Policy Gradient Algorithms with Monte Carlo Tree Learning for Non-Markov Decision Processes

Tetsuro Morimura
morimura_tetsuro@cyberagent.co.jp
CyberAgent

Kazuhiro Ota
CyberAgent

Kenshi Abe
CyberAgent

Peinan Zhang
CyberAgent

Abstract

Policy gradient (PG) is a reinforcement learning (RL) approach that optimizes a parameterized policy model for an expected return using gradient ascent. While PG can work well even in non-Markovian environments, it may encounter plateaus or peakiness issues. As another successful RL approach, algorithms based on Monte Carlo Tree Search (MCTS), which include AlphaZero, have obtained groundbreaking results, especially in the game-playing domain. They are also effective when applied to non-Markov decision processes. However, the standard MCTS is a method for decision-time planning, which differs from the online RL setting. In this work, we first introduce Monte Carlo Tree Learning (MCTL), an adaptation of MCTS for online RL setups. We then explore a combined policy approach of PG and MCTL to leverage their strengths. We derive conditions for asymptotic convergence with the results of a two-timescale stochastic approximation and propose an algorithm that satisfies these conditions and converges to a reasonable solution. Our numerical experiments validate the effectiveness of the proposed methods.

1 Introduction

Reinforcement learning (RL) attempts to learn a policy model so as to maximize the average of cumulative rewards (Sutton & Barto, 2018). Policy gradient (PG) algorithms employ gradient ascent on policy parameters (Gullapalli, 1990; Williams, 1992; Baxter & Bartlett, 2001). They can benefit much from recent advances in neural network models and have been applied in various challenging domains, such as robotics (Peters & Schaal, 2008), text generation (Rennie et al., 2017; Ouyang et al., 2022), and speech recognition (Zhou et al., 2018).

Monte Carlo Tree Search (MCTS) is another successful RL approach, combining Monte Carlo sampling with an optimistic tree search that balances exploration and exploitation (Kocsis & Szepesvári, 2006; Coulom, 2006; Browne et al., 2012). Notably, when integrated with deep learning, as in AlphaZero (Silver et al., 2017b;a) and MuZero (Schrittwieser et al., 2020), MCTS algorithms have achieved groundbreaking results in board games (Silver et al., 2016).

Ordinary RL assumes the environment has the Markov property, i.e., the reward process and system dynamics of the underlying process are Markovian. More specifically, they depend only on the current state (and action); in other words, given the current state, they are independent of the past states. It enables computationally effective dynamic programming techniques to learn policy models (Puterman, 1994; Bertsekas, 1995). However, in many real-world RL tasks, it is difficult to determine in advance a good state set or space that satisfies the Markov property (Yu et al., 2011; Friedrich et al., 2011; Berg et al., 2012; Clarke et al., 2015; Rennie et al., 2017; Paulus et al., 2018; Zhou et al., 2018; You et al., 2018).

There are at least two typical scenarios where the Markov property is violated. The first is related to the observation. If observations are limited and partial, the dynamics and rewards are not Markovian and need to be modeled with functions of the past observation sequence or functions of a latent state.

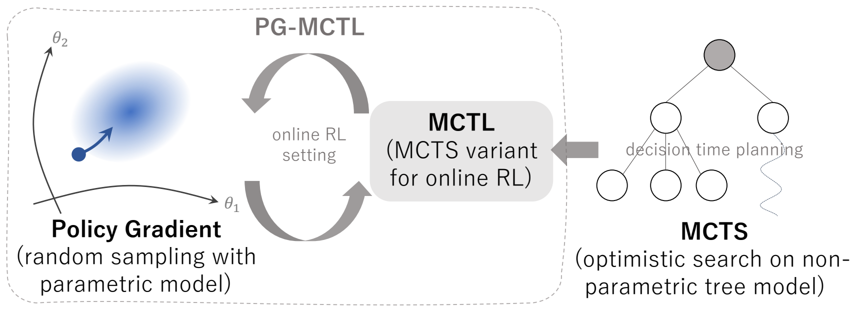
PG-MCTL
θ_2
θ_1
online RL setting
MCTL
(MCTS variant for online RL)
decision time planning
Policy Gradient
(random sampling with parametric model)
MCTS
(optimistic search on non-parametric tree model)

Figure 1: Overview of the proposed approach; *PG guided by Monte Carlo Tree Learning (PG-MCTL)*. Unlike MCTS, which requires a simulator to generate possible future states and rewards, MCTL builds a tree based on real trajectories experienced by an agent while still inheriting core MCTS properties. PG and MCTL have fundamentally different properties. PG-MCTL takes advantage of them.

Typical examples are dialog systems (Young et al., 2013) and robot navigation (Berg et al., 2012). The other case is when only the reward function is not Markovian. Generation tasks, such as text (Yu et al., 2017) and molecular graphs (You et al., 2018), are a typical examples since generated objects are usually evaluated not only from a local but also from a global perspective, such as an ad-quality score in the domain of text generation for search engine advertising (Kamigaito et al., 2021). The former scenario is often formulated as a partially observable Markov decision process (POMDP) (Kaelbling et al., 1996; Sondik, 1971), while the latter is a decision process with non-Markovian reward (Bacchus et al., 1996). The stochastic process that includes both is called a non-Markovian decision process (NMDP) or history-based decision process (HDP) (Whitehead & Lin, 1995; Bacchus et al., 1997; Majeed & Hutter, 2018), which is the focus of this paper.

Notably, both PG and MCTS algorithms are applicable to HDP-modeled tasks (Kimura et al., 1997; Aberdeen, 2003; Rennie et al., 2017; Browne et al., 2012), as they are less reliant on the Bellman optimality equation under the Markov assumption, unlike Q learning. Moreover, PG algorithms can effectively utilize function approximators like neural networks. However, PGs are known to occasionally get trapped on plateaus, slowing down learning (Kakade, 2002; Morimura et al., 2014; Ciosek & Whiteson, 2020). Furthermore, PGs can face the 'peakiness' issue, where the initially most probable actions will gain probability mass, even if they are not the most rewarding (Choshen et al., 2020; Kiegeland & Kreutzer, 2021). Meanwhile, MCTS-based algorithms aim for a global optimum through optimistic search, balancing exploration and exploitation (Kocsis & Szepesvári, 2006; Lattimore & Szepesvári, 2020; Świechowski et al., 2021). Yet, compared to PGs, they struggle with state generalization, often lacking information on states outside of their tree. Furthermore, while PGs do not require a simulator for action selection, MCTSs do. That is, they are decision-time planning (Sutton & Barto, 2018), in which planning is launched and completed for every action selection.

Based on the above, we believe that PG and MCTS can complement each other's difficulties, and their combination is a promising way to solve problems in HDPs. Specifically, even when PG is suffering from plateaus or the peakiness issue, MCTS is likely to be able to continue improving because its exploration strategy is fundamentally different from PG's. In addition, PG with an appropriately parameterized model would be able to cover the inefficiency in the state generalization of MCTS. It is also generally known that a combination of models can have a positive effect (Kuncheva, 2014).

This paper considers an online model-free RL problem in HDPs, where the environment will not be estimated. That is, no simulator is available. We adapt MCTS to the online RL setup and propose Monte Carlo Tree Learning (MCTL) that selects an action without a simulator. We then consider an approach that uses a mixture of PG and MCTL policies and adjusts its mixing probability through learning. We call this approach *a policy gradient guided by MCTL* (PG-MCTL) (Figure 1). We derive conditions for asymptotic convergence and find that naive mixing of PG and MCTL will not work in asymptotic convergence. We propose an algorithm that satisfies the conditions for convergence.

This paper is structured as follows: Section 2 covers the background of RL in HDPs, PG, and MCTS. In Section 3, we introduce the PG-MCTL approach, detail its convergence conditions using a two-timescale stochastic approximation, and present an implementation that meets these conditions. This is our main contribution. Section 4 reviews relevant literature. The effectiveness of the proposed approach is validated through experiments in Section 5, and Section 6 offers concluding remarks.

2 Preliminaries

We define our problem setting of RL in HDPs in Section 2.1. PG and MCTS algorithms are briefly reviewed in Sections 2.2 and 2.3, respectively.

2.1 Problem setting of RL in HDP

While problems of RL are usually formulated on a Markov decision process (MDP) for ease of learning (Sutton & Barto, 2018), as described in Section 1, it is difficult to define Markovian states in many real-world tasks. Here, we consider a discrete-time episodic HDP (Whitehead & Lin, 1995; Majeed & Hutter, 2018) as a general decision process without assuming the Markovian property. It is defined by a tuple $\mathsf{HDP} \triangleq \{\mathcal{O}, \mathcal{A}, T, p_{\mathrm{ini}}, p_{\mathrm{o}}, f_{\mathrm{r}}\}$, where $\mathcal{O}$ and $\mathcal{A}$ are finite sets of observations and actions, respectively. T is the length of each episode, $p_{\mathrm{ini}} : \mathcal{O} \to [0,1]$ is a probability function of the initial observation, $p_{\mathrm{ini}}(o_0) \triangleq \Pr(o_0)$[1], and $p_{\mathrm{o}} : \mathcal{O}\times\mathcal{H}_t\times\mathcal{A} \to [0,1]$ is a history-dependent observation probability function at each time step $t \in \{0, 1, \ldots, T-1\}$, $p_{\mathrm{o}}(o_{t+1} \,|\, h_t, a_t) \triangleq \Pr(O_{t+1}=o_{t+1} \,|\, H_t = h_t, A_t = a_t)$, where $h_t \triangleq [o_0, a_0, \ldots, o_{t-1}, a_{t-1}, o_t] = [h_{t-1}, a_{t-1}, o_t]$ is a history up to a time step t, and $\mathcal{H}_t \triangleq (\mathcal{O} \times \mathcal{A})^t \times \mathcal{O}$ is a set of histories at a time step t. For brevity, we notate the total history set $\mathcal{H} \triangleq \bigcup_{t=0}^{T} \mathcal{H}_t$ and the history transition probability function $p_{\mathrm{h}} : \mathcal{H}_{t+1} \times \mathcal{H}_t \times \mathcal{A} \to [0,1]$ such as $p_{\mathrm{h}}(h_{t+1} = [h_t, a_t, o_{t+1}] \,|\, h_t, a_t) \triangleq p_{\mathrm{o}}(o_{t+1}|h_t, a_t)$. The function $f_{\mathrm{r}} : \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ is a history-dependent bounded reward function, which defines an immediate reward $r_t = f_{\mathrm{r}}(h_t, a_t)$ at time step $t \in \{0, \ldots, T\}$.

A learning agent chooses an action according to a policy model $\pi : \mathcal{A} \times \mathcal{H} \to [0,1]$, which is a conditional action probability function $\pi(a|h_t) \triangleq \Pr(a \,|\, h_t, \pi)$ at each time step t. Without loss of generality, we assume that the agent can take any action $a \in \mathcal{A}$ in any $h_t \in \mathcal{H}_t$ at any $t \in \{0, \ldots, T\}$.

Here, we consider a standard online RL problem, where p_{ini}, p_{o}, and f_{r} are unknown to the agent. The agent learns the policy model π by experiencing episodes repeatedly. The objective function that the agent seeks to maximize is the expected return

$$\Upsilon(\pi) \triangleq \mathbb{E}^{\pi}[G_0], \tag{1}$$

where $\mathbb{E}^{\pi}[\,\cdot\,] \triangleq \mathbb{E}[\,\cdot\,|\,\mathsf{HDP}, \pi]$ is the expectation operator and $G_t \triangleq \sum_{\kappa=t}^{T} R_\kappa = \sum_{\kappa=t}^{T} f_{\mathrm{r}}(H_\kappa, A_\kappa)$ is a random variable of the return at time step t.

