Preference-based Reinforcement Learning with Finite-Time Guarantees

Yichong Xu[1]*, **Ruosong Wang**[2], **Lin F. Yang**[3], **Aarti Singh**[2], **Artur Dubrawski**[2]
[1]Microsoft
[2]Carnegie Mellon University
[3]University of California, Los Angles
yicxu@microsoft.com, {ruosongw,aarti,awd}@cs.cmu.edu, linyang@ee.ucla.edu

Abstract

Preference-based Reinforcement Learning (PbRL) replaces reward values in traditional reinforcement learning by preferences to better elicit human opinion on the target objective, especially when numerical reward values are hard to design or interpret. Despite promising results in applications, the theoretical understanding of PbRL is still in its infancy. In this paper, we present the first finite-time analysis for general PbRL problems. We first show that a unique optimal policy may not exist if preferences over trajectories are deterministic for PbRL. If preferences are stochastic, and the preference probability relates to the hidden reward values, we present algorithms for PbRL, both with and without a simulator, that are able to identify the best policy up to accuracy ε with high probability. Our method explores the state space by navigating to under-explored states, and solves PbRL using a combination of dueling bandits and policy search. Experiments show the efficacy of our method when it is applied to real-world problems.

1 Introduction

In reinforcement learning (RL), an agent typically interacts with an unknown environment to maximize the cumulative reward. It is often assumed that the agent has access to numerical reward values. However, in practice, reward functions might not be readily available or hard to design, and hand-crafted rewards might lead to undesired behaviors, like reward hacking [8, 1]. On the other hand, preference feedback is often straightforward to specify in many RL applications, especially those involving human evaluations. Such preferences help shape the reward function and avoid unexpected behaviors. Preference-based Reinforcement Learning (PbRL, [28]) is a framework to solve RL using preferences, and has been widely applied in multiple areas including robot teaching [20, 19, 11], game playing [29, 31], and in clinical trials [36].

Despite its wide applicability, the theoretical understanding of PbRL is largely open. To the best of our knowledge, the only prior work with a provable theoretical guarantee is the recent work by Novoseller et al. [25]. They proposed the Double Posterior Sampling (DPS) method, which uses Bayesian linear regression to derive posteriors on reward values and transition distribution. Combining with Thompson sampling, DPS has an asymptotic regret rate sublinear in T (number of time steps). However, this rate is based on the *asymptotic* convergence of the estimates of reward and transition function, whose complexity could be exponential in the time horizon H. Also, the Thompson sampling method in [25] can be very time-consuming, making the algorithm applicable only to MDPs with a few states. To fill this gap, we naturally ask the following question:

Is it possible to derive efficient algorithms for PbRL with finite-time guarantees?

*Work done while at Carnegie Mellon University.

While traditional value-based RL has been studied extensively, including recently [6, 35, 22], PbRL is much harder to solve than value-based RL. Most efficient algorithms for value-based RL utilize the value function and the Bellman update, both of which are unavailable in PbRL: the reward values are hidden and unidentifiable up to shifts in rewards. Even in simple tabular settings, we cannot obtain unbiased estimate of the Q values since any offset in reward function results in the same preferences. Therefore traditional RL algorithms (such as Q learning or value iteration) are generally not applicable to PbRL.

Our Contributions. We give an affirmative answer to our main question above, under general assumptions on the preference distribution.

- We study conditions under which PbRL can recover the optimal policy for an MDP. In particular, we show that when comparisons between trajectories are noiseless, there exists an MDP such that preferences between trajectories are not transitive; i.e., there is no unique optimal policy (Proposition 1). Hence, we base our method and analysis on a general assumption on preferences between trajectories, which is a generalization of the linear link function assumption in [25].
- We develop provably efficient algorithms to find ε-optimal policies for PbRL, with or without a simulator. Our method is based on a synthetic reward function similar to recent literature on RL with rich observations [14, 24] and reward-free RL [23]. We combine this reward-free exploration and dueling bandit algorithms to perform policy search. To the best of our knowledge, this is the first PbRL algorithm with finite-time theoretical guarantees. Our method is general enough to incorporate many previous value-based RL algorithms and dueling bandit methods as a subroutine.
- We test our algorithm against previous baselines in simulated environments. Our results show that our algorithm can beat previous baselines, while being very simple to implement.

Related Work. We refer readers to [28] for a complete overview of PbRL. In the PbRL framework, there are several ways that one can obtain preferences: We can obtain preferences on i) trajectories, where the labeler tells which of two trajectories is more rewarding; ii) actions, where for a fixed state s, the labeler tells which action gives a better action-value function; or iii) states, where similarly, the labeler tells which state gives a better value function. In this paper, we mainly study preferences over trajectories, which is also the most prevalent PbRL scenario in literature. Our method is potentially applicable to other forms of preferences as well.

PbRL is relevant to several settings in Multi-Armed Bandits. Dueling bandits [33, 10] is essentially the one-state version of PbRL, and has been extensively studied in the literature [34, 17, 16, 32]. However, PbRL is significantly harder because in PbRL the observation (preference) is based on the *sum* of rewards on a trajectory rather than individual reward values. For the same reason, PbRL is also close to combinatorial bandits with full-bandit feedback [12, 2, 26]. Although lower bounds for these bandit problems extends to PbRL, developing PbRL algorithms is significantly harder since we are not free to choose any combination of state-action pairs.

2 Problem Setup

MDP Formulation. Suppose a finite-time Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, H, r, p, s_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, H is the number of steps, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function[2], $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the (random) state transition function, and s_0 is the starting state. For simplicity we assume $S \geq H$. We consider finite MDPs with $|\mathcal{S}| = S$, $|\mathcal{A}| = A$. We start an episode from the initial state s_0, and take actions to obtain a trajectory $\tau = \{(s_h, a_h)\}_{h=0}^{H-1}$, following a policy $\pi : \mathcal{S} \to \mathcal{A}$. We also slightly overload the notation to use $r(\tau) = \sum_{(s,a)\in\tau} r(s, a)$ to denote the total reward of τ. For any policy π, let $\tau_h(\pi, s)$ be a (randomized) trajectory by executing π starting at state s from step h to the end.

Following prior works [14, 24], we further assume that the state space $\mathcal{S}$ can be partitioned into H disjoint sets $\mathcal{S} = \mathcal{S}_1 \cup \cdots \mathcal{S}_H$, where $\mathcal{S}_h$ denotes the set of possible states at step h. Let $\Pi : \{\pi : \mathcal{S} \to \mathcal{A}\}$ be the set of policies[3]. We use Π^H to denote the set of non-stationary policies;

[2]For simplicity, here we assume the reward function to be deterministic.

[3]We consider deterministic policies for simplicity, but our results carry to random policies easily.

here a policy $\pi = (\pi_1, ..., \pi_H) \in \Pi^H$ executes policy π_h in step h for $h \in [H]$ [4]. Also, let $\pi_{h_1:h_2}$ be the restriction of policy $\pi \in \Pi^H$ to step $h_1, h_1+1, ..., h_2$. We use the value function $v^\pi_{h_0}(s) = \mathbb{E}[\sum_{h=h_0}^{H-1} r(s_h, \pi(s_h)) | \pi, s_{h_0} = s]$ to denote the expected reward of policy π starting at state s in step h_0; for simplicity let $v^\pi = v^\pi_0$. Let $\pi^* = \arg\max_{\pi \in \Pi^H} v^\pi_0(s_0)$ denote the optimal (non-stationary) policy. We assume that $r(s,a) \in [0,1]$ for every $(s,a) \in \mathcal{S} \times \mathcal{A}$, and that $v^*(s) = v^{\pi^*}(s) \in [0,1]$ for every state s.

Preferences on Trajectories. In PbRL, the reward $r(s,a)$ is hidden and is not observable during the learning process, although we define the value function and optimal policy based on r. Instead, the learning agent can query to compare two trajectories τ and τ', and obtain a (randomized) preference $\tau \succ \tau'$ or $\tau' \succ \tau$, based on $r(\tau)$ and $r(\tau')$. We also assume that we can compare partial trajectories; we can also compare two partial trajectories $\tau = \{(s_h, a_h)\}_{h=h_0}^H$ and $\tau' = \{(s'_h, a'_h)\}_{h=h_0}^H$ for any $h_0 \in [H]$. Let $\phi(\tau, \tau') = \Pr[\tau \succ \tau'] - 1/2$ denote the preference between τ and τ'.

