Heuristic-Guided Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We provide a framework to accelerate reinforcement learning (RL) algorithms by heuristics that are constructed by domain knowledge or offline data. Tabula rasa RL algorithms require environment interactions or computation that scales with the horizon of the sequential decision-making task. Using our framework, we show how heuristic-guided RL induces a much shorter horizon sub-problem that provably solves the original task. Our framework can be viewed as a horizon-based regularization for controlling bias and variance in RL under a finite interaction budget. In theory, we characterize the properties of a good heuristic and the resulting impact on RL acceleration. In particular, we introduce the novel concept of an improvable heuristic that can allow any RL agent to conservatively extrapolate beyond its prior knowledge. In practice, we instantiate our framework to accelerate several state-of-the-art algorithms in simulated robotic control tasks and procedurally generated games. Our framework complements the rich literature on warm-starting RL using expert demonstrations or exploratory data-sets, and creates a unified channel to inject prior knowledge into RL.

1 Introduction

Many recent empirical successes of reinforcement learning (RL) require solving problems with very long decision making horizons. OpenAI Five [1] used episodes that were 20000 timesteps on average, while AlphaStar [2] used roughly 5000 timesteps. Long-term credit assignment is a very challenging statistical problem, with the sample complexity of required environment interactions growing quadratically (or worse) with the horizon [3]. Computational concerns also arise in long-horizon regimes and manifests as slow convergence in optimization [4]. Consequently RL algorithms typically require prohibitively large amounts of interactions and compute—even with tuned hyperparameters, AlphaStar needed over 10^8 samples and OpenAI Five needed over 10^7 PFLOPS of compute.

A popular approach to ameliorate the statistical and computational issues of tabula rasa RL methods is to warm-start or regularize learning with prior knowledge [1, 2, 5–10]. For instance, AlphaStar learned a policy and value function from human demonstrations and regularized the RL agent using imitation learning (IL). AWAC [9] warm-started a policy using batch policy optimization on exploratory datasets. While these approaches have been effective in different domains, none of them address explicitly the horizon-dependent difficulty of RL.

In this paper, we propose a complementary regularization technique by using heuristic value functions, or simply *heuristics*, to effectively shorten the problem horizon faced by the online RL agent. We call this approach Heuristic-Guided Reinforcement Learning (HuRL). The core idea is simple: given a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ and a heuristic $f : \mathcal{S} \to \mathbb{R}$, we select a mixing coefficient $\lambda \in [0, 1]$ and have the agent solve a new MDP $\widetilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \widetilde{r}, \widetilde{\gamma})$ with a reshaped reward and a smaller discount:

$$\widetilde{r}(s, a) \coloneqq r(s, a) + (1 - \lambda)\gamma \mathbb{E}_{s'|s,a}[f(s')] \quad \text{and} \quad \widetilde{\gamma} \coloneqq \lambda\gamma, \tag{1}$$

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

where $\mathbb{E}_{s'|s,a}$ denotes the transition expectation of the original MDP $\mathcal{M}$. Thus HuRL introduces a horizon-based regularization, where we regularize whether the long term value information should come from collected experiences or the heuristic. By modulating the effective horizon via λ, we trade off the bias and the complexity of solving the reshaped MDP. HuRL with $\lambda = 1$ recovers the original problem and with $\lambda = 0$ creates a contextual bandit problem which can be solved efficiently [11].

A heuristic represents a prior guess of the desired long-term return of states, which ideally is the optimal value function V^* of the unknown MDP $\mathcal{M}$. How do we get a good heuristic? In the planning literature, this is typically achieved by relaxing the problem using domain knowledge. Another source is via batch data collected by exploratory behavioral policies as in batch RL or by expert policies as in IL.[1] In this paper, we view heuristics as a simple unified representation of various forms of prior knowledge, such as expert demonstrations, exploratory datasets, and engineered heuristics.