2.2 Policy gradient

We assume that the policy model π^θ to be optimized by PG algorithms is parameterized by a parameter $\theta \in \mathbb{R}^d$ and π^θ is differentiable with respect to θ. Examples of π^θ include a neural network for sequence modeling. The PGs are based on the gradient method of the following update rule with a small learning rate $\alpha \geq 0$,

$$\theta := \theta + \alpha \nabla_\theta \Upsilon(\pi^\theta),$$

[1]Although it should be $\Pr(O_0 = o_0)$ for the random variable O_0 and realization o_0 to be precise, we write $\Pr(o_0)$ for brevity. The same rule is applied to the other probability functions if there is no confusion.

where := is the right-to-left substitution operator and $\nabla_\theta \Upsilon(\pi^\theta) \triangleq [\partial\Upsilon(\pi^\theta)/\partial\theta_1, ..., \partial\Upsilon(\pi^\theta)/\partial\theta_d]^\top$ is the gradient of $\Upsilon(\pi^\theta)$ with respect to θ. Because the analytical evaluation of $\nabla_\theta \Upsilon(\pi^\theta)$ is generally intractable, a PG method, called REINFORCE (Williams, 1992), updates θ_n after every episode n of experience $[o_0, a_0, r_0, \ldots, o_T, a_T, r_T]$ according to a stochastic gradient method as follows:

$$\theta_{n+1} = \theta_n + \alpha_n \sum_{t=0}^{T} \nabla_\theta \log \pi^{\theta_n}(a_t|h_t)\,(g_t - b(h_t)), \tag{2}$$

since the gradient $\nabla_\theta \Upsilon(\pi^\theta)$ is written as

$$\nabla_\theta \Upsilon(\pi^\theta) = \mathbb{E}^{\pi^\theta}\Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi^\theta(A_t, H_t)\,(G_t - b(H_t))\Bigg],$$

where g_t is the realized value of the return G_t and $b : \mathcal{H} \to \mathbb{R}$ is an arbitrary baseline function. The baseline function b is used for reducing the variance of the stochastic gradient, and does not induce any bias to the gradient because of $\mathbb{E}^{\pi^\theta}[\nabla_\theta \log \pi^\theta(A_t|h)\, b(h)] = b(h)\nabla_\theta \sum_{a\in\mathcal{A}} \pi^\theta(a|h) = b(h)\nabla_\theta 1 = 0$.

2.3 Monte Carlo tree search

Monte Carlo tree search (MCTS) is developed to identify the best action in a given situation for decision processes (Kocsis & Szepesvári, 2006; Coulom, 2006; Browne et al., 2012). It is typically employed for decision-time planning, where planning is initiated and completed per action selection.

It is typically employed for decision-time planning, where planning is initiated and completed per action selection with a simulator. Here, a simulator is an environment model that can generate possible future states and rewards given a current state and action. This allows the algorithm to simulate different action sequences to plan the optimal policy. In planning, MCTS iteratively runs an episode from a given situation and stores its result in a tree by expanding the tree and updating statistics in nodes of the tree. After a certain number of iterations, it estimates the best action in the situation by using statistics of the root node and terminates the planning.

In single-agent learning in a stochastic system, the tree usually has two kinds of nodes, a history node and a history-action node, alternating in the depth direction. A history node represents a history and does not store additional information. At a history node, the tree-search policy determines which child history-action node to transition to, based on the statistics in its child nodes. In contrast, the transition from a history-action node to a history node follows the transition probability p_h.

Each history-action node holds a return estimate q and the number of visits m as the statistics. Here, we notate those statistics in each history-action node (h, a) with a tabular representation, as $q(h, a)$ and $m(h, a)$, for simplicity. The tree policy at a history node h, often utilizing the Upper Confidence Bounds applied for Trees (UCT) formula (Kocsis & Szepesvári, 2006), decides an action based on the statistics of its child nodes:

$$\arg\max_a \left\{ q(h,a) + C\sqrt{\frac{\log(\sum_b m(h,b))}{m(h,a)}} \right\}, \tag{3}$$

where $C \geq 0$ is a hyper-parameter to control the balance between exploration and exploitation.

Each iteration of the MCTS consists of four consecutive phases:

(i) selection of child nodes from the root to a leaf node in the tree,
(ii) tree expansion by creating new child nodes that are initialized as $m{:=}1,\ q{:=}0$,
(iii) simulation from one of the new nodes according to a default policy to sample a return,
(iv) backpropagation of the results until the root node.

Note that (ii) and (iii) are skipped if the leaf node reached is a terminal node, and (iii) is the “Monte Carlo” part of the algorithm. The default policy in (iii) is usually a uniform random policy.

In the backpropagation phase of (iv), the statistics of the node visited at each depth t of iteration $n \in \{1, 2, \dots\}$ are updated as follows:

$$\begin{cases} m_{n+1}(h_t, a_t) = m_n(h_t, a_t) + 1, \\ q_{n+1}(h_t, a_t) = q_n(h_t, a_t) + \frac{1}{m_n(h_t, a_t)}(g_t - q_n(h_t, a_t)). \end{cases} \tag{4}$$

Note that many other update rules have been proposed, such as the TD(λ) learning type (Browne et al., 2012; Vodopivec et al., 2017).

3 Policy gradient guided by MCTL

In Section 3.1, we introduce Monte Carlo Tree Learning (MCTL) as an MCTS variant for online RL. We then outline our approach, PG guided by MCTL (PG-MCTL), in Section 3.2. Convergence analysis is discussed in Section 3.3, followed by a convergent implementation proposal in Section 3.4.

3.1 Monte Carlo tree learning (MCTL)

Our focus is an online model-free RL problem in HDPs, where no simulator is available. An agent learns through interactions with an unknown environment without estimating it. The standard MCTS, however, typically necessitates a simulator. In response, we propose a variant of MCTS that inherits its key features but eliminates the need for a simulator. This is referred to as a lazy MCTS or simply, MCTL.

MCTL gradually grows a tree according to the trajectories experienced by the agent interacting with an unknown environment. Unlike MCTS, MCTL maintains and updates a tree over multiple episodes.[2] Specifically, after experiencing an episode, the tree is updated according to MCTS's tree expansion and backpropagation procedures (e.g. Eq. (4)). Depending on the presence of a node for the current situation h in the tree, the MCTL policy adjusts its action selection. If an MCTL policy is queried on a history h that the tree contains, it selects an action according to the node selection procedure of MCTS (e.g. Eq. (3)). Otherwise, it selects an action randomly.

By design, an MCTL tree will have nodes for states near the initial state and/or frequently visited states. The presence of a node h implies that the history h is somewhat known to an MCTL policy. In contrast, the absence of a node h' implies that the history h' is unknown and the policy should select an action for exploration at h'. This is analogous to a common class of algorithms, *knows what it knows* (KWIK), for efficient exploration (Li et al., 2011).

We will notate an MCTL policy as π^ω, whose parameter is ω. Specific implementations, including update rules, are described in Section 3.4.

3.2 General approach

As discussed in the introduction, combining PG and MCTL aims to leverage the strengths and mitigate the weaknesses of each individual method. Our proposed approach, PG guided by MCTL (PG-MCTL), integrates them by randomly selecting a policy of either PG or MCTL at each time step. Specifically, we consider the following mixture of policies π^θ and π^ω:

$$\pi^{\theta,\omega}(a|h) \triangleq (1 - \lambda(h))\pi^\theta(a|h) + \lambda(h)\pi^\omega(a|h), \tag{5}$$

where λ is a mixing probability. The λ may be constant or depend on an observation o, history h, and parameters θ, ω. The parameters θ and ω are updated with modified update rules of a PG and MCTL, respectively, which are proposed in Section 3.4 by using the results in Section 3.3.

[2]In MCTS, the search tree is built and used within a single episode, often requiring a simulator to generate possible future states. In contrast, MCTL continuously updates a single tree over multiple episodes using only the states experienced by the agent. This allows MCTL to function without a simulator, as it relies solely on real-world interactions.

3.3 Convergence analysis

We present the convergence conditions of PG-MCTL on a few settings of the mixing probability λ_n in Eq. (5). Proofs are shown in Appendix A.

We will validate our assumptions in the next section. For now, let's assume the updates of parameters $\theta \in \mathbb{R}^d$ for a PG policy and $\omega \in \mathbb{R}^e$ for an MCTL policy can be rewritten as follows:

$$\theta_{n+1} = \theta_n + \alpha_n [\, k(\theta_n, \omega_n) + M^{(1)}_{n+1} + \epsilon^{(1)}_n], \tag{6}$$
$$\omega_{n+1} = \omega_n + \eta_n [\, l(\theta_n, \omega_n) + M^{(2)}_{n+1} + \epsilon^{(2)}_n], \tag{7}$$

where $k : \mathbb{R}^d \times \mathbb{R}^e \to \mathbb{R}^d$ and $l : \mathbb{R}^d \times \mathbb{R}^e \to \mathbb{R}^e$ are the expected update functions, $M^{(1)} \in \mathbb{R}^d$ and $M^{(2)} \in \mathbb{R}^e$ are noise terms, and $\epsilon^{(1)} \in \mathbb{R}^d$ and $\epsilon^{(2)} \in \mathbb{R}^e$ are bias terms.

We make the following assumptions about the noise and bias terms, which are common in the stochastic approximation (Borkar, 2008).

Assumption 1. *The stochastic series $\{M^{(i)}_n\}$ for $i = 1, 2$ is a martingale dierence sequence, i.e., with respect to the increasing σ-elds, $\mathcal{F}_n \triangleq \sigma(\theta_m, \omega_m, M^{(1)}_m, M^{(2)}_m, m \leq n)$, for some constant $K > 0$, the following holds for all $n \in \{1, 2, \dots\}$,*

$$\mathbb{E}[\, M^{(i)}_{n+1} \,|\, \mathcal{F}_n] = 0,$$
$$\mathbb{E}[\, \|M^{(i)}_{n+1}\|^2 \,|\, \mathcal{F}_n] \leq K(1 + \|\theta_n\|^2 + \|\omega_n\|^2).$$

Assumption 2. *The bias $\{\epsilon^{(i)}_n\}$ for $i = 1, 2$ is a deterministic or random bounded sequence which is $o(1)$, i.e., $\lim_{n\to\infty} \epsilon^{(i)}_n = 0$.*

In our problem setting, the size of the history set $\mathcal{H}$ is bounded, which ensures that the assumptions regarding the noise and bias terms in Eqs. (6) and (7) are realistic and not overly restrictive.

For our analysis, we use the ordinary dierential equation (ODE) approach for the stochastic approximation (Bertsekas & Tsitsiklis, 1996; Borkar, 2008). The limiting ODEs that Eqs. (6) and (7) might be expected to track asymptotically is, for $\tau \geq 0$,

$$\dot{\theta}(\tau) = k(\theta(\tau), \omega(\tau)), \tag{8}$$
$$\dot{\omega}(\tau) = l(\theta(\tau), \omega(\tau)). \tag{9}$$

We also make the assumption about the expected update functions k and l.