PAC learning and sample complexity. We consider PAC-style algorithms, i.e., an algorithm needs to output a policy $\hat{\pi}$ such that $v^{\hat{\pi}}(s_0) - v^*(s_0) \leq \varepsilon$ with probability at least $1-\delta$. In PbRL, comparisons are collected from human workers and the trajectories are obtained by interacting with the environment. So obtaining comparisons are often more expensive than exploring the state space; we therefore compute separately the sample complexity in terms of the number of total steps and number of comparisons. Formally, let $\mathrm{SC}_s(\varepsilon, \delta)$ be the number of exploration steps needed to obtain an ε-optimal policy with probability $1-\delta$, and SC_p be the number of preferences (comparisons) needed in this process. We omit ε, δ from the sample complexities when the context is clear.

Dueling Bandit Algorithms. Our proposed algorithms uses a dueling bandit algorithm as a subroutine. To utilize preferences, our algorithms use a PAC dueling bandit algorithm to compare policies starting at the same state. Dueling Bandits [33] has been well studied in the literature. Examples of PAC dueling bandit algorithms include Beat the Mean [34], KNOCKOUT [17], and OPT-Maximize [16]. We formally define a dueling bandit algorithm below.

Definition 1 ((ε, δ)-correct Dueling Bandit Algorithm). *Let $\varepsilon > 0, \delta > 0$. $\mathcal{M}$ is a (ε, δ)-correct PAC dueling bandit algorithm if for any given set of arms $\mathcal{X}$ with $|\mathcal{X}| = K$, i) $\mathcal{M}$ runs for at most $\Psi(\varepsilon, \delta)\varepsilon^{-\alpha}$ steps, where $\Psi(\varepsilon, \delta) = poly(K, \log(1/\varepsilon), \log(1/\delta))$ and $\alpha \geq 1$; ii) in every step, $\mathcal{M}$ proposes two arms a, a' to compare; iii) Upon completion, $\mathcal{M}$ returns an arm $\hat{a}$ such that $\Pr[\hat{a} \succ a] \geq 1/2 - \varepsilon$ for every arm $a \in \mathcal{X}$, with probability at least $1-\delta$.*

One important feature of existing PAC dueling bandit algorithms is whether they require knowing ε before they start - algorithms like KNOCKOUT and OPT-Maximize [17, 16] cannot start without knowledge of ε; Beat-the-Mean [34] does not need to know ε to start, but can return an ε-optimal arm when given the correct budget. We write $\mathcal{M}(\mathcal{X}, \varepsilon, \delta)$ for an algorithm with access to arm set $\mathcal{X}$, accuracy ε and success probability $1-\delta$; we write $\mathcal{M}(\mathcal{X}, \delta)$ for a dueling bandit algorithm without using the accuracy ε.

Results in the value-based RL and bandit setting. In traditional RL where we can observe the reward value at every step, the minimax rate is largely still open [21] [5]. The upper bound in e.g., [6] translates to a step complexity of $O\left(\frac{H^3SA}{\varepsilon^2}\right)$ to recover an ε-optimal policy, but due to scaling of the rewards the lower bound [13] translates to $\Omega\left(\frac{HSA}{\varepsilon^2}\right)$. Very recently, Wang et al. [27] show an upper bound of $O\left(\frac{HS^3A^2}{\varepsilon^3}\right)$, showing that the optimal H dependence might be linear. It is straightforward to show that the lower bound in [13] translates to a step complexity of $\Omega\left(\frac{HSA}{\varepsilon^2}\right)$ and a comparison complexity of $\Omega\left(\frac{SA}{\varepsilon^2}\right)$ for PbRL. Lastly, we mention that the lower bounds for combinatorial bandits with full-bandit feedback [12] can transform to a lower bound of the same scale for PbRL.

2.1 Preference Probablities

As in ranking and dueling bandits, a major question when using preferences is how to define the winner. One common assumption is the existence of a Condorcet winner: Suppose there exists an item (in our case, a policy) that beats all other items with probability greater than 1/2. However in PbRL, because we compare trajectories, preferences might not reflect the true relation between

[4] We follow the standard notation to denote $[H] = \{0, 1, ..., H-1\}$.

[5] Note that some prior works assumes that $r(s,a) \in [0, 1/H]$ [6, 35], which is different from our setting.

policies. For example, assume that the comparisons are perfect, i.e., $\tau_1 \succ \tau_2$ if $r(\tau_1) > r(\tau_2)$ and vice versa. Now suppose policy π_1 has a reward of 1 with probability 0.1 and 0 otherwise, and π_2 has a reward of 0.01 all the time. Then a trajectory from π_1 is only preferred to a trajectory from π_2 with probability 0.1 if the comparisons give deterministic results on trajectory rewards. Extending this argument, we can show that non-transitive relations might exist between policies:

Proposition 1. *Slightly overloading the notation, for any $h \in [H]$ and $s_h \in \mathcal{S}_h$, let $\phi_s(\pi_1, \pi_2) = \Pr[\tau_h(\pi_1, s) \succ \tau_h(\pi_2, s)] - 1/2$ denote the preference between policies π_1 and π_2 when starting at s_h in step h. Suppose comparisons are noiseless. There exists a MDP and policies π_0, π_1, π_2 such that for some state $s \in \mathcal{S}$, $\phi_s(\pi_0, \pi_1), \phi_s(\pi_1, \pi_2), \phi_s(\pi_2, \pi_0)$ are all less than 0.*

Proposition 1 shows that making assumptions on the preference distribution on trajectories cannot lead to a unique solution for PbRL. [6] Instead, since our target is an optimal policy, we make assumptions on the preferences between *trajectories*:

Assumption 1. *There exists a universal constant $C_0 > 0$ such that for any policies π_1, π_2 and state $s \in \mathcal{S}$ with $v_{\pi_1}(s) - v_{\pi_2}(s) > 0$, we have $\phi_s(\pi_1, \pi_2) \geq C_0(v_{\pi_1}(s) - v_{\pi_2}(s))$.*

I.e., the preference probabilities depends on the value function difference. Recall that $\phi_s(\pi_1, \pi_2)$ is the probability that a *random* trajectory from π_1 beats that from π_2; we do not make any assumptions on the comparison result of any *individual* trajectories. Assumption 1 ensures that a unique Condorcet winner (which is also the reward-maximizing policy) exists. Note that Assumption 1 also holds under many comparison models for ϕ_s, such as the BTL or Thurstone model. Following previous literatures in dueling bandits [34, 17], we also define the following properties on preferences:

Definition 2. *Define the following properties on preferences when the following holds for any h, $s \in \mathcal{S}_h$ and three policies π_1, π_2, π_3 such that $v_h^{\pi_1}(s) > v_h^{\pi_2}(s) > v_h^{\pi_3}(s)$:*
Strong Stochastic Transitivity: *$\phi_{s_h}(\pi_1, \pi_3) \geq \max\{\phi_{s_h}(\pi_1, \pi_2), \phi_{s_h}(\pi_2, \pi_3)\}$;*
Stochastic Triangle Inequality: *$\phi_{s_h}(\pi_1, \pi_3) \leq \phi_{s_h}(\pi_1, \pi_2) + \phi_{s_h}(\pi_2, \pi_3)$.*

These properties are not essential for our algorithms, but are required for some dueling bandit algorithms that we use as a subroutine. To see the relation between policy preferences and reward preferences, we show the next proposition on several special cases:

Proposition 2. *If either of the following is true, the preferences satisfy Assumption 1, SST and STI:*
i) There exists a constant C' such that for every pair of trajectories τ, τ' we have $\phi(\tau, \tau') = C'(r(\tau) - r(\tau'))$.
ii) The transitions are deterministic, and $\phi(\tau, \tau') = c$ for $r(\tau) > r(\tau')$ and some $c \in (0, 1/2]$.

The first condition in Proposition 2 is the same as the assumption in [25], so our Assumption 1 is a generalization of theirs. We also note that although we focus on preferences over trajectories, preference between policies, as in Assumption 1, is also used in practice [18].

