = 1K$ )	-19.76	11%	2000	4
	-19.20	12%	10000	5
MC-BRL ( $M = 2K$ )	-15.01	15%	4000	4
	-14.39	17%	10000	5
Q-Learning	-8.3	100%	0	4.5
MAXQ	-7.5	100%	0	6.5
Bayesian MAXQ	-7.88	100%	0	340



(episodes) to report an average total reward. This is a difficult task for a BRL algorithm due to a very large continuous state space (400-dimensional), so MC-BHRL could not find a near optimal policy in limited time. Since MC-BRL could not find a better policy than a random policy even after quite a long time, MC-BHRL still shows promising results over MC-BRL in terms of running time, success rate and average reward. The average reward is still less than zero while the success rate of MC-BHRL is high, because partly we let the macro action run until reaching its maximum length if not terminated (a movement takes a cost of -1.0) and partly it found a non-shortest path at each success. The online learning of MC-BHRL can be negligible since we already exploit the offline computing resource to compute a policy. This policy of MC-BHRL is with respect to all possibilities of underlying model. The online methods like Bayesian MAXQ must recompute the policy each time the problem's model changes.

### 5.2 Cheese-Taxi domain

The Cheese-Taxi domain was originally described in [30], as shown in Fig. 2. There are seven actions: 4 movement

actions (*North, South, East, West*), *Query, Pickup, and Put-down*. This is a partially observable MDP task, so the agent can not know exactly its location which can be disambiguated by taking a sequence of proper actions. It can receive one of ten observations ( $o_1, \dots, o_7, d_0, d_4, null$ ). If a movement action is taken, then 1 of first 7 observations will be observed. These are localization observation depending on the combination of walls around a location. If a Query action is taken, the agent will be informed a destination  $d_0$  or  $d_4$  if the passenger is on Taxi, and *null* otherwise. The passenger can change his destination with a probability of 0.05 if the Taxi is navigating through state 2 or 6. Thus, the agent must take Query action again to know the new destination. If either Pickup or Putdown is taken, *null* is always returned. The reward function is similar to Taxi domain described in previous section.

In this task, we use a similar task hierarchy as described in [47], which is slightly different from Taxi's hierarchy as in Fig. 3. The GET (PUT) abstract actions are similar to the ones in Taxi domain which are defined on NAVIGATE abstract action and a primitive action Pickup (Putdown). The NAVIGATE abstract action consists of 4 movement

S0 <b>D ?</b>	S1	S2	S3	S4 <b>D ?</b>
S5 		S6		S7
S8		S10 		S9



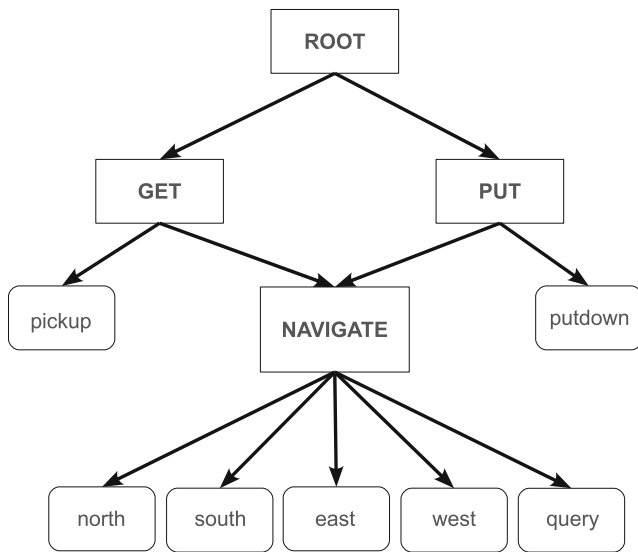
S0 <b>D ?</b>	S1	S2	S3	S4 <b>D ?</b>
S5		S6		S7
S8 	S9 <b>D ?</b>	S10	S11 <b>D ?</b>	S12
S13		S14 <b>D ?</b>		S15
S16	S17	S18 <b>D ?</b>	S19	S20 <b>D ?</b>
S21 		S22		S23
S24 <b>D ?</b>	S25	S26	S27	S28
		S29 <b>D ?</b>		

Fig. 2 Cheese-Taxi (left) and large Cheese-Taxi (right) problems



**Fig. 3** Action hierarchy of Cheese-Taxi and large Cheese-Taxi problems

actions and Query in order to both navigate to and reason about the taxi's location and the passenger's destination.