Assumption 3. *The functions k and l be Lipschitz continuous maps.*

Assumption 4. *The ODE of Eq. (9) has a globally asymptotically stable equilibrium $\varphi(\theta)$, where $\varphi : \mathbb{R}^d \to \mathbb{R}^e$ is a Lipschitz map.*

Assumption 5. $\sup_n(\|\theta_n\| + \|\omega_n\|) < \infty$.

We first show the convergence analysis result for the case of that the mixing probability λ is a constant or a fixed function such as $\lambda : \mathcal{H} \to [0, 1]$.

Proposition 1. *Assume Assumptions 1–5 hold. Let the mixing probability function $\lambda : \mathcal{H} \to [0, 1]$ be invariant to the number of episodes n, and the learning rates α_n and η_n satisfying*

$$\begin{cases} \lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n = \lim_{N\to\infty} \sum_{n=0}^{N} \eta_n = \infty, \\ \lim_{N\to\infty} \sum_{n=0}^{N} \left(\alpha_n^2 + \eta_n^2\right) < \infty, \\ \lim_{N\to\infty} \frac{\alpha_N}{\eta_N} = 0. \end{cases} \tag{10}$$

Then, almost surely, the sequence $\{(\theta_n, \omega_n)\}$ generated by Eqs. (6) and (7) converges to a compact connected internally chain transitive invariant set of Eqs. (8) and (9), respectively.

The above results indicate how the learning rates α_n and η_n should be set for convergence. Since η_n in an MCTL policy is basically proportional to $\frac{1}{n}$, an obvious choice of α_n will be $\frac{1}{1+n\log n}$. Note that, if a deterministic policy such as UCT (Eq. (3)) is used, k and l will not be the Lipschitz maps and thus the above convergence results cannot be applied. Therefore, we will use the softmax function (Sutton & Barto, 2018) in the implementation in Section 3.4.

The invariant condition of λ can be relaxed.

Proposition 2. *Let $\lambda_{\theta_n} : \mathcal{H} \to [0,1]$ be a function parameterized by a part of θ and a Lipschitz continuous map with respect to θ. Assume that all the conditions of Proposition 1 are satisfied except for λ. The consequence of Proposition 1 still holds.*

Finally, we consider a specific scenario that λ_n is a decreasing function of the number of episodes n, where an MCTL policy π^ω is just used for guiding the PG. The goal, in this case, will be to obtain a parameterized policy π^θ that demonstrates good performance by itself. In the case of λ_n decreasing, the convergence condition can be significantly relaxed as follows.

Proposition 3. *Assume Assumptions 1 and 2 only for $i = 1$ hold, k is Lipschitz continuous map, and $\sup_n(\|\theta_n\|) < \infty$ holds. Let the mixing probability λ_n be $o(1)$ and satisfy $0 \le \lambda_n \le 1-\varepsilon$ for all n and a constant $\varepsilon > 0$, and the learning rate of a PG policy satisfy*

$$\lim_{N\to\infty}\sum_{n=0}^{N}\alpha_n = \infty, \quad \lim_{N\to\infty}\sum_{n=0}^{N}\alpha_n^2 < \infty.$$

Then, almost surely, the sequence $\{\theta_n\}$ generated by Eqs. (6) and (7) converges to a compact connected internally chain transitive invariant set of the ODE, $\dot{\theta}(\tau) = \nabla_\theta \Upsilon(\pi^{\theta(\tau)})$.

This proposition shows that, unlike the previous cases, the convergence property is guaranteed even if a deterministic policy like UCT is used for an MCTL policy.

3.4 Implementation

We present an implementation of PG-MCTL that satisfies the convergence conditions and converges to a reasonable solution.

First, we consider revising the update of a PG policy π^θ. Since the goal is to maximize the expected return of Eq. (1), the following update at each episode n will be appropriate, instead of the ordinary one of Eq. (2):

$$\begin{aligned}\theta_{n+1} &= \theta_n + \alpha_n \sum_{t=0}^{T} \nabla_\theta \log \pi^{\theta_n,\omega_n}(a_t|h_t)(g_t - b(h_t)) \\ &= \theta_n + \alpha_n \sum_{t=0}^{T} \rho_t \nabla_\theta \log \pi^{\theta_n}(a_t|h_t)(g_t - b(h_t)), \end{aligned} \tag{11}$$

where we assume π^{θ_n,ω_n} is the behavior policy of episode n. The ρ_t is a scaled probability ratio or a kind of importance weight (Sutton & Barto, 2018),

$$\rho_t \triangleq \frac{(1-\lambda(h_t))\pi^\theta(a_t|h_t)}{\pi^{\theta,\omega}(a_t|h_t)}. \tag{12}$$

Note that according to Proposition 2, learning the mixture probability λ via this PG update, rather than assuming λ is predefined and fixed, still ensures convergence. When adopting this approach, termed **PG-MCTL-adpt**, we assume that λ_θ is parameterized by a subset of parameters within θ. The update rule of θ is derived as

$$\theta_{n+1} = \theta_n + \alpha_n \sum_{t=0}^{T}\left\{\rho_t \nabla_\theta \log \pi^{\theta_n}(a_t|h_t) + \frac{(\pi^{\omega_n}(a_t|h_t) - \pi^{\theta_n}(a_t|h_t))}{\pi^{\theta_n,\omega_n}(a_t|h_t)} \nabla_\theta \lambda_{\theta_n}(h_t)\right\}(g_t - b(h_t)). \tag{13}$$

From here on, we will only consider the case where the mixing probability λ is constant, but the results presented here will be straightforwardly applied to other settings of λ.

Next, we consider an implementation of an MCTL policy π^ω. The update rule of MCTS that MCTL follows differs from the general stochastic approximation in Eq. (7). In particular, the learning rate in the MCTS update of Eq. (4) varies among nodes. On the other hand, the stochastic approximation assumes a global learning rate η_n as seen in Eq. (7). Furthermore, although Assumption 5, vital for convergence, states that parameters should remain bounded, the parameter m, the number of visits, could diverge. To reconcile these differences, we introduce a tree-inclusion probability and reformulate the MCTS update by replacing $m(h,a)$ with $u\colon\mathcal{H}\times\mathcal{A}\to[0,1]$ so that $m(h,a)=1/u(h,a)$. Consequently, the parameter of the MCTL policy π^ω becomes $\omega\triangleq\{u,q\}$. With a learning rate of $\eta_n=1/n$ and initializing $u:=1$ and $q:=0$, the conventional MCTS update in Eq. (4) can be rewritten as:

$$\begin{cases} u_{n+1}(h_t,a_t) = u_n(h_t,a_t) + \eta_n\kappa_{n,t}\frac{-u_n(h_t,a_t)}{u_n(h_t,a_t)+1}, \\ q_{n+1}(h_t,a_t) = q_n(h_t,a_t) + \eta_n\kappa_{n,t}(g_t - q_n(h_t,a_t)), \end{cases} \tag{14}$$

where $\kappa_{n,t}$ is the following and can be regarded as an adjustment term for the learning rate per node,

$$\kappa_{n,t} \triangleq p_{n,t}\,\frac{u_n(h_t,a_t)}{\eta_n},$$

and $p_{n,t}$ is the following tree-inclusion probability,

$$p_{n,t} \triangleq \begin{cases} 1, & \text{if } t=0, \\ \min\left(\frac{1}{u_n(h_{t-1},a_{t-1})}-1,1\right), & \text{otherwise.} \end{cases} \tag{15}$$

If $u_n(h_{t-1},a_{t-1})=1$, the tree-inclusion probability $p_{n,t}$ is zero, and thus the values of (h_t,a_t) are not updated. It corresponds to the case where the tree does not have a node (h_t,a_t), indicating that this node hasn't been expanded. If $u_n(h_{t-1},a_{t-1})\le 0.5$, the values of a node (h_t,a_t) are updated with probability 1. The equivalence of the original MCTS update and Eq. (14) is shown in Appendix A.5.

We next investigate Assumption 3 about Lipschitz continuity of the expected update functions k and l. The PG update of Eq. (11) is based on the gradient ascent, and thus the expected update functions k with an ordinary implementation will satisfy Lipschitz continuity. However, the update of Eq. (14) does not allow l to have Lipschitz continuity since $\kappa_{n,t}$ diverges as $\eta_n\to 0$. This problem can be solved by modifying $\kappa_{n,t}$ with a large value $M>0$ as follows:

$$\bar{\kappa}_{n,t} \triangleq \min(\kappa_{n,t}, M). \tag{16}$$

Theorem 1. *Let the PG-MCTL update the parameterized policy π^θ by Eq. (11) and the MCTL policy π^ω by the rule in which κ in Eq. (14) is replaced by $\bar{\kappa}$ of Eq. (16), and the learning rates satisfy the conditions of Eq. (10). Also let π^θ be defined on a compact parameter space and have always bounded first and second partial derivatives, and π^ω be a softmax policy with hyper-parameters $\beta\ge 0$ and $C\ge 0$,*

$$\pi^\omega(a|h) \propto \exp\left(\beta\left\{q(h,a) + C\sqrt{u(h,a)\log\textstyle\sum_b \frac{1}{u(h,b)}}\right\}\right). \tag{17}$$

Then, $\lim_{n\to\infty}\nabla_\theta\Upsilon(\pi^{\theta_n,\omega_n})=0$ *holds.*

Finally, we propose a heuristic to avoid a vanishing gradient problem of the PG update of Eq. (11). By the definition of ρ_t in Eq. (12), if π^θ and $\pi^{\theta,\omega}$ are significantly different, ρ_t can be close to zero and thus the stochastic gradient at time t can vanish. In order to avoid this problem, we modify ρ_t to $\underline{\rho}_t$ as, with $\upsilon\in[0,1]$,

$$\underline{\rho}_t \triangleq \max(\upsilon, \rho_t). \tag{18}$$

Algorithm 1 A PG-MCTS implementation

```
1: given:
2:   - an initialized PG policy $\pi^\theta(a|h)$ and mixing probability function $\lambda_\theta(h)$ [3]
3:   - an initialized MCTL policy $\pi^\omega(a|h)$, e.g., Eq. (17)
4:   - hyper-parameters for the PG and MCTS policies
5: while within computational budget do
6:     // interaction with environment HDP
7:     observe an initial observation $h_0 \sim p_{\mathrm{ini}}$
8:     empty a memory $\mathcal{M}$ and store $h_0$ in $\mathcal{M}$
9:     for $t = 0$ to $T$ do
10:        choose a policy $\pi \in \{\pi^\theta, \pi^\omega\}$, using $\lambda(h_t)$ (see Eq. (5))
11:        choose and execute an action $a_t \sim \pi(\cdot|h_t)$
12:        observe a reward $r_t := f_{\mathrm{r}}(h_t, a_t)$
13:        observe a new history $h_{t+1} \sim p_{\mathrm{h}}(\cdot|h_t, a_t)$
14:        store $r_t$ and $h_{t+1}$ in the memory $\mathcal{M}$
15:    end for
16:    // update of policies $\pi^\theta$, $\pi^\omega$ and mixing probability $\lambda_\theta$ with $M$
17:    compute the return $g_t$, $\forall t \in \{0, \dots, T\}$
18:    update $\pi^\theta$ and $\lambda_\theta$ by the PG update of Eqs. (11) and (13) with (18)
19:    update $\pi^\omega$ by the MCTL update of Eq. (14) with (16)
20: end while
21: return the learned policy $\pi^{\theta,\omega} \triangleq \lambda\pi^\theta + (1-\lambda)\pi^\omega$
```

When $\upsilon_n = o(1)$, the convergence property in Theorem 1 still holds because υ_n is absorbed into ϵ_n in Eq. (6). Note also that ρ_t is upper bounded by 1. Thus there is no need to care about ρ_t taking a large value. The entire PG-MCTL implementation is shown in Algorithm 1.