3 PbRL with a Simulator

In this section, we assume access to a simulator (generator) that allows access to any state $s \in \mathcal{S}$, executes an action $a \in \mathcal{A}$ and obtains the next state $s' \sim p(\cdot|s, a)$. We first introduce our algorithm, to then follow with theoretical results. We also show a lower bound that our comparison complexity is almost optimal.

Although value-based RL with a simulator has been well-studied in the literature [3, 4], methods like value iteration cannot easily extend to PbRL, because the reward values are hidden. Instead, we base our method on dynamic programming and policy search [7] - by running a dueling bandit algorithm on each state, one can determine the corresponding optimal action. The resulting algorithm, Preference-based Policy Search (PPS), is presented in Algorithm 1. By inducting from $H - 1$ to 0, PPS solves a dueling bandit problem at every state $s_h \in \mathcal{S}_h$, with arm rewards specified by $a \circ \hat{\pi}_{h+1:H}$ for $a \in \mathcal{A}$, where $\hat{\pi}$ is the estimated best policy after step $h + 1$, and $\circ$ stands for concatenation of policies. By allocating an error of $O(\varepsilon/H)$ on every state, we obtain the following guarantee:

[6] One might consider other winner notions when a Condorcet winner policy does not exist. e.g., the von Neumann winner [15]. However, finding such winners usually involves handling an exponential number of policies and is out of the scope of the current paper.

Algorithm 1 PPS: Preference-based Policy Search

```
Require: Dueling bandit algorithm $\mathcal{M}$, dueling accuracy $\varepsilon_1$, sampling number $N_2$, success probability $\delta$
 1: Initialize $\hat{\pi} \in \Pi^H$ randomly
 2: for $h = H - 1, ..., 0$ do
 3:     for $s_h \in \mathcal{S}_h$ do
 4:         for $n = 1, 2, ..., N_2$ do
 5:             Start an instance of $\mathcal{M}\left(\mathcal{A}, \varepsilon_1, \delta/S\right)$
 6:             Receive query $(a, a')$ from $\mathcal{M}$
 7:             Rollout $a \circ \hat{\pi}_{h+1:H}$ from $s_h$, get trajectory $\tau$
 8:             Rollout $a' \circ \hat{\pi}_{h+1:H}$ from $s_h$, get trajectory $\tau'$
 9:             Compare $\tau, \tau'$ and return the result to $\mathcal{M}$
10:         end for
11:         if $\mathcal{M}$ has finished then
12:             Update $\hat{\pi}_h$ to the optimal action according to $\mathcal{M}$
13:         end if
14:     end for
15: end for
Ensure: Policy $\hat{\pi}$
```

Theorem 3. *Suppose the preference distribution satisfies Assumption 1. Let* $\varepsilon_1 = \frac{C_0\varepsilon}{H}, N_0 = \Psi(\varepsilon_1, \delta/S)\varepsilon_1^{-\alpha}$*, where* C_0 *is defined in Assumption 1. Algorithm 1 returns an* ε*-optimal policy with probability* $1 - \delta$ *using* $O\left(\frac{H^{\alpha+1}S\Psi(A,\varepsilon/H,\delta/S)}{\varepsilon^{\alpha}}\right)$ *simulator steps and* $O\left(\frac{H^{\alpha}S\Psi(A,\varepsilon/H,\delta/S)}{\varepsilon^{\alpha}}\right)$ *comparisons.*

We can plug in existing algorithms to instantiate Theorem 3. For example, under SST, OPT-Maximize [16] achieves the minimax optimal rate for dueling bandits. In particular, it has a comparison complexity of $O\left(\frac{K\log(1/\delta)}{\varepsilon^2}\right)$ for selecting an ε-optimal arm with probability $1 - \delta$ among K arms. Plugging in this rate we obtain the following corollary:

Corollary 4. *Suppose the preference distribution satisfies Assumption 1 and SST. Using OPT-Maximize as* $\mathcal{M}$*, Algorithm 1 returns an* ε*-optimal policy with probability* $1 - \delta$ *using* $O\left(\frac{H^3SA}{\varepsilon^2}\log(S/\delta)\right)$ *simulator steps, and* $O\left(\frac{H^2SA}{\varepsilon^2}\log(S/\delta)\right)$ *comparisons.*

The proof of Theorem 3 follows from using the performance difference lemma, combined with properties of $\mathcal{M}$. Our result is similar to existing results for traditional RL with a simulator: For example, in the case of infinite-horizon MDPs with a decaying factor of γ, [5] shows a minimax rate of $O\left(\frac{SA}{\varepsilon^2(1-\gamma)^3}\right)$ on step complexity. This is the same rate as in Corollary 4 by taking $H = \frac{1}{1-\gamma}$ effective steps.

4 Combining Exploration and Policy Search for General PbRL

In this Section, we present our main result for PbRL without a simulator. RL without a simulator is a challenging problem even in the traditional value-based RL setting. In this case, we will have to explore the state space efficiently to find the optimal policy. Existing works in value-based RL typically derive an upper bound on the Q-value function and apply the Bellman equation to improve the policy iteratively [6, 22, 35]. However for PbRL, since the reward values are hidden, we cannot apply traditional value iteration and Q-learning methods. To go around this problem, we use a synthetic reward function to guide the exploration. We present our main algorithm in Section 4.1, along with the theoretical analysis. We discuss relations to prior work in Section 4.2.

4.1 Preferece-based Exploration and Policy Search (PEPS)

We call our algorithm Preference-based Exploration and Policy Search (PEPS), and present it in Algorithm 2. As the name suggests, the algorithm combines exploration and policy search. For every

Algorithm 2 PEPS: Preferece-based Exploration and Policy Search

```
Require: Dueling Bandit algorithm $\mathcal{M}$, quantity $N_0$, success probability $\delta$
 1: Initialize $\hat{\pi} \in \Pi^H$ randomly
 2: for $h = H, H-1, ..., 1$ do
 3:     for $s_h \in \mathcal{S}_h$ do
 4:         Let $r_{s_h}(s, a) = 1_{s=s_h}$ for all $s \in S, a \in A$
 5:         Start an instance of EULER$(r_{s_h}, N_0, \delta/(4S))$
 6:         Start an instance of $\mathcal{M}$ $(\mathcal{A}, \delta/(4S))$
 7:         Get next query $(a, a')$ from $\mathcal{M}$
 8:         $U \leftarrow \emptyset$
 9:         for $n \in [N_0]$ do
10:             Obtain a policy $\hat{\pi}_n$ from EULER, and execute $\hat{\pi}_n$ until step $h$
11:             Return the trajectory and reward to EULER
12:             if current state is $s_h$ then
13:                 Let $\bar{\pi} = a \circ \hat{\pi}_{h+1:H}$ if $|U| = 0$, otherwise $a' \circ \hat{\pi}_{h+1:H}$
14:                 Execute $\bar{\pi}$ till step $H$, obtain trajectory $\tau$
15:                 $U \leftarrow U \cup \tau$
16:             end if
17:             if $|U| = 2$ then
18:                 Compare the two trajectories in $U$ and return to $\mathcal{M}$
19:                 $(a, a') \leftarrow$ next action from $\mathcal{M}$
20:             end if
21:             If $\mathcal{M}$ has finished, break
22:         end for
23:         if $\mathcal{M}$ has finished then
24:             Update $\hat{\pi}_h(s_h)$ to the optimal action according to $\mathcal{M}$
25:         end if
26:     end for
27: end for
Ensure: Policy $\hat{\pi}$
```

step $h \in [H]$ and $s_h \in \mathcal{S}_h$, we set up an artificial reward function $r_{s_h}(s, a) = 1_{s=s_h}$, i.e., the agent gets a reward of 1 once it gets to s_h, and 0 everywhere else (Step 4). This function is also used in recent reward-free learning literatures [14, 24, 23]. But rather than using the reward function to obtain a policy cover (which is costly in both time and space), we use it to help the dueling bandit algorithm. We can use arbitrary tabular RL algorithm to optimize r_{s_h}; here we use EULER [35] with a budget of N_0 and success probability $\delta/4S$. The reason that we chose EULER is mainly for its beneficial regret guarantees; but we can also use other algorithms in practice. We also start an instance of $\mathcal{M}$, the dueling bandit algorithm, without setting the target accuracy; one way to achieve this is to use a pre-defined accuracy for $\mathcal{M}$, or use algorithms like Beat-the-Mean (see Theorem 5 and 7 respectively).