For simplicity, assuming that we are given the deterministic transition and reward functions, the observation function is partially known at all next state and action pairs, with two unknown switching probabilities at states 2 and 6. We evaluate MC-BHRL and MC-BRL with  $M = 10$ , and sample from a uniform prior belief. Both algorithms use Macro-MCVI with a setting of  $N = 50$ . The performance results are averaged over 100 offline simulations. Each simulation is online evaluated with 1000 trials to report an average total reward.

All performance results are reported in Table 2. PolCA is a hierarchical POMDP solver described in [30], its performance result is with an assumption of

a known POMDP. This problem is small enough so that both algorithms could quickly find a near-optimal policy. However, MC-BHRL with a pre-defined action hierarchy is hundred times faster than MC-BRL, a flat BRL solver.

### 5.3 Large cheese-taxi domain

The large cheese-taxi [47] is a larger variant of the cheese-taxi domain. This task, shown in Fig. 2, has more possible taxi locations (30 as compared to 11 in cheese-taxi domain), 9 possible passenger's destinations, thus it also has a larger observation space. The passenger can switch his destination between state 0 or 4 if the Taxi is navigating through state 2 or 6 and the destination is 0 or 4. Similarly, he also can switch his destination between states 20 or 24 if the Taxi is navigating through states 18, 22, or 26 and the destination is 20 or 24.

Similar to previous task, we assume that we are given the deterministic transition and reward functions. The observation function is partially known at all next state and action pairs, with five unknown switching probabilities at states 2, 6, 18, 22, and 26. We evaluate MC-BHRL and MC-BRL with values of  $M = 1000, 2000$ , and sample from a uniform prior belief. Both algorithms use Macro-MCVI with a setting of  $N = 2000$  to guarantee Monte-Carlo backup's error small. The performance results are averaged over 10 offline simulations. Each simulation is online evaluated with 1000 trials to report an average total reward.

All performance results are reported in Table 2. Flat-DDN is a hierarchical POMDP solver described in [47], its performance result is with an assumption of a known POMDP. The results show that MC-BHRL outperforms MC-BRL due to the unfinished long running time of MC-BRL. This problem has only five unknown parameters, but MC-BRL still has difficulties in finding a near-optimal

**Table 2** Comparisons on average total rewards for cheese-taxi and large cheese-taxi domains

Algorithm	Rewards	Time (seconds)
Cheese-Taxi		
MC-BHRL ( $M = 10$ )	<b>8.434 ± 0.16</b>	2
MC-BRL ( $M = 10$ )	-32.4 ± 2.24	2
	6.257 ± 0.10	15
	8.223 ± 0.09	450
PolCA	~ 8.50	N/A
Large Cheese-Taxi		
MC-BHRL ( $M = 1000$ )	-7.56 ± 1.16	1000
MC-BHRL ( $M = 2000$ )	<b>4.23 ± 0.49</b>	1000
MC-BRL ( $M = 1000$ )	-22.6 ± 5.53	1000
	-22.0 ± 5.70	10000
MC-BRL ( $M = 2000$ )	-20.4 ± 4.46	1000
	-18.6 ± 5.09	10000
Flat-DDN	8.40	N/A

policy. There is a performance gap between MC-BHRL and an optimal policy of Flat-DDN. We have checked that MC-BHRL has 100% completed the tasks, however it found longer paths due to the use of policy graph. This task needs the agent to both disambiguate its state and find the shortest path to the goal. Each time a NAVIGATE macro action is used, the agent implementing NAVIGATE's policy graph starts as if it just starts to do disambiguation, and ignores all knowledge from previous called NAVIGATE(s). This is different from the case if we use  $\alpha$ -functions to represent each macro action's policy like in [30, 47] with which we can always select the best  $\alpha$ -function with respect to the belief of the current time NAVIGATE call. There is no similar gap performance in small Cheese-Taxi domain, because the passenger's position has its own distinct observation,  $o_7$ , so it does not need to do more disambiguation after reaching this position.