It should be noted that the memory size may continue to grow over time as more episodes are experienced. Additionally, the maximum memory usage of the MCTL policy is on the order of the tree size, $\mathcal{O}(|\mathcal{H}|)$. While this can be significant, it is generally not expected to explore all trajectories and construct a complete tree, resulting in a smaller memory consumption.

4 Related work

Numerous studies integrate MCTS and RL algorithms. Most of them are based on the standard MCTS setting (decision time planning) and propose to use value-based RL (Vodopivec et al., 2017; Jiang et al., 2018; Efroni et al., 2019) or supervised learning (Guo et al., 2014; Silver et al., 2017b; Anthony et al., 2017; Schrittwieser et al., 2020; Dam et al., 2021), where deep neural networks are trained from targets generated by the MCTS iterations. The latter approach is also known as expert iteration (Anthony et al., 2017). AlphaZero and MuZero are prominent algorithms adopting this approach. The key distinction between expert iteration and PG-MCTL lies in their policy updates. While the PG-MCTL is weighting the experiences with the return g_t and importance weight ρ_t (see Eq. (11)), the standard expert iteration does not, assuming that all instances are positive examples since they are the result of MCTS iteration. Another difference pertains to the type of learning. Specifically, the expert iteration is classified as decision-time planning or model-based RL. In contrast, PG-MCTL is model-free and doesn't rely on a simulator. Grill et al. (2020) also extend AlphaZero or MuZero with the notion of PG, which is also model-based RL.

Several studies, among others, combine PG and MCTS. Soemers et al. (2019) runs MCTS to compute a value function that PG uses. Guo et al. (2016) uses PG to design reward-bonus functions to improve the performance of MCTS. Anthony et al. (2019) uses PG for updating local policies and investigates

[3]The mixing probability function λ may be given as a hyper-parameter instead of parameterizing it with θ. In that case, the update λ of line 18 by Eq. (13) is skipped.

planning without an explicit tree search. Dieb et al. (2020) uses PG in the tree expansion phase to choose a promising child node to be created, assuming a situation where the number of actions is huge.

From another perspective, the PG-MCTL can be regarded as using MCTS for PG to enhance the efficiency of exploration. Most exploration approaches in PG focus on designing reward functions, often incorporating a bonus of intrinsic motivation or curiosity to explore unknown states (Bellemare et al., 2016; Tang et al., 2017; Zheng et al., 2018; Burda et al., 2019). Haarnoja et al. (2018) demonstrates remarkable success using an entropy bonus to aid exploration in benchmark control tasks. Unlike many approaches, PG-MCTL does not modify the objective function, however, it remains compatible with most of them.

For RL in an HDP or NMDP, there are two major directions. The first one assumes the existence of latent dynamics and considers the identification of the dynamics (Thiébaux et al., 2006; Poupart & Vlassis, 2008; Silver & Veness, 2010; Singh et al., 2012; Doshi-Velez et al., 2015; Brafman & Giacomo, 2019). A POMDP (Kaelbling et al., 1998) serves as a widely-recognized mathematical model for this purpose. Doshi-Velez et al. (2015) identifies an environment as a POMDP with Bayesian non-parametric methods and then compute a policy by solving the POMDP. The other direction is to use a function approximator whose output depends not only on a current observation but also on past observations (Loch & Singh, 1998; Hernandez-Gardiol & Mahadevan, 2000; Bakker, 2002; Hausknecht & Stone, 2015; Rennie et al., 2017; Qin et al., 2023). One of the successful approaches uses a neural network for sequence learning as a policy model and optimizes it by PG (Wierstra et al., 2010; Rennie et al., 2017; Paulus et al., 2018; Kamigaito et al., 2021), as corresponds to the PG policy in our proposed PG-MCTL.

While the proposed implementation of PG-MCTL integrates standard PG and MCTS algorithms in a well-designed way, there are a lot of studies on enhancing those algorithms, such as stabilization of PG by a conservative update (Kakade, 2002; Schulman et al., 2015; 2017), the entropy regularization for explicitly controlling the exploration-exploitation trade-off (Haarnoja et al., 2017; 2018; Xiao et al., 2019; Grill et al., 2020), and extensions of MCTS to continuous spaces (Couëtoux et al., 2011; Mansley et al., 2011; Kim et al., 2020; Mao et al., 2020). Incorporating these technologies, including the expert iteration, into the PG-MCTL is an interesting avenue for future work.

5 Numerical Experiments

We apply the PG-MCTL algorithm to two different tasks in HDPs. The first task is a randomly synthesizing task, which does not contain domain-specific structures and is not overly complex. Therefore, this task will help investigate the primary performance of algorithms. The second task is the long-term dependency T-maze, which is known as a standard benchmark for learning a deep-memory POMDP (Bakker, 2002; Wierstra et al., 2010). Details of the experimental setup are given in Appendix B.

The goal here is not to nd the best model for the above two tasks, but to investigate if/how combining the PG and MCTL (MCTS variant) by the PG-MCTL is effective. Therefore, the applied algorithms here are simple, not state-of-the-art algorithms. In this regard, model-based RLs including MuZero (Schrittwieser et al., 2020) are also out of the scope of this work. We used REINFORCE with a baseline (Williams, 1992) for the PG and MCTL for the original MCTS, which are introduced in Sections 2.2 and 3.2, respectively. Note that REINFORCE, although it is classic, is still appealing due to its good empirical performance and simplicity (Grooten et al., 2022; Zhang et al., 2021). It and its variants are used in many applications (Rennie et al., 2017; Paulus et al., 2018; Chen et al., 2019; Xia et al., 2020; Wang et al., 2021; Liu et al., 2022). Thus, we believe that improving REINFORCE itself is still important in the practical implementation of RL.

While our focus is on fundamental algorithms, we also included the proximal policy optimization (PPO) (Schulman et al., 2017), a modern and practical variation of the PG algorithm, to provide a comparative perspective on performance. Additionally, we utilized a simple version of AlphaZero

(Silver et al., 2017b), termed lazy AlphaZero. It employs the same adaptation to the online RL setting as MCTL (also called lazy MCTS). The parameterized policy model, serving as the prior policy in lazy AlphaZero, is updated based on online experiences according to likelihood maximization. Furthermore, we also implemented a naive mixture of the PG and MCTL that follows Eq. (5) but uses the learning rules of the standalone REINFORCE and MCTL. For fair evaluation, we first tuned the hyper-parameters of standalone algorithms and then used them for the PG-MCTL and the naive mixture model.

5.1 Randomly synthesized task

This task is a randomly synthesized non-Markovian model. It is analogous to generation tasks such as text generation and compound synthesis, presenting as a simple yet challenging HDP. There are five observations and ten actions. The observation probability function p_{o} was synthesized to depend on the time-step, observation, and action. The reward function f_{r} was composed of the sum of the per-step sub-reward function r_{local} and the history-based sub-reward function r_{global}. The function r_{global} was synthesized by using a Gaussian process, such that the more similar the histories, the closer their rewards tend to be. This reward function f_{r} can be interpreted in the context of text generation as follows: r_{local} represents the quality of local word connections, and r_{global} represents the quality of the generated text. The policy π^{θ} was a softmax and parameterized by using the reward structure. We set $T = 15$. Thus, there are an enormous number of variations in the histories ($\sim 10^{25}$).

Figure 2 (a) shows the results of ten independent runs, where 'PG-MCTL' and 'PG-MCTL-adpt' are the proposed methods. PG-MCTL uses a fixed mixing probability λ_n, while PG-MCTL-adpt learns λ_θ with the PG update of (13). In each run, an HDP environment was independently generated, as described above.

The results indicate that REINFORCE learned most quickly but often fell into sub-optimal policies. In contrast, the MCTL and lazy AlphaZero methods continued to improve but was slow to learn. This slowness is probably due to the lack of the ability to learn state representation, more concretely, if trajectories are even slightly different, the nodes differ from each other, and thus information is not shared among them. Whereas, the proposed PG-MCTL methods were able to continuously improve and was not slow to learn. It implies that the PG-MCTL approach can successfully incorporate the advantages of both the PG and MCTL. Specifically, while PG offers rapid learning, MCTL provides continuous improvement without easily falling into sub-optimal policies.

However, it is worth noting that the poor performance of the naive mixture indicates that the simple mixing approach of the PG and MCTL policies cannot work. Also, note that their performances of PG-MCTL and PG-MCTL-adpt were similar. This similarity suggests that in this task, learning the mixing probability λ did not provide a significant advantage over the fixed probability approach in PG-MCTL. However, it is important to consider the potential benefits of adaptive methods in more structured tasks, as explored in Section 5.2.

5.2 T-maze task

The T-maze task of Figure 3 is non-Markovian and designed to test the ability to identify associations between events with long-time lags (Bakker, 2002; Wierstra et al., 2010). An agent has to remember an observation made at the first time step until the episode ends. We use a long short-term memory (LSTM) as a policy model. Since the original setting with the initial position $s_0 = 0$ is not so difficult, we prepared a more difficult setting, where initial position s_0 is the center of the corridor. In this setting, a policy that chooses the left or right action with probability 0.5 can be sub-optimal. Note that this is not true in the original setting ($s_0 = 0$) because going left occurs a negative reward.

The results are shown in Figure 2 (b) and indicate the effectivity of the proposed methods. In this experiment, the performance of PPO and REINFORCE showed only marginal differences, suggesting

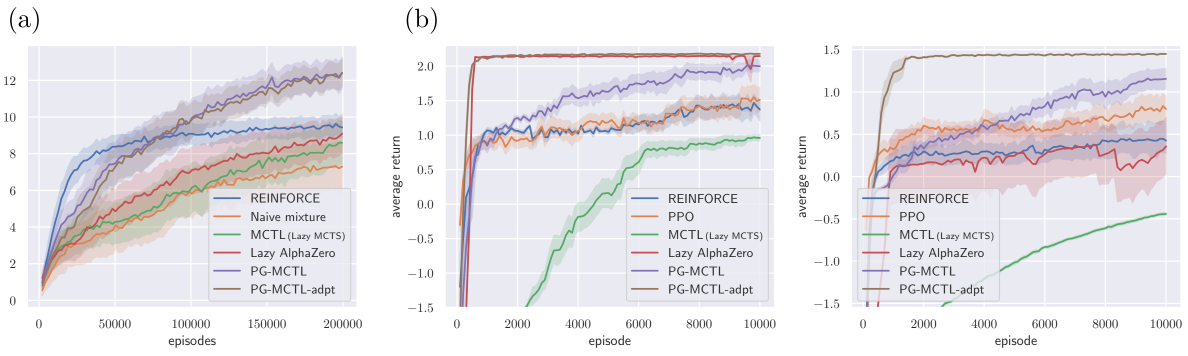

Figure 2: Performance comparison by ten independent runs, where the error bar represents the standard error of the mean: (a) the randomly synthesized task ($T = 15$). (b) T-maze task; the plot on the left is the result of an easy setting (the length of corridor $L = 30$ and the initial position $s_0 = 0$). The plot on the right is for a more difficult setting, where there exist more sub-optimal policies (the length of corridor $L = 100$ and the initial position $s_0 = 50$).