Once we get to s_h (Step 12), we execute the queried action a or a' from $\mathcal{M}$, followed by the current best policy $\hat{\pi}$, as in PPS. If we have collected two trajectories, we compare them and feed it back to $\mathcal{M}$. Upon finishing N_0 steps of exploration, we update the current best policy $\hat{\pi}$ according to $\mathcal{M}$. We return $\hat{\pi}$ when we have explored every state $s \in \mathcal{S}$.

Throughout this section, we use $\iota = \log\left(\frac{SAH}{\varepsilon\delta}\right)$ to denote log factors. We present two versions of guarantees with different sample complexity. The first version has a lower comparison complexity, while the second version has a lower total step complexity. The different guarantee comes from setting a slightly different target for $\mathcal{M}$ when it finishes in Step 21. In the following first version, we set $\mathcal{M}$ to finish when it finds a $O(\varepsilon/H)$-optimal policy.

Theorem 5. *Suppose the preference distribution satisfies Assumption 1. There exists a constant c_0 such that the following holds: Let $N_0 = c_0\left(\frac{H^\alpha S\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^{\alpha+1}} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$, recall $\iota = \log\left(\frac{SAH}{\varepsilon\delta}\right)$. By setting the target accuracy as $\frac{C_0\varepsilon}{2H}$ for $\mathcal{M}$, PEPS obtains an ε-optimal policy with probability*

$1-\delta$ *using step complexity*

$$O(HSN_0) = O\left(\frac{H^{\alpha+1}S^2\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^{\alpha+1}} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{H^\alpha S\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^\alpha}\right)$.

Since we set the target accuracy before the algorithm starts, any PAC dueling bandit algorithm can be used to instantiate Theorem 5. So similar to Theorem 3, we can plug in OPT-Maximize [16] to obtain the following corollary:

Corollary 6. *Suppose the preference distribution satisfies Assumption 1 and SST. There exists a constant* c_0 *such that the following holds: Let* $N_0 = c_0\left(\frac{SH^2A\log(S/\delta)}{\varepsilon^3} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$. *Using OPT-Maximize with accuracy* ε/H *as* $\mathcal{M}$, *PEPS obtains an* ε*-optimal policy with probability* $1-\delta$ *using step complexity*

$$O(HSN_0) = O\left(\frac{H^3S^2A\log(S/\delta)}{\varepsilon^3} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{H^2SA\log(S/\delta)}{\varepsilon^2}\right)$.

In the second version, we simply do not set *any* performance target for $\mathcal{M}$; it will explore until we finish all the N_0 episodes. Therefore, we never break in Step 21 and $\mathcal{M}$ is always finished in Step 23. Since different states will be reached for a different number of times, we cannot pre-set a target accuracy for $\mathcal{M}$. Effectively, we explore every state s to the accuracy of $\varepsilon_s = O\left(\frac{\varepsilon}{(\mu(s_h)SH^{\alpha-1})^{1/\alpha}}\right)$ (see proof in appendix for details), where $\mu(s)$ is the maximum probability to reach s using any policy. Although this version leads to a slightly higher comparison complexity, it leads to a better step complexity since all exploration steps are used efficiently. We have the following result:

Theorem 7. *Suppose the preference distribution satisfies Assumption 1. There exists a constant* c_0 *such that the following holds: Let* $N_0 = c_0\left(\frac{H^{\alpha-1}S\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$. *PEPS obtains an* ε*-optimal policy with probability* $1-\delta$ *using step complexity*

$$O(HSN_0) = \left(\frac{H^\alpha S^2\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{H^{\alpha-1}S^2\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha}\right)$.

We need to instantiate Theorem 7 with an $\mathcal{M}$ that does not need a pre-set accuracy. Under SST and STI, we can use Beat-the-Mean [34], which returns an ε-optimal arm among K arms with probability $1-\delta$, if it runs for $\Omega\left((\frac{K}{\varepsilon^2}\log\left(\frac{K}{\varepsilon\delta}\right)\right)$ steps. We therefore obtain the following corollary:

Corollary 8. *Suppose the preference distribution satisfies Assumption 1, SST and STI. There exists a constant* c_0 *such that the following holds: Let* $N_0 = c_0\left(\frac{HSA\iota}{\varepsilon^2} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$. *Using Beat-the-Mean as* $\mathcal{M}$, *PEPS obtains an* ε*-optimal policy with probability* $1-\delta$ *using step complexity*

$$O(HSN_0) = O\left(\frac{H^2S^2A\iota}{\varepsilon^2} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{HS^2A\iota}{\varepsilon^2}\right)$.

4.2 Discussion

Comparing Corollary 6 and 8. Considering only the leading terms, the step complexity in Corollary 8 is better by a factor[7] of $\tilde{O}(H/\varepsilon)$ but the comparison complexity is worse by a factor of $\tilde{O}(S/H)$ (recall that we assume $S > H$). Therefore the two theorems depict a tradeoff between the step complexity and comparison complexity.

[7]We use $\tilde{O}$ to ignore log factors.

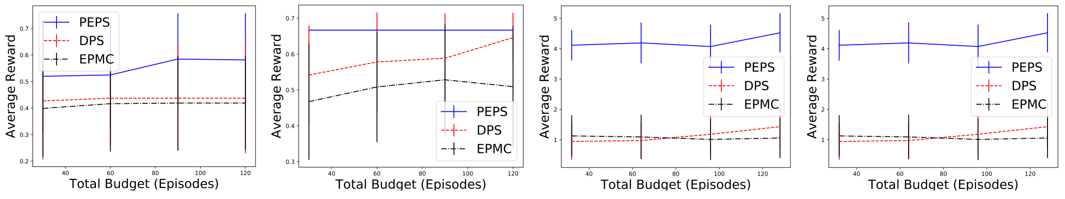
Comparing with lower bounds and value-based RL. Corollary 6 has the same comparison complexity as Corollary 4. The lower bound for comparison complexity is $\tilde{O}(\frac{SA}{\varepsilon^2})$, a $\tilde{O}(H^2)$ factor from our result. Our comparison complexity is also the same as the current best upper bound in value-based RL (in terms of number of episodes), which shows that our result is close to optimal.

Compared with the $\tilde{O}(\frac{HSA}{\varepsilon^2})$ lower bound on step complexity, Corollary 8 has an additional factor of $\tilde{O}(HS)$. The current best upper bound in value-based RL is $\tilde{O}(\frac{H^3SA}{\varepsilon^2})$, which is $O(H/S)$ times our result (recall that we assume $H < S$).

Comparing with previous RL method using policy covers. Some literature on reward-free learning and policy covers [14, 24, 23] use a similar form of synthetic reward function as ours. [14, 24] considers computing a policy cover by exploring the state space using a similar synthetic function as PEPS. However, our results are generally not comparable since they assume that $\mu(s) \geq \eta$ for some $\eta > 0$, and their result depends on η. Our result do not need to depend on this η. Closest to our method is the recent work of [23], which considers reward-free learning with unknown rewards during exploration. However, their method cannot be applied to PbRL since we do not have the reward values even in the planning phase. We note that the $O(S^2)$ dependence on the step complexity is also common in these prior works. Nevertheless, our rates are better in H dependence because we do not need to compute the policy cover explicitly.

Fixed budget setting. While we present PPS and PEPS under the fixed confidence setting (with a given ε), one can easily adapt it to the fixed budget setting (with a given N, number of episodes) by dividing N evenly among the states. We present a fixed budget version in the appendix.

Two-phase extension of PEPS. A drawback of Theorem 7 is that it requires $\mathcal{M}$ to work without a pre-set target accuracy, limiting the algorithm that we can use to instantiate $\mathcal{M}$. In the appendix, we present a two-phase version of PEPS that allows for arbitrary PAC algorithm $\mathcal{M}$, with slightly improved log factors on the guarantee.



(a) GridWorld, $c = 1$ (b) GridWorld, $c = 0.001$ (c) Random MDP, $c = 1$ (d) Random MDP, $c = 0.001$

Figure 1: Experiment Results comparing PEPS to baselines (DPS & EPMC).