This performance gap can be easily suppressed by changing the Macro-MCVI solver a little bit: If each time calling a macro action, the agent checks all possible nodes of the macro action's policy graph and chooses the best one to start with. This would make the online stage run much longer. The better solution, which is also our future research, is to change the POSMDP solver. Each time a new macro action node established in higher-level macro action's policy graph, we choose the best node from the macro action's policy graph to connect to, instead of always connect to the root node.

## 6 Discussion & conclusion

We have proposed an efficient and simple method to solve a BRL, called Monte Carlo Bayesian Hierarchical RL (MC-BHRL), by utilizing a given action decomposition. MC-BHRL was inspired from two previous algorithms MC-BRL and Macro-MCVI. We formulated the underlying BHRL problem as a POSMDP. The newly formulated POSMDP has a continuous state space, discrete action and observation spaces. To solve it, we use a similar method of MC-BRL to sample from a POSMDP's prior belief, then apply the Macro-MCVI solver. This results in an offline BHRL solver, which is new and brings an alternative solution compared to the recently introduced online solver Bayesian MAXQ. We have shown, through three RL domains, that action hierarchy is necessary for solving BRL, as argued in [8]. The performance of BHRL is better than that of flat BRL in terms of both reward and especially solving time in at least one order of magnitude. There are a number of future research directions of MC-BHRL. An automatic hierarchy discovery may also be integrated into MC-BHRL framework. With POSMDP formulation for BHRL, online POSMDP solvers can be applied to solve BHRL.

**Acknowledgments** This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2010-0012609).

## References

1. Abbeel P, Coates A, Quigley M, Ng AY (2006) An application of reinforcement learning to aerobatic helicopter flight. In: Advances in neural information processing systems (NIPS), pp 1–8
2. Abdoos M, Mozayani N, Bazzan ALC (2014) Hierarchical control of traffic signals using q-learning with tile coding. *Appl Intell* 40(2):201–213
3. Asmuth J, Littman ML (2011) Learning is planning: near Bayes-optimal reinforcement learning via Monte-Carlo tree search. In: UAI, pp 19–26
4. Atkeson CG (1997) Nonparametric model-based reinforcement learning. In: Advances in neural information processing systems (NIPS)
5. Bai H, Hsu D, Lee WS, Vien NA (2010) Monte Carlo value iteration for continuous-state POMDPs. In: Algorithmic foundations of robotics IX, pp 175–191
6. Barto AG, Mahadevan S (2003) Recent advances in hierarchical reinforcement learning. *Discrete Event Dyn Syst* 13(4): 341–379
7. Baxter J, Tridgell A, Weaver L (2000) Learning to play chess using temporal differences. *Mach Learn* 40(3):243–263
8. Cao F, Ray S (2012) Bayesian hierarchical reinforcement learning. In: Bartlett P, Pereira F, Burges C, Bottou L, Weinberger K (eds) Advances in neural information processing systems (NIPS), pp 73–81
9. Castro PS, Precup D (2007) Using linear programming for Bayesian exploration in Markov decision processes. In: IJCAI, pp 2437–2442
10. Dearden R, Friedman N, Russell SJ (1998) Bayesian Q-learning. In: AAAI, pp 761–768
11. Dietterich TG (2000) Hierarchical reinforcement learning with the MAXQ value function decomposition. *J Artif Intell Res (JAIR)* 13:227–303
12. Duff M (2002) Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes. PhD thesis, University of Massachusetts Amherst
13. Engel Y, Mannor S, Meir R (2003) Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In: Proceedings of the international conference on machine learning, pp 154–161
14. Engel Y, Mannor S, Meir R (2005) Reinforcement learning with Gaussian processes. In: Proceedings of the International Conference on Machine Learning, pp 201–208
15. Furnstun T, Barber D (2010) Variational methods for reinforcement learning. In: AISTATS, pp 241–248
16. Ghavamzadeh M, Engel Y (2006) Bayesian policy gradient algorithms. In: Advances in neural information processing systems (NIPS), pp 457–464
17. Ghavamzadeh M, Engel Y (2007) Bayesian actor-critic algorithms. In: Proceedings of the international conference on machine learning, pp 297–304
18. Granmo OC, Glimsdal S (2012) Accelerated Bayesian learning for decentralized two-armed bandit based decision making with applications to the goore game. *Appl Intell*
19. 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