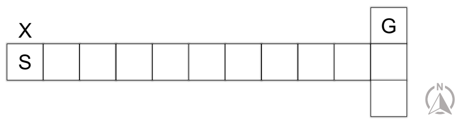

Figure 3: Long-term dependency T-maze task: an agent starts at the position S. Only at the initial time step $t = 0$, it can observe a signal 'up' or 'down' that indicates it should go north or south at the T-junction in this episode.

that using PPO as the PG module in PG-MCTL would offer limited benefits. The significant superiority of our approach over PPO, however, underscores the effectiveness of combining PG with MCTS. Moreover, PG-MCTL-adpt outperformed PG-MCTL, demonstrating that adjusting the mixing probability through learning, especially at T-junctions, was effective in this task.

6 Conclusion

This paper focused on online reinforcement learning problems in history-based decision processes (HDPs). We investigated the PG-MCTL approach, a mixture policy approach for the PG and online MCTS variant (MCTL) that takes advantage of the features of the PG and MCTS algorithms. We provided the convergence analysis and then proposed an implementation that converges to a reasonable solution. Through the numerical experiments on two HDP tasks with different characteristics, we confirmed the significant effect of the proposed approach for the mixture of the PG and MCTL policies.

In future work, we will apply our algorithms with state-of-the-art neural networks for sequence data to more practical and challenging domains, such as advertising text generation and incomplete information games. Also, the analysis of convergence points is crucial because the PG is a local optimization while MCTS is a global optimization method. For example, guidance from MCTS may help PGs overcome a bad local optimum and a learning plateau. Another exciting direction will incorporate state-of-the-art PG and MCTS techniques, such as entropy regularization and the natural gradient.

References

D. Aberdeen. *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, Australian National University, 2003.

T. Anthony, Z. Tian, and D. Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, 2017.

T. Anthony, R. Nishihara, P. Moritz, T. Salimans, and J. Schulman. Policy gradient search: Online planning and expert iteration without search trees. In *arXiv preprint arXiv:1904.03646*, 2019.

F. Bacchus, C. Boutilier, and A. Grove. Rewarding behaviors. In *National Conference on Artificial Intelligence*, volume 2, pp. 11601167. AAAI Press, 1996.

F. Bacchus, C. Boutilier, and A. Grove. Structured solution methods for non-Markovian decision processes. In *National Conference on Artificial Intelligence*, pp. 112117. AAAI Press, 1997.

B. Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems*, 2002.

J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

J. Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. In *International Journal of Robotics Research*, pp. 1263–1278, 2012.

D. P. Bertsekas. *Dynamic Programming and Optimal Control, Volumes 1 and 2.* Athena Scientific, 1995.

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

V. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint.* Cambridge University Press, 2008.

R. I. Brafman and G. D. Giacomo. Regular decision processes: A model for non-Markovian domains. In *International Joint Conference on Artificial Intelligence*, pp. 5516–5522, 2019.

C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. Chi. Top-k off-policy correction for a reinforce recommender system. In *International Conference on Web Search and Data Mining*, 2019.

L. Choshen, L. Fox, Z. Aizenbud, and O. Abend. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*, 2020.

K. Ciosek and S. Whiteson. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21:1–51, 2020.

A. M. Clarke, J. Friedrich, E. M. Tartaglia, S. Marchesotti, W. Senn, and M. H. Herzog. Human and machine learning in non-Markovian decision making. *PLOS One*, 10(4):e0123105, 2015.

A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard. Continuous upper confidence trees. In *International Conference on Learning and Intelligent Optimization*, pp. 433445, 2011.

R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*, pp. 72–83, 2006.

T. Dam, C. D'Eramo, Jan Peters, and J. Pajarinen. Convex regularization in Monte-Carlo tree search. In *International Conference on Machine Learning*, pp. 2365–2375, 2021.

S. Dieb, Z. Song, W. J. Yin, and M. Ishii. Optimization of depth-graded multilayer structure for X-ray optics using machine learning. *Journal of Applied Physics*, 128(7):074901, 2020.

F. Doshi-Velez, D. Pfau, F. Wood, and N. Roy. Bayesian nonparametric methods for partially-observable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):394–407, 2015.

Y. Efroni, G. Dalal, B. Scherrer, and S. Mannor. How to combine tree-search methods in reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 3494–3501, 2019.

J. Friedrich, R. Urbanczik, and W. Senn. Spatio-temporal credit assignment in neuronal population learning. *PLOS Computational Biology*, 7(6):e1002092, 2011.

J. B. Grill, F. Altché, Y. Tang, T. Hubert, M. Valko, I. Antonoglou, and R. Munos. Monte-Carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pp. 3769–3778, 2020.

B. Grooten, J. Wemmenhove, M. Poot, and J. Portegies. Is vanilla policy gradient overlooked? analyzing deep reinforcement learning for hanabi. In *Adaptive and Learning Agents Workshop at AAMAS*, 2022.

V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3(6):671–692, 1990.

X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time atari game play using offline Monte-Carlo tree search planning. In *Advances in Neural Information Processing Systems*, 2014.

X. Guo, S. Singh, R. Lewis, and H. Lee. Deep learning for reward design to improve Monte Carlo tree search in ATARI games. In *International Joint Conference on Artificial Intelligence*, 2016.

T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, volume 70, pp. 1352–1361, 2017.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1856–1865, 2018.

M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Conference on Artificial Intelligence*, 2015.

N. Hernandez-Gardiol and S. Mahadevan. Hierarchical memory-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2000.

D. R. Jiang, E. Ekwedike, and H. Liu. Feedback-based tree search for reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pp. 2284–2293, 2018.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

S. M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

H. Kamigaito, P. Zhang, H. Takamura, and M. Okumura. An empirical study of generating texts for search engine advertising. In *Conference of the North American Chapter of the Association for Computational Linguistics: Industry Papers*, pp. 255–262, 2021.

S. Kiegeland and J. Kreutzer. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

B. Kim, K. Lee, S. Lim, L. P. Kaelbling, and T. Lozano-Pérez. Monte Carlo tree search in continuous spaces using Voronoi optimistic optimization with regret bounds. In *AAAI Conference on Artificial Intelligence*, 2020.

H. Kimura, K. Miyazaki, and S. Kobayashi. Reinforcement learning in POMDPs with function approximation. In *International Conference on Machine Learning*, 1997.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pp. 282–293, 2006.

L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms, 2nd Edition.* John Wiley and Sons, 2014.

T. Lattimore and C. Szepesvári. *Bandit Algorithms.* Cambridge University Press, 2020.

L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Knows what it knows: A framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.

X. Liu, W. Lei, J. Lv, and J. Zhou. Abstract rule learning for paraphrase generation. In *International Joint Conference on Artificial Intelligence*, 2022.

J. Loch and S. Singh. Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *International Conference on Machine Learning*, 1998.

S. J. Majeed and M. Hutter. On Q-learning convergence for non-Markov decision processes. In *International Joint Conference on Artificial Intelligence*, pp. 2546–2552, 2018.

C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *International Conference on Automated Planning and Scheduling*, 2011.

W. Mao, K. Zhang, Q. Xie, and T. Başar. POLY-HOOT: Monte-Carlo planning in continuous space mdps with non-asymptotic analysis. In *Advances in Neural Information Processing Systems*, 2020.

T. Morimura, T. Osogami, and T. Shirai. Mixing-time regularized policy gradient. In *AAAI Conference on Artificial Intelligence*, 2014.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

P. Poupart and N. Vlassis. Model-based Bayesian reinforcement learning in partially observable domains. In *International Symposium on Artificial Intelligence and Mathematics*, 2008.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, 1994.

A. Qin, F. Gao, Q. Li, S.-C. Zhu, and S. Xie. Learning non-markovian decision-making from state-only sequences. In *Advances in Neural Information Processing Systems*, 2023.

S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195, 2017.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, go, chess and shogi by planning with a learned model. In *Nature*, volume 588, 2020.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, volume 23, pp. 2164–2172, 2010.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

D. Silver, T Hubert, J. Schrittwieser, I Antonoglou, M. Lai, A Guez, M. Lanctot, L Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017a.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017b.

S. Singh, M. James, and M. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

D. J. N. J. Soemers, É. Piette, M. Stephenson, and C. Browne. Learning policies from self-play with policy gradients and MCTS value estimates. In *IEEE Conference on Games*, 2019.

E. J. Sondik. *The optimal control of partially observable Markov processes.* PhD thesis, Stanford University, 1971.

R. S. Sutton and A. G. Barto. *Reinforcement Learning.* MIT Press, 2nd edition, 2018.

M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. In *arXiv preprint arXiv:2103.04931*, 2021.

H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y.Duan, J. Schulman, F. DeTurck, and P. Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2750–2759, 2017.

S. Thiébaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza. Decision-theoretic planning with non-markovian rewards. *Journal of Articial Intelligence Research*, 25:17–74, 2006.

T. Vodopivec, S. Samothrakis, and B. Šter. On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936, 2017.

X. Wang, Y. Du, S. Zhu, L. Ke, Z. Chen, J. Hao, and J. Wang. Ordering-based causal discovery with reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2021.

S. D. Whitehead and L. J. Lin. Reinforcement learning of non-Markov decision processes. *Artificial Intelligence*, 73(1-2):271–306, 1995.

D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634, 2010.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Y. Xia, J. Zhou, Z. Shi, C. Lu, and H. Huang. Generative adversarial regularized mutual information policy gradient framework for automatic diagnosis. In *AAAI Conference on Artificial Intelligence*, 2020.

C. Xiao, R. Huang, J. Mei, D. Schuurmans, and M. Müller. Maximum entropy Monte-Carlo planning. In *Advances in Neural Information Processing Systems*, 2019.

J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, 2018.

S. Young, M. Gašić, B. Thomson, and J. D. Williams. POMDP-based statistical spoken dialog systems: A review. In *Proceedings of the IEEE*, volume 101, pp. 1160–1179. IEEE, 2013.

L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence*, 2017.

T. Yu, B. Zhou, K. W. Chan, L. Chen, and B. Yang. Stochastic optimal relaxed automatic generation control in non-Markov environment based on multi-step Q(*lambda*) learning. *IEEE Transactions on Power Systems*, 26(3):1272 – 1282, 2011.

J. Zhang, J. Kim, B. O'Donoghue, and S. Boyd. Sample efficient reinforcement learning with reinforce. In *AAAI Conference on Artificial Intelligence*, 2021.

Z. Zheng, J. Oh, and Singh S. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Y. Zhou, C. Xiong, R. Socher, and R. Socher. Improving end-to-end speech recognition with policy learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

A Proofs

A.1 Preliminaries

We first introduce the basic results of the ordinary differential equation (ODE) based approach for the stochastic approximation (Borkar, 2008). We consider the following update rule of $\theta \in \mathbb{R}^d$ with an initial value $\theta_0 \in \mathbb{R}^d$ for all $n \in [0, 1, \dots] = \mathbb{N}_{\geq 0}$,

$$\theta_{n+1} = \theta_n + \alpha_n [\, k(\theta_n) + M_{n+1} + \epsilon_n]. \tag{19}$$

To take the ODE approach, we extend the above discrete-time stochastic process of θ_n to a continuous, piecewise-linear counterpart $\bar{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ as follows: Define a time-instant function $t : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such as

$$t(n) \triangleq \begin{cases} 0 & \text{if } \; n = 0, \\ \sum_{m=0}^{n-1} \alpha_m & \text{otherwise,} \end{cases}$$

and set $\bar{\theta}(t(n)) := \theta_n,\ \forall n \in \mathbb{N}_{\geq 0}$. Then, for any $n \in \mathbb{N}_{\geq 0}$, we derive the following linear interpolation,

$$\bar{\theta}(\tau) \triangleq \theta_n + (\theta_{n+1} - \theta_n)\frac{\tau - t(n)}{t(n+1) - t(n)}, \quad \tau \in I_n, \tag{20}$$

where $I_n \triangleq [t(n), t(n+1)]$. As we will show later, the key result of the ODE approach to the analysis of Eq. (19) is that $\bar{\theta}(\tau)$ asymptotically almost surely approaches the solution set of the following ODE,

$$\dot{\theta}(\tau) = k(\theta(\tau)), \quad \tau \in \mathbb{R}_{\geq 0}. \tag{21}$$

For this purpose, we need to make the following assumptions.