5 Experiments

We performed experiments in synthetic environments to compare PEPS with previous baselines. We consider two environments:
Grid World: We implemented a simple Grid World on a 4×4 grid. The agent goes from the upper left corner to the lower right corner and can choose to go right or go down at each step. We randomly put a reward of $1/3$ on three blocks in the grid, and the maximal total reward is $2/3$.
Random MDP: We followed the method in [25] but adapted it to our setting. We consider an MDP with 20 states and 5 steps, with 4 states in each step. The transitions are sampled from a Dirichlet prior (with parameters all set to 0.1) and the rewards are sampled from an exponential prior with scale parameter $\lambda = 5$. The rewards are then shifted and normalized so that the minimum reward is 0 and the mean reward is 1.

Compared methods. We compared to three baselines: i) DPS [25]: We used the version with Gaussian process regression since this version gets the best result in their experiments; ii) EPMC [30], which uses preferences to simulate a Q-function. We followed the default parameter settings for both DPS and EPMC. Details of the algorithms and hyperparameter settings are included in the appendix.

Experiment Setup. Our baselines are not directly comparable to PEPS since their goal is to minimize the regret, instead of getting an ε-optimal policy. However, we can easily adapt all the algorithms (including PEPS) to the fixed budget setting for optimal policy recovery. For the baselines, we ran them until N episodes and then evaluated the current best policy. For PEPS, we used the fixed budget version described in detail in the appendix. For both environments, we varied the budget $N \in [2S', 8S']$, where S' is the number of non-terminating states. The comparisons are generated following the Bradley-Terry-Luce model [9]: $\phi(\tau_1, \tau_2) = \frac{1}{1+\exp(-(r(\tau_1)-r(\tau_2))/c)}$, with c being either 0.001 or 1. In the first setting of c, the preferences are very close to deterministic while comparison between equal rewards is uniformly random; in the latter setting, the preferences are close to linear in the reward difference. We repeated each experiment for 32 times and computed the mean and standard deviation.

Results. The results are summarized in Figure 1. Overall, PEPS outperforms both baselines in both environments. In Grid World, while all three methods get a relatively high variance when $c = 1$, for $c = 0.001$ PEPS almost always get the exact optimal policy. Also for random MDP, PEPS outperforms both baselines by a large margin (larger than two standard deviations). We note that EPMC learns very slowly and almost does not improve as the budget increases, and this is consistent with the observation in [25]. One potential reason that makes PEPS outperform the baselines is because of the exploration method: Both DPS and EPMC need to estimate the Q function well in order to perform efficient exploration. This estimation can take time exponential in H, and it is not even computationally feasible to test until the Q function converges. As a result, both DPS and EPMC explore almost randomly and cannot recover the optimal policy. On the other hand, our method uses a dueling bandit algorithm to force exploration, so it guarantees that at least states with high reach probability have their optimal action.

6 Conclusion

We analyze the theoretical foundations of the PbRL framework in detail and present the first finite-time analysis for general PbRL problems. Based on reasonable assumptions on the preferences, the proposed PEPS method recovers an ε-optimal policy with finite-time guarantees. Experiments show the potential efficacy of our method in practice, and that it can work well in simulated environments. Future work includes i) testing PEPS in other RL environments and applications, ii) developing algorithms for PbRL with finite-time regret guarantees, iii) algorithms for the case when Assumption 1 holds with some error, and iv) PbRL in non-tabular settings.

Broader Impact Discussion

Reinforcement learning is increasingly being used in a myriad of decision-making applications ranging from robotics and drone control, to computer games, to tutoring systems, to clinical trials in medicine, to social decision making. Many of these applications have goals that are hard to capture mathematically using a single reward definition, and the common practice of reward hacking [8, 1] (trying to achieve best performance under the specified metric) often leads to undesired behaviors with respect to the original goal of the application. Preference based RL provides an appealing alternative where the user can specify their preferences over alternative trajectories without hard-coding a single reward metric and this way avoid unexpected behavior caused by misspecified reward metrics. This paper substantially expands theoretical understanding of preference based RL by providing the first finite time guarantees, and it could help broaden applicability of this learning modality across multiple areas of its potential beneficial use, in particular where it is currently considered too expensive to set up and deploy.

References

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[2] J.-Y. Audibert, S. Bubeck, and G. Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, 2014.

[3] M. G. Azar, R. Munos, M. Ghavamzadaeh, and H. J. Kappen. Speedy q-learning. 2011.

[4] M. G. Azar, R. Munos, and B. Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

[5] M. G. Azar, R. Munos, and H. J. Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

[6] M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. *arXiv preprint arXiv:1703.05449*, 2017.

[7] J. A. Bagnell, S. M. Kakade, J. G. Schneider, and A. Y. Ng. Policy search by dynamic programming. In *Advances in neural information processing systems*, pages 831–838, 2004.

[8] F. Berkenkamp, A. Krause, and A. P. Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016.

[9] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[10] R. Busa-Fekete, E. Hüllermeier, and A. E. Mesaoudi-Paul. Preference-based online learning with dueling bandits: A survey. *arXiv preprint arXiv:1807.11398*, 2018.

[11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[12] A. Cohen, T. Hazan, and T. Koren. Tight bounds for bandit combinatorial optimization. *arXiv preprint arXiv:1702.07539*, 2017.

[13] C. Dann and E. Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

[14] S. S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudík, and J. Langford. Provably efficient rl with rich observations via latent state decoding. *arXiv preprint arXiv:1901.09018*, 2019.

[15] M. Dudík, K. Hofmann, R. E. Schapire, A. Slivkins, and M. Zoghi. Contextual dueling bandits. *arXiv preprint arXiv:1502.06362*, 2015.

[16] M. Falahatgar, Y. Hao, A. Orlitsky, V. Pichapati, and V. Ravindrakumar. Maxing and ranking with few assumptions. In *Advances in Neural Information Processing Systems*, pages 7060–7070, 2017.

[17] M. Falahatgar, A. Orlitsky, V. Pichapati, and A. T. Suresh. Maximum selection and ranking under noisy comparisons. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1088–1096. JMLR. org, 2017.

[18] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2):123–156, 2012.

[19] A. Jain, S. Sharma, T. Joachims, and A. Saxena. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, 34(10):1296–1313, 2015.

[20] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *Advances in neural information processing systems*, pages 575–583, 2013.

[21] N. Jiang and A. Agarwal. Open problem: The dependence of sample complexity lower bounds on planning horizon. In *Conference On Learning Theory*, pages 3395–3398, 2018.

[22] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is q-learning provably efficient? *arXiv preprint arXiv:1807.03765*, 2018.

[23] C. Jin, A. Krishnamurthy, M. Simchowitz, and T. Yu. Reward-free exploration for reinforcement learning. *arXiv preprint arXiv:2002.02794*, 2020.

[24] D. Misra, M. Henaff, A. Krishnamurthy, and J. Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. *arXiv preprint arXiv:1911.05815*, 2019.

[25] E. R. Novoseller, Y. Sui, Y. Yue, and J. W. Burdick. Dueling posterior sampling for preference-based reinforcement learning. 2019.

[26] I. Rejwan and Y. Mansour. Top-k combinatorial bandits with full-bandit feedback. In *Algorithmic Learning Theory*, pages 752–776, 2020.

[27] R. Wang, S. S. Du, L. F. Yang, and S. M. Kakade. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning? *arXiv preprint arXiv:2005.00527*, 2020.

[28] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017.

[29] C. Wirth and J. Fürnkranz. First steps towards learning from game annotations. 2012.

[30] C. Wirth and J. Fürnkranz. Epmc: Every visit preference monte carlo for reinforcement learning. In *Asian Conference on Machine Learning*, pages 483–497, 2013.

[31] C. Wirth and J. Fürnkranz. On learning from game annotations. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(3):304–316, 2015.

[32] Y. Xu, A. Joshi, A. Singh, and A. Dubrawski. Zeroth order non-convex optimization with dueling-choice bandits. *arXiv preprint arXiv:1911.00980*, 2019.

[33] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[34] Y. Yue and T. Joachims. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 241–248, 2011.

[35] A. Zanette and E. Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. *arXiv preprint arXiv:1901.00210*, 2019.

[36] Y. Zhao, D. Zeng, M. A. Socinski, and M. R. Kosorok. Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics*, 67(4):1422–1433, 2011.

A Another Version to Accommodate Arbitrary PAC Dueling Algorithm

A drawback of PEPS is that the version in Theorem 7 requires a dueling algorithm $\mathcal{M}$ that does not need a target accuracy, restricting the possible types of algorithms we can use. In this section, we present a two-phase version of our algorithm to allow arbitrary $\mathcal{M}$.