20. Hauskrecht M, Meuleau N, Kaelbling LP, Dean T, Boutilier C (1998) Hierarchical solution of Markov decision processes using macro-actions. In: UAI, pp 220–229
21. He R, Brunskill E, Roy N (2010) PUMA: Planning under uncertainty with macro-actions. In: Proceedings of the association for the advancement of artificial intelligence (AAAI)
22. Hong J, Prabhu VV (2004) Distributed reinforcement learning control for batch sequencing and sizing in just-in-time manufacturing systems. *Applied Intelligence* 20(1):71–87
23. Iglesias A, Martínez P, Aler R, Fernández F. (2009) Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. *Appl Intell* 31(1):89–106
24. Jong NK, Stone P (2008) Hierarchical model-based reinforcement learning: Rmax + MAXQ. In: Proceedings of the international conference on machine learning
25. Li J, Li Z, Chen J (2011) Microassembly path planning using reinforcement learning for improving positioning accuracy of a 1 cm<sup>3</sup> omni-directional mobile microrobot. *Appl Intell* 34(2):211–225
26. Lim ZW, Hsu D, Sun LW (2011) Monte Carlo value iteration with macro-actions. In: Advances in neural information processing systems (NIPS), pp 1287–1295
27. Ngo H, Luciw M, Förster A, Schmidhuber J (2012) Learning skills from play: Artificial curiosity on a Katana robot arm In: Proceedings of the international joint conference of neural networks (IJCNN)
28. Ngo H, Luciw M, Förster A, Schmidhuber J (2013) Confidence-based progress-driven self-generated goals for skill acquisition in developmental robots. *Front Psychol* 4
29. Pakizeh E, Palhang M, Pedram MM (2012) Multi-criteria expertness based cooperative Q-learning. *Appl Intell*
30. Pineau J (2004) Tractable planning under uncertainty: exploiting structure. Ph.D. thesis. Robotics Institute, Carnegie Mellon University
31. Pineau J, Thrun S (2001) An integrated approach to hierarchy and abstraction for POMDPs. Tech. rep. Carnegie Mellon University, Robotics Institute
32. Porta JM, Vlassis NA, Spaan MTJ, Poupart P (2006) Point-based value iteration for continuous POMDPs. *JMLR* 7:2329–2367
33. Poupart P, Vlassis NA, Hoey J, Regan K (2006) An analytic solution to discrete Bayesian reinforcement learning. In: Proceedings of the international conference on machine learning, pp 697–704
34. Ross S, Chaib-draa B, Pineau J (2007) Bayes-adaptive POMDPs. In: Advances in neural information processing systems (NIPS)
35. Ross S, Pineau J (2008) Model-based bayesian reinforcement learning in large structured domains. In: UAI, pp 476–483
36. Samuel AL (1959) Some studies in machine learning using the game of checkers. *IBM J Res Dev* 3(3):210–229
37. Singh SP, Bertsekas D (1996) Reinforcement learning for dynamic channel allocation in cellular telephone systems. In: Advances in neural information processing systems (NIPS), pp 974–980
38. Strens MJA (2000) A Bayesian framework for reinforcement learning. In: Proceedings of the international conference on machine learning, pp 943–950
39. Sun S (2013) A review of deterministic approximate inference techniques for Bayesian machine learning. *Neural Comput Appl* 23(7-8):2039–2050
40. Sutton RS, Barto AG (1998) Reinforcement learning: An introduction. MIT Press, Cambridge, MA