Assumption 6. *The learning rates $\{\alpha_n\}$ are positive scalars satisfying*

$$\lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n = \infty, \qquad \lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n^2 \leq \infty. \tag{22}$$

Assumption 7. *The function $k : \mathbb{R}^d \to \mathbb{R}^d$ is a Lipschitz continuous map, i.e., for some constant $0 < L < \infty$,*

$$\|k(\theta) - k(\theta')\| \leq L\|\theta - \theta'\|, \qquad \forall(\theta, \theta') \in \mathbb{R}^d \times \mathbb{R}^d.$$

Assumption 8. *The stochastic series $\{M_n\}$ is a martingale difference sequence with respect to the increasing family of σ-fields*

$$\mathcal{F}_n \triangleq \sigma(\theta_i, M_i, \epsilon_i, i \leq n).$$

That is, the following holds,

$$\mathbb{E}[M_{n+1} \,|\, \mathcal{F}_n] = 0 \quad a.s., \quad \forall n \geq 0.$$

Furthermore, M_n is always square-integrable with

$$\mathbb{E}[\,\|M_{n+1}\|^2 \,|\, \mathcal{F}_n] = K(1 + \|\theta_n\|^2) \quad a.s., \quad \forall n \geq 0, \tag{23}$$

for some constant $K \geq 0$.

Assumption 9. *The series of bias $\{\epsilon_n\}$ is a deterministic or random bounded sequence which is $o(1)$.*

Assumption 10. *The updates of Eq. (19) remain bounded almost surely, i.e.,*

$$\sup_n \|\theta_n\| < \infty, \quad a.s.$$

Lemma 1. *Assume Assumptions 6–10 hold. Let $\theta^s(\tau)$, $\tau \geq s \geq 0$, denote the trajectory of Eq. (21) starting at time $s \in \mathbb{R}_{\geq 0}$:*

$$\dot{\theta}^s(\tau) = k(\theta^s(\tau)), \quad \forall \tau \in \mathbb{R}_{\geq s},$$

with $\theta^s(s) = \bar{\theta}(s)$. Then, for any $\nu > 0$, the following holds almost surely,

$$\lim_{s\to\infty} \sup_{\tau\in[s,s+\nu]} \|\bar{\theta}(\tau) - \theta^s(\tau)\| = 0,$$
$$\lim_{s\to\infty} \sup_{\tau\in[s-\nu,s]} \|\bar{\theta}(\tau) - \theta^s(\tau)\| = 0.$$

Proof: This lemma is a simple extension of Lemma 1 in Section 2 of (Borkar, 2008) with a bias term ϵ_n, and this proof mostly follows from it. We will only prove the first claim since the same applies to the proof of the second claim.

For $n \in \mathbb{N}_{\geq 0}$ and $m \in \mathbb{N}_{\geq 1}$, by the construction, $\bar{\theta}$ can be written down as follows:

$$\bar{\theta}(t(n+m)) = \bar{\theta}(t(n)) + \sum_{i=0}^{m-1} \alpha_{n+i}\, k(\bar{\theta}(t(n+i))) + \delta_{n,n+m}. \tag{24}$$

where

$$\delta_{n,n+m} \triangleq \xi_{n+m} - \xi_n + \sum_{i=0}^{m-1} \alpha_{n+i}\, \epsilon_{n+i},$$
$$\xi_n \triangleq \begin{cases} 0, & \text{if } \ n = 0, \\ \sum_{i=0}^{n-1} \alpha_i M_{i+1}, & \text{if } \ n \in \mathbb{N}_{\geq 1}. \end{cases}$$

We will show $\sup_{m \geq 0} \|\delta_{n,n+m}\| = 0$ as $n \to \infty$. By Assumptions 8 and 10, the series $\{\xi_n\}$ is a zero mean, square-integrable martingale with respect to the σ-fields $\mathcal{F}_n$. Furthermore, by Assumptions 6, 8, and 10, we have

$$\sum_{n=0}^{\infty} \mathbb{E}[\, \|\xi_{n+1} - \xi_n\|^2 \,|\, \mathcal{F}_n] = \sum_{n=0}^{\infty} \alpha_n^2\, \mathbb{E}[\, \|M_{n+1}\|^2 \,|\, \mathcal{F}_n] < \infty, \quad \text{a.s.}$$

From the above and the martingale convergence theorem (Theorem 11 of Appendix in (Borkar, 2008)), it can be said that $\{\xi_n\}$ converges. The third term of $\delta_{n,n+m}$ also converges to zero as $n \to \infty$ because $\{\epsilon_n\}$ is $o(1)$ by Assumption 9. Thus, the following holds,

$$\lim_{n \to \infty} \|\delta_{n,n+m}\| = 0, \quad \text{a.s.} \tag{25}$$

Next, we will look into θ^s. It can be written down as follows:

$$\begin{aligned} \theta^{t(n)}(t(n+m)) &= \bar{\theta}(t(n)) + \int_{t(n)}^{t(n+m)} k(\theta^{t(n)}(\tau)) d\tau \\ &= \bar{\theta}(t(n)) + \sum_{i=0}^{m-1} \alpha_{n+i}\, k(x^{t(n)}(t(n+i))) + \int_{\tau=t(n)}^{t(n+m)} \big(k(\theta^{t(n)}(\tau)) - k(\theta^{t(n)}(\tilde{\tau}))\big) d\tau, \end{aligned} \tag{26}$$

where

$$\tilde{\tau} \triangleq \max\{t(n) \,|\, t(n) \leq \tau,\, n \in \mathbb{N}_{\geq 0}\}.$$

We investigate the integral on the right-hand side in Eq. (26). Let $C_0 \triangleq \sup_n \|\theta_n\|$. Note that $C_0 < \infty$ a.s. by Assumption 10. By Assumption 7, $\|k(\theta) - k(0)\| \leq L\|\theta\|$, and so

$$\|k(\theta)\| \leq \|k(0)\| + L\|\theta\|. \tag{27}$$

Therefore, the following holds, for $\tau \in [s, s+\nu]$,

$$\begin{aligned} \|\theta^s(\tau)\| &\leq \|\bar{\theta}(s)\| + \int_{x=s}^{\tau} \big(\|k(0)\| + L\|\theta^s(x)\|\big) dx \\ &\leq C_0 + \|k(0)\|\nu + L \int_{x=s}^{\tau} \|\theta^s(x)\| dx. \end{aligned}$$

By Gronwall's inequality (Lemma 6 of Appendix in (Borkar, 2008)), we obtain

$$\|\theta^s(\tau)\| \leq \big(C_0 + \|k(0)\|\nu\big) \exp(L\nu), \quad \forall \tau \in [s, s+\nu]. \tag{28}$$

Thus, from Eq. (27), we have the following bound,

$$C_\nu \triangleq \|k(0)\| + L(C_0 + \|k(0)\|\nu)\exp(L\nu) < \infty, \quad \text{a.s.}$$

such that, for all $\tau \in [s, s+\nu]$,

$$\|k(\theta^s(\tau))\| \leq C_\nu. \tag{29}$$

Here we assume ν is larger than $t(n+m) - t(n)$ without loss of generality. For $i \in \{0, \dots, m-1\}$ and $\tau \in [t(n+i), t(n+i+1)]$, the bound C_t gives

$$\begin{aligned}\|\theta^{t(n)}(\tau) - \theta^{t(n)}(t(n+i))\| &\leq \left\| \int_{\iota=t(n+i)}^{\tau} k(\theta^{t(n)}(\iota))d\iota \right\| \\ &\leq C_\nu(\tau - t(n+i)) \\ &\leq C_\nu \alpha_{n+i}.\end{aligned}$$

The inequality gives the bound of the integral in Eq. (26) as follows: because

$$\begin{aligned}\left\| \int_{\tau=t(n)}^{t(n+m)} \big(k(\theta^{t(n)}(\tau)) - k(\theta^{t(n)}(\tilde{\tau}))\big) d\tau \right\| &\leq \int_{\tau=t(n)}^{t(n+m)} L\|\theta^{t(n)}(\tau) - \theta^{t(n)}(\tilde{\tau})\| \\ &= L \sum_{i=0}^{m-1} \int_{\tau=t(n+i)}^{t(n+i+1)} \|\theta^{t(n)}(\tau) - \theta^{t(n)}(t(n+i))\| d\tau \\ &\leq C_\nu L \sum_{i=0}^{m-1} \alpha_{n+i}^2\end{aligned}$$

Thus, by Assumption 6, we have

$$\lim_{n\to\infty} \left\| \int_{\tau=t(n)}^{t(n+m)} \big(k(\theta^{t(n)}(\tau)) - k(\theta^{t(n)}(\tilde{\tau}))\big) d\tau \right\| \leq C_\nu L \sum_{i=0}^{m-1} \lim_{n\to\infty} \alpha_{n+i}^2 = 0, \quad \text{a.s.} \tag{30}$$

By subtracting Eq. (24) from Eq. (26) and taking a norm, we have

$$\|\bar{\theta}(t(n+m)) - \theta^{t(n)}(t(n+m))\| \leq L \sum_{i=0}^{m-1} \alpha_{n+i} \|\bar{\theta}(t(n+i)) - \theta^{t(n)}(t(n+i))\| + K_{n,\nu},$$

where

$$\begin{aligned}K_{n,\nu} &\triangleq C_\nu L \sum_{i\geq 0}^{\acute{m}_{n,\nu}-1} \alpha_{n+i}^2 + \|\delta_{n,n+\acute{m}_{n,\nu}}\|, \\ \acute{m}_{n,\nu} &\triangleq \max\{m \,|\, t(n+m) - t(n) \leq \nu,\ m \in \mathbb{N}_{\geq 0}\}.\end{aligned}$$

Note that

$$\lim_{n\to\infty} K_{n,\nu} = 0, \quad \text{a.s.} \tag{31}$$

holds by Eqs. (25) and (30). By applying the discrete Gronwall lemma (Lemma 8 of Appendix in (Borkar, 2008)) to the above inequality, we have

$$\sup_{i\in\{0,\dots,m\}} \|\bar{\theta}(t(n+i)) - \theta^{t(n)}(t(n+i))\| \leq K_{n,\nu} \exp(L\nu), \quad \text{a.s.} \tag{32}$$