The two-phase version is presented in Algorithm 3 as PEPS2. PEPS2 separates the process of obtaining a policy cover and using dueling bandits for policy search; in the first phase (Step 1-9) uses the synthetic reward function to obtain a policy cover with EULER. We then estimate $\mu(s)$ for every state $s \in \mathcal{S}$ by executing the policy that we obtain. Using the estimate $\hat{\mu}_s$, we follow the process in PEPS with the target accuracy specified in Step 12.

Algorithm 3 PEPS2: Preferece-based Exploration and Policy Search with 2 phases

```
Require: Target Accuracy $\varepsilon$, Dueling Bandit algorithm $\mathcal{M}$, quantities $N_0, N_1, N_2$, success probability $\delta$
 1: for $h \in [H]$ do
 2:     $\tilde{S}_h = \emptyset$
 3:     for $s_h \in \mathcal{S}_h$ do
 4:         Let $r_{s_h}(s, a) = 1_{s=s_h}$ for all $s \in S, a \in A$
 5:         Obtain a policy $\hat{\pi}_{s_h}$ by using EULER$(r_{s_h}, N_0, \delta/(4S))$ to optimize $r_{s_h}(s, a)$
 6:         Rollout $\hat{\pi}_{s_h}$ for $N_1$ times and record reward the total reward $\hat{R}_{s_h}$ under $r_{s_h}$
 7:         Let $\hat{\mu}_{s_h} \leftarrow \min\{1, 2\hat{R}_{s_h}/N_1 + 2\sqrt{\frac{\log(4S/\delta)}{N}}\}$
 8:     end for
 9: end for
10: Initialize $\hat{\pi} \in \Pi^H$ randomly
11: for $h = H, H-1, ..., 1$ do
12:     Let $\varepsilon_{s_h} \leftarrow \frac{\varepsilon}{4(\hat{\mu}(s_h)SH^{\alpha-1})^{1/\alpha}}$
13:     for $s_h \in \mathcal{S}_h$ do
14:         Start an instance of $\mathcal{M}$ $(\mathcal{A}, \varepsilon_{s_h}, \delta/(4S))$
15:         Get next query $(a, a')$ from $\mathcal{M}$
16:         $U \leftarrow \emptyset$
17:         for $n \in [N_0]$ do
18:             Execute $\hat{\pi}_{s_h}$ until step $h$
19:             if current state is $s_h$ then
20:                 Let $\overline{\pi} = a \circ \hat{\pi}_{h+1:H}$ if $|U| = 0$, otherwise $a' \circ \hat{\pi}_{h+1:H}$
21:                 Execute $\overline{\pi}$ till step $H$, obtain trajectory $\tau$
22:                 $U \leftarrow U \cup \tau$
23:             end if
24:             if $|U| = 2$ then
25:                 Compare the two trajectories in $U$ and return to $\mathcal{M}$
26:             end if
27:             If $\mathcal{M}$ has finished, break
28:         end for
29:         if $\mathcal{M}$ has finished then
30:             Update $\hat{\pi}_h(s_h)$ to the optimal action according to $\mathcal{M}$
31:         end if
32:     end for
33: end for
Ensure: Policy $\hat{\pi}$
```

For every state $s \in \mathcal{S}$, let $\mu(s) = \max_{\pi \in \Pi} p_\pi(s)$ be the maximum probability to reach s.

Theorem 9. *There exists a constant c_0 such that the following holds. Let $N_0 = c_0 \frac{S^3 A H^2 \iota^3}{\varepsilon}$, $N_1 = \frac{c_0 S^2 \log(4S/\delta)}{\varepsilon^2}$, $N_2 = \frac{H^{\alpha-1} S \Psi(A, \frac{\varepsilon}{HS}, \delta/4S)}{\varepsilon^\alpha}$. With probability $1 - \delta$, Algorithm 3 returns an ε-optimal policy using $HS(N_0 + N_1 + N_2)$ steps and SN_2 comparisons.*

We can plug in the guarantee of OPT-Maximize instantiate Theorem 9. This result is better in terms of the log terms than the result in Corollary 8.

Corollary 10. *There exists constants* c_0 *such that the following holds: Let* $N_0 = c_0 \left(\frac{HSA \log(\delta/S)}{\varepsilon^2} + \frac{S^3AH^2\iota^3}{\varepsilon} \right)$*, where* $\iota = \log(\frac{SAH}{\varepsilon\delta})$*. Using OPT-Maximize as* $\mathcal{M}$*, PEPS2 obtains an* ε*-optimal policy with probability* $1 - \delta$ *using step complexity*

$$O(HSN_0) = O\left(\frac{H^2S^2A \log(\delta/S)}{\varepsilon^2} + \frac{S^4AH^3\iota^3}{\varepsilon} \right)$$

and comparison complexity $O\left(\frac{HS^2A \log(S/\delta)}{\varepsilon^2} \right)$.

B Adapting PEPS to the Fixed Budget setting

We adapt PEPS to the fixed budget setting, described in detail in Algorithm 4. To do this, we set $N_0 = N/S$, where S is the number of non-terminating states. Before the algorithm start, we start an instance of $\mathcal{M}$ for every state $s \in \mathcal{S}$; and instead of only compare when reaching the target state, we simply explore according to $\mathcal{M}$ regardless of the current state at step h.

In our experiments We realize PEPS with Q-learning instead of EULER because of its computational efficiency; we use the formulation of [22] with a Hoeffding upper bound on the Q function. For $\mathcal{M}$, we use Beat-the-Mean because it can use any budget.

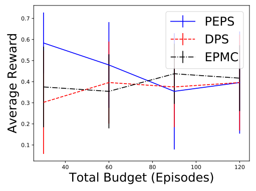
Algorithm 4 PEPS with Fixed Budget

```
Require: Budget $N$, dueling bandit algorithm $\mathcal{M}$, success probability $\delta$
 1: Initialize $\bar{\pi} \in \Pi^H$ randomly
 2: Start an instance of $\mathcal{M}$ $(\mathcal{A}, \delta/(4S))$ at every state $s \in \mathcal{S}$ (denote it by $\mathcal{M}_s$), and get first action
    $(a_s, a'_s)$
 3: for $h = H, H-1, ..., 1$ do
 4:     for $s_h \in \mathcal{S}_h$ do
 5:         Let $r_{s_h}(s, a) = 1_{s=s_h}$ for all $s \in S, a \in A$
 6:         Start an instance of EULER$(r_{s_h}, N_0, \delta/(4S))$
 7:         $U \leftarrow \emptyset$
 8:         for $n \in [N/S]$ do
 9:             Obtain a policy $\hat{\pi}_n$ from EULER, and execute $\hat{\pi}_n$ until step $h$
10:             Return the trajectory and reward to EULER
11:             $\tilde{s} \leftarrow$ current state
12:             Let $\bar{\pi} = a_{\tilde{s}} \circ \hat{\pi}_{h+1:H}$ if $|U| = 0$, otherwise $a'_{\tilde{s}} \circ \hat{\pi}_{h+1:H}$
13:             Execute $\bar{\pi}$ till step $H$, obtain trajectory $\tau$
14:             $U \leftarrow U \cup \tau$
15:             if $|U| = 2$ then
16:                 Compare the two trajectories in $U$ and return to $\mathcal{M}_{\tilde{s}}$
17:                 Get next action $(a_{\tilde{s}}, a'_{\tilde{s}})$ from $\mathcal{M}_{\tilde{s}}$
18:                 Update $\hat{\pi}_h(\tilde{s})$ to the optimal action according to $\mathcal{M}_{\tilde{s}}$
19:             end if
20:         end for
21:     end for
22: end for
Ensure: Policy $\hat{\pi}$
```

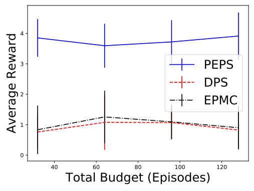
C Additional Experiment Details

C.1 Hyperparameters for Experiments

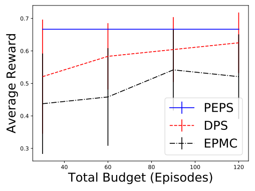
For PEPS, we search the hyperparameters for Q learning and Beat-the-Mean. This includes the learning rate of Q-learning (in range $\{0.1, 0.3, 1.0\}$), the ucb bound ratio for Q-learning (in range $\{0.01, 0.1, 1.0\}$), and the ucb bound ratio for Beat-the-Mean (in range $\{0.2, 0.5, 1.0\}$). We also allow the algorithm to random explore with probability in $\{0.05, 0.1, 0.2, 0.5\}$ during Q-learning. For DPS, we follow the default settings to use kernel variance 0.1 and kernel noise 0.001 for the Gaussian process regression. For EPMC, we use $\alpha = 0.2$ and $\eta = 0.8$.