41. Sutton RS, Precup D, Singh SP (1999) Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif Intell* 112(1-2):181–211
42. Szepesvári C (2010) Algorithms for reinforcement learning. *Synth Lect Artif Intell Mach Learn* 4(1):1–103
43. Tesauro G (1992) Practical issues in temporal difference learning. *Mach Learn* 8:257–277
44. Tesauro G (1994) TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput* 6(2):215–219
45. Tesauro G (1995) Temporal difference learning and TD-Gammon. *Commun ACM* 38(3):58–68
46. Theodorou G, Kaelbling LP (2003) Approximate planning in POMDPs with macro-actions. In: Advances in neural information processing systems (NIPS)
47. Turkett WH (1998) Robust multiagent plan generation and execution with decision theoretic planners. Ph.D. thesis, Department of Computer Science and Engineering, University of South Carolina
48. Vien NA, Chung T (2007) Natural gradient policy for average cost SMDP problem. In: Proceedings of the IEEE international conference on tools with artificial intelligence, pp 11–18
49. Vien NA, Chung T (2008) Policy gradient semi-Markov decision process. In: Proceedings of the IEEE international conference on tools with artificial intelligence, pp 11–18
50. Vien NA, Ertel W, Chung T (2013) Learning via human feedback in continuous state and action spaces. *Appl Intell* 39(2)
51. Vien NA, Ertel W, Dang VH, Chung T (2013) Monte-Carlo tree search for Bayesian reinforcement learning. *Appl Intell* 39(2):345–353
52. Vien NA, Ngo H, Ertel W (2014) Monte Carlo Bayesian hierarchical reinforcement learning. In: Proceedings of the international conference on autonomous agents and multi-agent systems (AAMAS), pp 1551–1552. International Foundation for Autonomous Agents and Multiagent Systems
53. Vien NA, Viet NH, Lee S, Chung T (2007) Heuristic search based exploration in reinforcement learning. In: IWANN, pp 110–118
54. Vien NA, Viet NH, Lee S, Chung T (2007) Obstacle avoidance path planning for mobile robot based on ant-q reinforcement learning algorithm. In: ISNN (1), pp 704–713
55. Vien NA, Viet NH, Lee S, Chung T (2009) Policy gradient SMDP for resource allocation and routing in integrated services networks. *IEICE Trans* 92-B(6):2008–2022
56. Vien NA, Yu H, Chung T (2011) Hessian matrix distribution for Bayesian policy gradient reinforcement learning. *Info Sci* 181(9):1671–1685
57. Viet NH, Vien NA, Chung T (2008) Policy gradient SMDP for resource allocation and routing in integrated services networks. In: ICNSC, pp 1541–1546
58. Wang T, Lizotte DJ, Bowling MH, Schuurmans D (2005) Bayesian sparse sampling for on-line reward optimization. In: Proceedings of the international conference on machine learning, pp 956–963
59. Wang Y, Won KS, Hsu D, Lee WS (2010) Monte Carlo Bayesian reinforcement learning. In: Proceedings of the international conference on machine learning
60. White CC (1976) Procedures for the solution of a finite-horizon, partially observed, semi-Markov optimization problem. *Oper Res* 24(2):348–358
61. Wu B, Zheng HY, Feng YP (2014) Point-based online value iteration algorithm in large pomdp. *Appl Intell*:546–555
62. Zhang W, Dietterich TG (1995) A reinforcement learning approach to job-shop scheduling. In: International joint conferences on artificial intelligence, pp 1114–1120