Let $\tau \in [t(n+i), t(n+i+1)]$ for $0 \leq i \leq m-1$. Then we have

$$\bar{\theta}(\tau) = \lambda\theta(t(n+i)) + (1-\lambda)\bar{\theta}(t(n+i+1))$$

for some $\lambda \in [0,1]$, and thus the following inequality is obtained,

$$\begin{aligned}
\|\bar{\theta}(\tau) - \theta^{t(n)}(\tau)\| &= \|\lambda(\bar{\theta}(t(n+i)) - \theta^{t(n)}(\tau)) + (1+\lambda)(\bar{\theta}(t(n+i+1)) - \theta^{t(n)}(\tau))\| \\
&\leq \lambda \left\| \bar{\theta}(t(n+i)) - \theta^{t(n)}(t(n+i)) - \int_{\iota=t(n+i)}^{\tau} k(\theta^{t(n)}) d\iota \right\| \\
&\quad + (1-\lambda) \left\| \bar{\theta}(t(n+i+1)) - \theta^{t(n)}(t(n+i+1)) + \int_{\iota=\tau}^{t(n+i+1)} k(\theta^{t(n)}) d\iota \right\| \\
&\leq \lambda \|\bar{\theta}(t(n+i)) - \theta^{t(n)}(t(n+i))\| + (1-\lambda)\|\bar{\theta}(t(n+i+1)) - \theta^{t(n)}(t(n+i+1))\| \\
&\quad + \int_{\iota=t(n+i)}^{t(n+i+1)} \|k(\theta^{t(n)}(\iota))\| d\iota \\
&\leq K_{n,\nu} \exp(L\nu) + C_\nu \alpha_{n+i}, \quad \text{a.s.},
\end{aligned}$$

where the last inequality is derived by using Eqs. (32) and (29). The above inequality is easily generalized to, with some constant $C \geq 0$

$$\sup_{\tau \in [s, s+\nu]} \|\bar{\theta}(\tau) - \theta^{t(n)}(\tau)\| \leq C K_{\tilde{s},\nu} \exp(L\nu) + C_\nu \alpha_{\tilde{s}},$$

where $\tilde{s} \triangleq \max\{t(n) \mid t(n) \leq s,\, n \in \mathbb{N}_{\geq 0}\}$. As $s \to \infty$, we have the first claim in this lemma. □

By applying Lemma 1 to Theorem 2 of Section 2 and Theorem 2 of Section 6 in (Borkar, 2008), we instantly obtain the following lemmas.

Lemma 2. *Assume Assumptions 6–10 hold. Then, the sequence $\{\theta_n\}$ generated by Eq. (19) almost surely converges to a (possibly sample path dependent) compact connected internally chain transitive invariant set of Eq. (23).*

Lemma 3. *Let the sequence $\{(\theta_n, \omega_n)\}$ is generated by*

$$\theta_{n+1} = \theta_n + \alpha_n[\, k(\theta_n, \omega_n) + M_{n+1}^{(1)} + \epsilon_n^{(1)}], \tag{6}$$

$$\omega_{n+1} = \omega_n + \eta_n[\, l(\theta_n, \omega_n) + M_{n+1}^{(2)} + \epsilon_n^{(2)}], \tag{7}$$

where $k : \mathbb{R}^d \times \mathbb{R}^e \to \mathbb{R}^d$ and $l : \mathbb{R}^d \times \mathbb{R}^e \to \mathbb{R}^e$ are the expected update functions, $M^{(1)} \in \mathbb{R}^d$ and $M^{(2)} \in \mathbb{R}^e$ are noise terms, and $\epsilon^{(1)} \in \mathbb{R}^d$ and $\epsilon^{(2)} \in \mathbb{R}^e$ are bias terms. Also, let the learning rates α_n and η_n of Eqs. (6) and (7) satisfying

$$\begin{cases}
\lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n = \lim_{N\to\infty} \sum_{n=0}^{N} \eta_n = \infty, \\
\lim_{N\to\infty} \sum_{n=0}^{N} \left(\alpha_n^2 + \eta_n^2\right) < \infty, \\
\lim_{N\to\infty} \dfrac{\alpha_N}{\eta_N} = 0.
\end{cases}$$

Assume Assumptions Assumptions 1–5 hold. Then, the sequence $\{(\theta_n, \omega_n)\}$ almost surely converges to a (possibly sample path dependent) compact connected internally chain transitive invariant set A of the following ODEs

$$\dot{\theta}(\tau) = k(\theta(\tau), \omega(\tau)), \tag{8}$$

$$\dot{\omega}(\tau) = l(\theta(\tau), \omega(\tau)). \tag{9}$$

Any pair $(\theta, \omega) \in A$ has the relation $\omega = \varphi(\theta)$, where $\varphi(\theta)$ is defined in Assumption 4 and denotes the globally asymptotically stable equilibrium of the ODE (9) of ω given θ.

A.2 Propositions 1 and 2

By applying Lemma 3 to the update rule of the proposed PG-MCTL algorithm (Eqs. (6) and (7)), we immediately obtain Propositions 1 and 2.

Proposition 1. *Assume Assumptions 1–5 hold. Let the mixing probability function $\lambda_n : \mathcal{H} \to [0,1]$ be invariant to the number of episodes n and the learning rates α_n and η_n satisfying*

$$\begin{cases} \lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n = \lim_{N\to\infty} \sum_{n=0}^{N} \eta_n = \infty, \\ \lim_{N\to\infty} \sum_{n=0}^{N} \left(\alpha_n^2 + \eta_n^2\right) < \infty, \\ \lim_{N\to\infty} \frac{\alpha_N}{\eta_N} = 0. \end{cases} \tag{13}$$

Then, almost surely, the sequence $\{(\theta_n, \omega_n)\}$ generated by Eqs. (6) and (7) converges to a compact connected internally chain transitive invariant set of Eqs. (8) and (9).

Proposition 2. *Let $\lambda_n : \mathcal{H} \to [0,1]$ be a function parameterized by a part of θ (and be a Lipschitz continuous map with respect to its parameter). Assume that all the conditions of Proposition 1 are satisfied expect for λ_n. Still, the consequence of Proposition 1 holds.*

A.3 Proposition 3

Proposition 3. *Assume Assumptions 1 and 2 only for $i = 1$ hold, k is Lipschitz continuous map, and $\sup_n(\|\theta_n\|) < \infty$ holds. Let the mixing probability λ_n be $o(1)$ and satisfy $0 \leq \lambda_n \leq 1 - \varepsilon$ for all n and a constant $\varepsilon > 0$, and the learning rate of the PG policy satisfy*

$$\lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n = \infty, \quad \lim_{N\to\infty} \sum_{n=0}^{N} \alpha_n^2 < \infty.$$

Then, almost surely, the sequence $\{\theta_n\}$ generated by Eqs. (6) and (7) converges to a compact connected internally chain transitive invariant set of the ODE, $\dot{\theta}(\tau) = \nabla_\theta \Upsilon(\pi^{\theta(\tau)})$.

Proof: The update rule of θ (Eq. (6)) is rewritten as

$$\begin{aligned} \theta_{n+1} &= \theta_n + \alpha_n \sum_{t=0}^{T} \nabla_\theta \log \pi^{\theta_n,\omega_n}(a_t|h_t)(g_t - b(h_t)) \\ &= \theta_n + \alpha_n \left(\sum_{t=0}^{T} \nabla_\theta \log \pi^{\theta_n}(a_t|h_t)(g_t - b(h_t)) - \lambda_t \sum_{t=0}^{T} \pi^{\omega_n}(a_t|h_t) \nabla_\theta \log \pi^{\theta_n}(a_t|h_t)(g_t - b(h_t)) \right) \end{aligned} \tag{33}$$

By the definition of the PG-MCTL policy (Eq. (5))

$$\pi^{\theta_n,\omega_n}(a|h) \triangleq (1 - \lambda_n)\pi^{\theta_n}(a|h) + \lambda_n \pi^{\omega_n}(a|h)$$

and the assumption of the proposition, $1 - \lambda_n \geq \varepsilon,\ n \geq 0$, the expected value of the second terms of the right side of Eq. (33) is

$$\begin{aligned} &\mathbb{E}^{\pi^{\theta_n,\omega_n}}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi^{\theta_n}(A_t|H_t)(G_t - b(H_t)) - \lambda_t \sum_{t=0}^{T} \pi^{\omega_n}(A_t|H_t) \nabla_\theta \log \pi^{\theta_n}(A_t|H_t)(G_t - b(H_t)) \right] \\ &\qquad = \varepsilon^T \mathbb{E}^{\pi^{\theta_n}}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi^\theta(A_t, H_t)\,(G_t - b(h_t)) \right] + \epsilon'_t, \end{aligned}$$

where the sequence $\{\epsilon'_n\}$ is $o(1)$. Thus, Eq. (33) can be rewritten as

$$\theta_{n+1} = \theta_n + \alpha_n(\tilde{k}(\theta_n) + M'_n + \epsilon'_n),$$

where $\tilde{k} : \mathbb{R}^d \to \mathbb{R}^d$ is the expected update function

$$\tilde{k}(\theta) \triangleq \varepsilon^T \Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi^\theta(A_t, H_t)\,(G_t - b(h_t)) \Bigg] = \varepsilon^T \nabla_\theta \Upsilon(\pi^{\theta(\tau)}),$$

and $\{M'_n\}$ is a zero mean, square-integrable martingale difference sequence with respect to $\mathcal{F}_n$.

From the above, we can apply Lemma 2 and so the claim follows. □

A.4 Theorem 1

Theorem 1. *Let the PG-MCTL update the parameterized policy π^θ by Eq. (11) and the MCTL policy π^ω by the rule in which κ in Eq. (14) is replaced by $\bar{\kappa}$ of Eq. (16), and the learning rates satisfy the conditions of Eq. (10). Also let π^θ be defined on a compact parameter space and have always bounded first and second partial derivatives, and π^ω be a softmax policy with hyper parameters $\beta \geq 0$ and $C \geq 0$ as*

$$\pi^\omega(a|h) \propto \exp\!\left(\beta \left\{ q(h,a) + C\sqrt{u(h,a) \log\!\left(\sum_b \frac{1}{u(h,b)} \right)} \right\}\right). \tag{17}$$

Then, $\lim_{n\to\infty} \nabla_\theta \Upsilon(\pi^{\theta_n,\omega_n}) = 0$ *holds.*

Proof: The proof consists of two major steps. First, we will show that the parameter $\{(\theta_n, \omega_n)\}$ converges to a compact connected internally chain transitive invariant set. Then we will prove that any element in that set satisfies the properties claimed in the theorem.