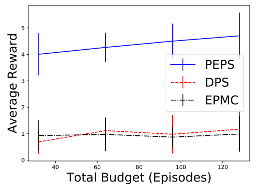
(a) Linear, GridWorld



(b) Linear, Random MDP

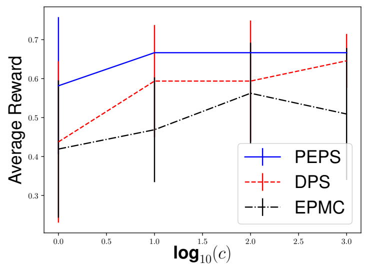


(c) Deterministic, GridWorld

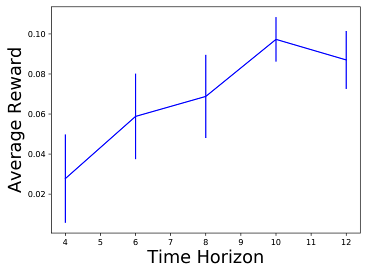


(d) Deterministic, Random MDP

Figure 2: Experiment results under linear preferences (a,b) and deterministic preferences (c,d).



(a) Varying c in BTL model



(b) Varying H

Figure 3: Performance dependence on BTL model parameter c, and time horizon H.

C.2 Additional Experiment results

Results under other preference models. We test the performance of PEPS and other baselines under the linear preference model and deterministic preferences in Figure 2. For linear preferences, we generate the preferences through a linear link function $\phi(\tau_1, \tau_2) = \alpha(r(\tau_1) - r(\tau_2))$, and we use $\alpha = 0.01$ in our experiment. This results in a very noisy preference model. PEPS performs similarly to the baselines in GridWorld, potentially because of the noisy preferences; but it still outperforms the baselines in random MDP. Under deterministic preferences, PEPS outperforms both baselines in both environments. **Dependence on parameters.** We test the performance versus various parameters in Figure 3. In Figure 3a we vary the BTL model parameter c. Our PEPS consistently outperforms the baselines. In Figure 3b we test the regret versus the time horizon H. It shows a close to linear relation, which fits our rate in Corollary 8 (the $O(H^2/\varepsilon^2)$ term translates to a linear dependence of ε on H).

D Proofs

D.1 Proof of Proposition 1

Proof. Let $S = 6$, $\mathcal{S} = \{s_0, ..., s_5\}$, and $A = 3$, $\mathcal{A} = \{a_0, a_1, a_2\}$, and let $H = 2$. We start at s_0. Let $r(s_0, a) = 0$ for all $a \in \mathcal{A}$. Executing a_0 in s_0 goes to s_1 w.p. 0.2, and to s_2 w.p. 0.8. Executing a_1 in s_0 goes to s_3 with probability 1. Executing a_2 in s_0 goes to s_4 w.p. 0.6 and to s_5 w.p. 0.4. For every action a, let $r(s_1, a) = 1, r(s_2, a) = 0.01, r(s_3, a) = 0.02, r(s_4, a) = 0.5, r(s_5, a) = 0$. See Figure 4 for a graphical explanation.

Let $\pi_i(s_0) = a_i$ for $i \in \{0, 1, 2\}$ (actions in other states do not matter). It is easy to verify that $\phi_{s_0}(\pi_0, \pi_1) = -0.3$, $\phi_{s_0}(\pi_1, \pi_2) = -0.1$, and $\phi_{s_0}(\pi_2, \pi_0) = -0.02$. $\square$

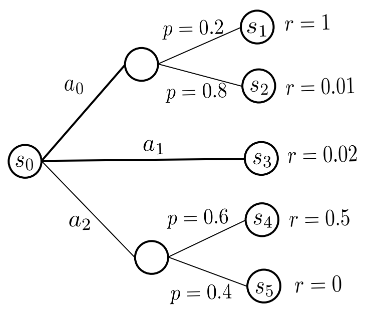


Figure 4: Example for proof of Proposition 1.

D.2 Proof of Proposition 2

Proof. For i), by the linearity of expectation we have for two random trajectories τ, τ', we have $E[\phi(\tau, \tau')] = E[C(r(\tau) - r(\tau'))]$. Therefore for two policies π_1, π_2

$$\begin{aligned}\phi_s(\pi_1, \pi_2) &= \Pr[\tau(\pi_1, s) \succ \tau(\pi_2, s)] - 1/2 \\ &= E[\phi(\tau(\pi_1, s), \tau(\pi_2, s))] \\ &= E[C(r(\tau(\pi_1, s)) - r(\tau(\pi_2, s)))] = C(v_s(\pi_1) - v_s(\pi_2)).\end{aligned}$$

For ii), when transitions are deterministic each policy corresponds to only one trajectory. Let τ_1 and τ_2 be the two trajectories corresponding to π_1 and π_2 starting at s. Then we have $\phi_s(\pi_1, \pi_2) = \phi(\tau_1, \tau_2)$. Then Assumption 1 is satisfied with $C_0 = c$. □

D.3 Proof of Theorem 3

Proof. We only need to show that the output policy $\hat{\pi}$ is ε-optimal. Algorithm 1 loops over $h = 1, 2, ..., H$. At every step h for every state $s \in \mathcal{S}_h$, let $\tilde{\pi}^*(s) = \arg\max_a V(s; a \circ \hat{\pi}_{h+1:H})$. Let P_h^* denote the state distribution of π^* at step h. With probability $1 - \delta$, all instances of $\mathcal{M}$ has finished. Under this event, from the property of $\mathcal{M}$ and the setup of N_0 we have for every state $s \in \mathcal{S}$, $\Pr[\phi_s(\hat{\pi}, \tilde{\pi}^*(s))] \geq 1/2 - \varepsilon_1$. Therefore from Assumption 1 we have

$$|v_{\tilde{\pi}^*(s)}(s) - v_{\hat{\pi}}(s)| \leq \frac{1}{C_0} \phi_s(\tilde{\pi}^*(s), \hat{\pi}) \leq \frac{\varepsilon}{H}.$$

Therefore using the performance difference lemma we have

$$\begin{aligned}V^{\pi^*}(s_0) - V^{\hat{\pi}}(s_0) &= \sum_{h=1}^{H} E_{s_h \sim P_h^*} \left[v(s_h; \pi_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\ &\leq \sum_{h=1}^{H} E_{s_h \sim P_h^*} \left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\ &\leq \sum_{h=1}^{H} \frac{\varepsilon}{H} = \varepsilon.\end{aligned}$$

□

D.4 Proof of Theorem 5

Proof. Let $\tilde{\mathcal{S}}_h = \{s \in \mathcal{S}_h | \mu(s) \geq \varepsilon/2S\}$ be the set of "good" states that are reachable with probability at least $\varepsilon/(2S)$. Also let $\tilde{\mathcal{S}} = \bigcup_h \tilde{\mathcal{S}}_h$ be the set of all good states.

Now we show that the output policy $\hat{\pi}$ is ε-optimal. We first present the performance of EULER [35]:

Theorem 11 (Theorem 1, [35]). *Let Q^* be the value of the optimal policy. For any reward function r, the regret of EULER is at most*

$$R \leq O(\sqrt{Q^* SATL} + \sqrt{S}SAH^2L^3(\sqrt{S} + \sqrt{H}))$$

with probability $1 - \delta$, with $L = \log(HSAT/\delta)$.

For any state s, let N_s be the number of times that we reach s in Step 12. Using Theorem 11 with $T = HN_0$, we have for some constant C,

$$\mu(s)N_0 - N_s \le C(\sqrt{\mu(s)SAHN_0L} + \sqrt{S}SAH^2L^3(\sqrt{S} + \sqrt{H})) \tag{1}$$

with probability $1 - \delta/4S$. By a union bound, suppose (1) holds for every state $s \in \mathcal{S}$, which happens with probability $1 - \delta/4$. Now consider any $s \in \tilde{\mathcal{S}}$. With $N_0 = \Omega(\frac{S^3AH^2\iota^3}{\varepsilon})$, we have $N_s \ge 1/2\mu(s)N_0$. Now also using $N_0 = \Omega\left(\frac{SH^\alpha\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^{\alpha+1}}\right)$, we have

$$N_s \ge \frac{1}{2}\mu(s)N_0 \ge \Omega\left(\frac{H^\alpha\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^\alpha}\right).$$

By setting the constant in N_0 properly, from the definition of Ψ we know that with probability $1 - \delta/4S$, $\mathcal{M}$ returns a $\frac{C_K\varepsilon}{H}$ optimal arm; here c_K is a constant to be specified later.

Algorithm 2 loops over $h = 1, 2, ..., H$. At every step h for every state $s \in \mathcal{S}_h$, let $\tilde{a}_s^* = \arg\max_a V(s; a \circ \hat{\pi}_{h+1:H})$. From the guarantees of OPT-Maximize we have

$$v(s_h; \tilde{a}_s^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \le \frac{1}{C_0}\phi_s(\tilde{a}_s^*, \hat{\pi}_{h+1:H}) \le \frac{\varepsilon}{2H}.$$

The first inequality comes from Assumption 1; we set c_K small enough to satisfy the latter inequality.

Let P_h^* denote the state distribution of π^* at step h. We have

$$\begin{aligned}
V^{\pi^*}(s_0) - V^{\hat{\pi}}(s_0) &= \sum_{h=1}^H E_{s_h\sim P_h^*}\left[v(s_h; \pi_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\le \sum_{h=1}^H E_{s_h\sim P_h^*}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\le \sum_{h=1}^H \Pr_{\pi^*}[s_h \notin \tilde{\mathcal{S}}] + \Pr[s_h \in \tilde{\mathcal{S}}]E_{s_h\sim P_h^*, s_h\in\tilde{\mathcal{S}}}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\le \varepsilon/(2S)\cdot S + \sum_{h=1}^H \varepsilon/(2H) \le \varepsilon.
\end{aligned}$$

Here the first equality is the performance difference lemma (Lemma 13, [24]). The first inequality comes from the definition of $\tilde{\pi}^*$; the third inequality comes from definition of $\tilde{\mathcal{S}}$, and the guarantee of $\mathcal{M}$. Therefore we show that the output policy is ε-optimal.

Now we compute the sample complexity. It is obvious that the step complexity is HSN_0 since we iterate through all $s \in \mathcal{S}$, and each episode contains H steps. For comparison complexity, we only need to finish S instances of $\mathcal{M}$; therefore the comparison complexity is $O\left(\frac{H^\alpha S\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^\alpha}\right)$.

□

D.5 Proof of Theorem 7

Proof. The proof follows most parts of that of 5. For any $s \in \tilde{\mathcal{S}}$, we have $N_s \ge \frac{1}{2}\mu(s)N_0$. Guarantee of Beat-the-Mean is as follows:

For simplicity, let $p = 1/\alpha$ and $q = (\alpha-1)/\alpha$. For $s_h \in \tilde{\mathcal{S}}_h$, let $\varepsilon_{s_h} = \frac{\varepsilon}{2\mu^p(s_h)S^pH^q}$. For $N_0 = \Omega\left(\frac{H^{\alpha-1}S\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha}\right)$, we have

$$N_{s_h} \ge 1/2\mu(s_h)N_0 = \Omega\left(\frac{\mu(s_h)H^{\alpha-1}S\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha}\right) \ge \Omega\left(\Psi(A,\varepsilon_{s_h},\delta/4S)\varepsilon_{s_h}^{-\alpha}\right).$$

Similar to proof of Theorem 5, we can set the constants properly to obtain that

$$v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \le \varepsilon_{s_h} \tag{2}$$

with probability $1-\delta/4S$. Now suppose (2) holds for every $h \in [H]$ and $s_h \in \mathcal{S}_h$, which holds with probability $1-\delta$. We have

$$\begin{aligned}
V^{\pi^*}(s_0) - V^{\hat{\pi}}(s_0) &= \sum_{h=1}^{H} E_{s_h \sim P_h^*}\left[v(s_h; \pi_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\le \sum_{h=1}^{H} E_{s_h \sim P_h^*}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\le \sum_{h=1}^{H} \Pr_{\pi^*}[s_h \notin \tilde{\mathcal{S}}] + \Pr_{\pi^*}[s_h \in \tilde{\mathcal{S}}] E_{s_h \sim P_h^*, s_h \in \tilde{\mathcal{S}}}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\le \varepsilon/(2S) \cdot S + \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h)\left(v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right) \\
&\le \varepsilon/2 + \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h)\varepsilon_{s_h}.
\end{aligned}$$

Here the first equality is the performance different lemma (Lemma 13, [24]), and P_h^* is the state distribution of π^* on step h. The first inequality comes from the definition of $\tilde{\pi}^*$. Now note that $P_h^*(s_h) \le \mu(s_h)$, and that $\sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h) = H$; we have

$$\begin{aligned}
\sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h)\varepsilon_{s_h} &= \frac{\varepsilon}{2S^p H^q} \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h)\mu^{-p}(s_h) \\
&\le \frac{\varepsilon}{2S^p H^q} \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} (P_h^*(s_h))^{1-p} \le \varepsilon/2.
\end{aligned}$$

The inequality holds from Hölder's inequality. Therefore we show that the output policy is ε-optimal.

Now we compute the sample complexity. The step complexity is simply $O(HSN_0)$. For comparison complexity, we roll out at most SN_0 trajectories, so the comparison complexity is at most SN_0.

□

D.6 Proof of Theorem 9

Proof. The proof of Theorem 9 is largely the same as Theorem 7. For every state s, let $\tilde{\mu}(s)$ be the probability that $\hat{\pi}_s$ visits s. Similar to Theorem 5 and 7, we have $N_s \ge 1/2\mu(s)N_0$ for $s \in \tilde{\mathcal{S}}$. Therefore by randomly pick a policy from the N_0 episodes, we have $\tilde{\mu}(s) \ge 1/2\mu(s)$.

Using Hoeffding's inequality and property of EULER we have that with probability $1-\delta/2$, for every state s we have

$$\hat{R}_{s_h}/N_1 \ge \tilde{\mu}(s) - \sqrt{\frac{\log(4S/\delta)}{N_0}} \ge 1/2\mu(s) - \sqrt{\frac{\log(4S/\delta)}{N_0}},$$

and therefore $\mu(s) \le \hat{\mu}(s)$. On the other hand, we also have

$$\hat{\mu}(s) \le 2\hat{R}_{s_h}/N_1 + 2\sqrt{\frac{\log(4S/\delta)}{N_0}} \le 2\tilde{\mu}(s) + 4\sqrt{\frac{\log(4S/\delta)}{N_0}} \le 2\mu(s) + 4\varepsilon/S \le 6\mu(s).$$

Define $\varepsilon'_{s_h} \leftarrow \frac{\varepsilon}{2(\mu(s_h)SH^{\alpha-1})^{1/\alpha}}$ as in Theorem 7. Therefore we know that with probability $1-\delta/2$, we have $\varepsilon_{s_h} \le \varepsilon'_{s_h}$ and that $\varepsilon_{s_h} = \Theta(\varepsilon'_{s_h})$. The rest of the proof follows the same process as Theorem 7.

□

E Auxiliary Lemma

We present the performance difference lemma here for completeness. Here we use the version adapted to episode MDPs as in [24].

Lemma 12 (Lemma 13, [24])**.** *For any episode MDP with reward function r and two policies $\pi_{0:H-1}$ and $\pi'_{0:H-1}$, For any $h \in [H]$, let $Q_h(s)$ be the distribution of state s at step h induced by policy $\pi_{0:H-1}$. We have*

$$v_{\pi_{0:H-1}}(s_0) - v_{\pi'_{0:H-1}}(s_0) = \sum_{h=0}^{H-1} E_{s\sim Q_h(s)}[V_{\pi_h \circ \pi'_{h+1:H}}(s) - V_{\pi'_{h:H}}(s)].$$