To apply Lemma 3, we will investigate whether the conditions of Lemma 3 are satisfied. By the construction of the sequence $\{\omega_n\}$ of the parameter of the MCTL policy,

$$\sup_n \|\omega_n\| < \infty$$

holds. It means that Assumption 5 holds, taking into account the condition $\sup_n \|\theta_n\| < \infty$, and also ensures that π^ω always has bounded first and second derivatives, as well as π^θ. In order to check Lipschitz continuity of the expected update functions $k(\theta,\omega)$ and $l(\theta,\omega)$ (Assumption 3), we define them as

$$k(\theta,\omega) \triangleq \mathbb{E}^{\pi^{\theta,\omega}} \Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi^{\theta,\omega}(A_t|H_t)(G_t - b(H_t)) \Bigg] = \nabla_\theta \Upsilon(\pi^{\theta,\omega}), \tag{34}$$

and

$$\begin{cases} [l(\theta,\omega)]_{u(h_t,a)} \triangleq -d^{\theta,\omega}(h_t,a) \dfrac{\bar{\kappa}_\omega(h_t,a)u(h_t,a)}{u(h_t,a)+1}, \ \forall (h_t,a) \in \mathcal{H}_t \times \mathcal{A}, \quad \forall t \in \{0,\dots,T\}, \\ [l(\theta,\omega)]_{q(h_t,a)} \triangleq d^{\theta,\omega}(h_t,a)\bar{\kappa}_\omega(h_t,a)\big(\mathbb{E}^{\pi^{\theta,\omega}}[G_t \,|\, H_t = h_t, A_t = a] - q(h_t,a)\big), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall (h_t,a) \in \mathcal{H}_t \times \mathcal{A}, \quad \forall t \in \{0,\dots,T\}, \end{cases} \tag{35}$$

where $[l(\theta,\omega)]_x$ denotes the output corresponding to the parameter x in ω, the function $d^{\theta,\omega}(h_t,a)$ is the experiencing probability of (h_t,a) under $\pi^{\theta,\omega}$,

$$d^{\theta,\omega}(h_t,a) \triangleq \Pr(H_t = h_t, A_t = a \,|\, \mathsf{HDP}, \pi^{\theta,\omega}),$$

and $\bar{\kappa}_\omega$ is the counterpart to $\bar{\kappa}_{n,t}$ defined in Eq. (16). Thus, with the above properties and the boundedness of the reward function, we can see that k and l are Lipschitz continuous maps, i.e., Assumption 3 holds. The above observations also indicate that Assumptions 1 and 2 hold. Furthermore, since the second term in the MCTL policy (Eq. (17)) is asymptotically negligible, our task is episodic, and θ is basically updated with a naive Monte Carlo method, the ODE corresponding to l has a globally asymptotically stable equilibrium $\varphi(\theta)$, which will depend on θ, i.e., Assumption 4 holds. From the above results, Lemma 3 can be applied to this setup. Thus, it is proven that $\{(\theta_n, \omega_n)\}$ converges to a compact connected internally chain transitive invariant set $\mathcal{S}$ of the ODEs corresponding to Eqs. (34) and (35), and $\omega = \varphi(\theta)$ holds for all $(\theta, \omega) \in \mathcal{S}$.

With the above observations, we can instantly prove $\lim_{n\to\infty} \nabla_\theta \Upsilon(\pi^{\theta_n,\omega_n}) = 0$ by contradiction. (This is because θ_n converges to a compact connected internally chain transitive invariant set of the ODE $\nabla_\theta \Upsilon(\pi^{\theta,\omega})$ and $\Upsilon(\pi^{\theta,\omega})$ is bounded by the HDP definition.)

□

A.5 Equivalence of the standard MCTS update and Eq. (14)

By construction of $u(h,a)$ in Eq. (14), the initial value u is 1 for all (h,a). If u is updated once or more than once at (h,a), $u(h,a)$ is equal to or less than 0.5. Thus, by the definition of the tree-inclusion probability $p_{n,t}$ in Eq. (15), if $u(h_{t-1}, a_{t-1})$ has been updated even once in past episodes, the tree-inclusion probability $p_{n,t}(h_t, a_t)$ is one, otherwise it is zero. It indicates that in addition to the case $t = 0$, as long as a node corresponding to (h_{t-1}, a_{t-1}), $t \in \mathbb{N}_{\geq 1}$, would be included in a tree if the standard MCTS update were used, the tree-inclusion probability $p_{n,t}(h_t, a_t)$ is 1, and thus u and q of (h_t, a_t) will be updated with probability 1. Otherwise, Eq. (14) will not change u and q of (h_t, a_t) at all. The above is the same as the standard MCTS update.

All that remains is to show that the update rule in Eq. (14) can be derived from the standard MCTS update rule in Eq. (4) when $p_{n,t} = 1$. Because of $\eta_n \triangleq 1/n$ and $\kappa_{n,t} \triangleq p_{n,t} u_n(h_t, a_t)/\eta_n = n u_n(h_t, a_t)$, the update of m in Eq. (4) can be transformed as

$$
\begin{aligned}
& m_{n+1}(h_t, a_t) = m_n(h_t, a_t) + 1 \\
\Leftrightarrow \quad & \frac{1}{u_{n+1}(h_t, a_t)} = \frac{1}{u_n(h_t, a_t)} + 1 \\
\Leftrightarrow \quad & u_{n+1}(h_t, a_t) = \frac{u_n(h_t, a_t)}{1 + u_n(h_t, a_t)} \\
& = \frac{u_n(h_t, a_t)(1 + u_n(h_t, a_t)) - u_n(h_t, a_t)^2}{1 + u_n(h_t, a_t)} \\
& = u_n(h_t, a_t) - \frac{u_n(h_t, a_t)^2}{1 + u_n(h_t, a_t)} \\
& = u_n(h_t, a_t) - \frac{1}{n} n u_n(h_t, a_t) \frac{u_n(h_t, a_t)}{1 + u_n(h_t, a_t)} \\
& = u_n(h_t, a_t) + \eta_n \kappa_{n,t} \frac{-u_n(h_t, a_t)}{1 + u_n(h_t, a_t)}.
\end{aligned}
$$

The update of q in Eq. (4) can also be transformed into

$$
\begin{aligned}
q_{n+1}(h_t, a_t) &= q_n(h_t, a_t) + \frac{1}{m_n(h_t, a_t)}(g_t - q_n(h_t, a_t)), \\
&= q_n(h_t, a_t) + u_n(h_t, a_t)(g_t - q_n(h_t, a_t)), \\
&= q_n(h_t, a_t) + \frac{1}{n} n u_n(h_t, a_t)(g_t - q_n(h_t, a_t)), \\
&= q_n(h_t, a_t) + \eta_n \kappa_{n,t}(g_t - q_n(h_t, a_t)).
\end{aligned}
$$

Eq. (14) is derived. □

B Experimental setup

B.1 Randomly synthesized task

The first test problem is a non-Markovian task that is a simple but illustrative HDP. It is randomly synthesized to be analogous to a generation task such as text generation and compound synthesis. There are 5 observations and 10 actions, i.e., $|\mathcal{O}| = 5$ and $|\mathcal{A}| = 10$. The observation probability function p_{o}, which corresponds to the history transition probability p_{h}, was synthesized to depend on the time-step, observation, and action. Specifically, a probability vector for the observation was generated by the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha} = [0.2, \ldots, 0.2])$ independently for each (t, o, a). The reward function f_{r} was synthesized to have the following structure:

$$f_{\mathrm{r}}(h_t, a_t) = \begin{cases} \frac{1}{T} x(o_t, a_t) & \text{if } t < T, \\ y(h_t) + 10 z(o_0, o_1, \ldots, o_t) & \text{otherwise,} \end{cases}$$

where x and y are the per-step reward function and the history-based reward function, respectively. Each value of those functions was initialized independently by the normal distribution $\mathcal{N}(\mu{=}0, \sigma^2{=}1)$. The values of the function z were set by using a Gaussian process so that the more similar the observation series $(o_0, \ldots, o_T)$ and $(o'_0, \ldots, o'_T)$ were, the closer $z(o_0, \ldots, o_T)$ and $z(o'_0, \ldots, o'_T)$ tended to be. Its covariance function was defined with Hamming distance, and the variance was equal to 1.

This reward function f_{r} can be interpreted in the context of text generation as follows. The function z, which is a dominant part in f_{r}, represents the quality of the generated text, x represents the quality of local word connections, and y is like noise.

The policy π^θ was a softmax and parameterized to have the same structure as the reward function. It will correspond to using domain knowledge and will be a usual setting since the reward function is often predefined by the user. Specifically, π^θ had a parameter for each (o_t, a_t), $(o_0, o_1, .., o_t, a_t)$, and (h_t, a_t)), though it was a redundant parameterization. The hyper-parameters of the applied algorithms are shown in Table 1. As described in Section 5, for fair evaluation, we rst tuned the hyper-parameters of the REINFORCE and MCTL algorithms and then used them for the PG-MCTL and the naive mixture algorithms. The hyper-parameters of the lazy AlphaZero were tuned independently.

It should be noted that the experiments here were conducted on an ordinary Laptop, and the computation time was only a few days.

Table 1: Hyper-parameters used in the randomly synthesized task.

Algorithm	α	C	λ	β	M
REINFORCE	0.01	-	-	-	-
Naive mixture	0.01	5	0.2	-	-
MCTL	-	5	-	-	-
Lazy AlphaZero	0.0067	15	-	-	-
PG-MCTL	0.01	5	0.2	100	50000
PG-MCTL-adpt	0.01	5		100	50000

B.2 T-maze task

There are four possible actions: moving north, east, south, or west. At the initial time step $t = 0$, the agent starts at position S and perceives a signal $\mathsf{X} \in \{1000, 0100\}$, indicating whether the goal position G is situated on either the north or south side of the T-junction. At each subsequent time step $t \in \{1, 2, \ldots\}$, the agent observes its current location type. In the corridor, the observation is

Table 2: Hyper-parameters used in the T-maze task.

	$L = 30$, $s_0 = 0$							$L = 100$, $s_0 = 50$						
Algorithm	α	C	λ	β	M	epoch	clip	α	C	λ	β	M	epoch	clip
REINFORCE	0.2	-	-	-	-	-	-	0.1	-	-	-	-	-	-
PPO	0.06	-	-	-	-	3	0.2	0.03	-	-	-	-	3	0.2
MCTL	-	0.3	-	-	-	-	-	-	0.3	-	-	-	-	-
Lazy AlphaZero	0.02	1	-	-	-	-	-	0.01	1	-	-	-	-	-
PG-MCTL	0.2	0.1	0.2	100	3000	-	-	0.1	0.1	0.2	100	5000	-	-
PG-MCTL-adpt	0.2	0.1	-	100	3000	-	-	0.1	0.1	-	100	5000	-	-

0010. At the T-junction, the observation is 0001, lacking any information about the goal's position. Therefore, the agent must memorize the initial observation to navigate effectively.

The reward settings are as follows. If the correct action is chosen at the T-junction, e.g., move north if X is 1000 and south if 0100, the agent receives a reward of 4.0, otherwise a reward of -0.1. In both cases, the episode ends. When it is in the corridor and chooses to move north or south, it stays there and receives a reward of -0.1. Otherwise, the reward will be zero.

The settings for our model are as follows: The LSTM network, following standard architectures used in reinforcement learning (Bakker, 2002; Wierstra et al., 2010), takes four input units corresponding to the observation dimensions and processes them through a hidden layer with eight memory cells. The LSTM's output is concatenated with the original observation, to form a feature vector. This feature vector is subsequently utilized in linear layers to compute the action values q and the baseline b. This design aims to capture temporal dependencies and learn effective representations for HDPs.

The hyper-parameters of the applied algorithms are listed in Table 2. The discount rate for cumulative rewards was set to $\gamma = 0.98$. The means of selecting the hyper-parameters is the same as for the randomly synthesized task (see B.1). However, while it was necessary to reduce the hyper-parameter C to 0.1 for MCTL to obtain the optimal policy, this setting required a large number of episodes. We therefore set C to a slightly larger value of 0.3 to balance the learning accuracy and speed. On the other hand, for both PG-MCTL and PG-MCTL-adpt, the hyper-parameter C remained at 0.1, as these methods did not necessitate an excessive number of episodes with this configuration.

It is noteworthy that these experiments were conducted on a public cloud, utilizing a single NVIDIA Tesla T4 GPU, with the total computation time being approximately one week.