Although using heuristics to accelerate search has been popular in planning and control, such as A* [12], MCTS [13], and MPC [7, 14–16], its theory is less developed for the setting where the MDP is *unknown*. In this work, we provide a theoretical foundation of HuRL to enable adopting heuristics to accelerate an RL algorithm. Our framework subsumes the guidance discount literature [17–22] which effectively uses a constant heuristic. In many applications we can have a better guess of the long-term value in some states and that can lead to faster learning than guidance discount techniques.

In theory, we derive a bias-variance decomposition of HuRL's horizon-based regulation to characterize the solution quality as a function of λ and f. Using this insight, we provide sufficient conditions to achieve an effective trade-off, including properties needed from a base RL algorithm that solves the reshaped MDP $\widetilde{\mathcal{M}}$. Furthermore, we define the novel concept of an improvable heuristic and prove that good heuristics for HuRL can be constructed from data using existing *pessimistic* offline RL algorithms (such as pessimistic value iteration [23, 24]).

The effectiveness of HuRL depends on the heuristic quality, so we design HuRL to adopt a sequence of mixing coefficients (i.e. λ) that increase as the agent gathers more data from the environment. Such a strategy induces a learning curriculum that makes HuRL remain robust to non-ideal heuristics. HuRL starts off by guiding the agent's search direction with a heuristic. As the agent becomes more experienced, it gradually removes the guidance and lets the agent directly optimize the true long-term return. We empirically validate HuRL in MuJoCo [25] robotics control problems and Procgen games [26] with various heuristics and base RL algorithms. The experimental results demonstrate the versatility and effectiveness of HuRL in accelerating many RL algorithms.

2 Preliminaries

2.1 Notation

We focus on discounted infinite-horizon Markov Decision Processes (MDPs) for ease of exposition. The technique proposed here can be extended to other MDP settings. A discounted infinite-horizon MDP is denoted as a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s,a)$ is the transition dynamics, $r(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. Without loss of generality, we assume $r : \mathcal{S} \times \mathcal{A} \to [0,1]$; we allow the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ to be either discrete or continuous. Let $\Delta(\cdot)$ denote the space of probability distributions. A decision making policy π is a conditional distribution $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, which can be deterministic. We define some shorthand for writing expectations: For a state distribution $d \in \Delta(\mathcal{S})$ and a function $V : \mathcal{S} \to \mathbb{R}$, we define $V(d) \coloneqq \mathbb{E}_{s\sim d}[V(s)]$; similarly, for a policy π and a function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, we define $Q(s,\pi) \coloneqq \mathbb{E}_{a\sim\pi(\cdot|s)}[Q(s,a)]$. Lastly, we define $\mathbb{E}_{s'|s,a} \coloneqq \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}$.

Central to solving MDPs are the concepts of value functions and average distributions. For a policy π, we define its state value function V^π as $V^\pi(s) \coloneqq \mathbb{E}_{\rho_s^\pi}\left[\sum_{t=0}^\infty \gamma^t r(s_t,a_t)\right]$, where ρ_s^π denotes the trajectory distribution of $s_0, a_0, s_1, \ldots$ induced by running π starting from $s_0 = s$. We define the state-action value function (or the Q-function) as $Q^\pi(s,a) \coloneqq r(s,a) + \gamma\mathbb{E}_{s'|s,a}[V^\pi(s')]$. We denote the optimal policy as π^* and its state value function as $V^* \coloneqq V^{\pi^*}$. Under the bounded reward assumption, we have $V^\pi(s), Q^\pi(s,a) \in [0, \frac{1}{1-\gamma}]$ for all π, $s \in \mathcal{S}$, and $a \in \mathcal{A}$. We denote the initial state distribution of interest as $d_0 \in \Delta(\mathcal{S})$ and the state distribution of the policy π at time t as d_t^π with $d_0^\pi = d_0, \forall\pi$. Given d_0, we define the average state distribution of a policy π as $d^\pi \coloneqq (1-\gamma)\sum_{t=0}^\infty \gamma^t d_t^\pi$. With slight abuse of notation, we also write $d^\pi(s,a) \coloneqq d^\pi(s)\pi(a|s)$.

[1] We consider the RL setting where we suppose the rewards of expert trajectories are available.

Algorithm 1 Heuristic-Guided Reinforcement Learning (HuRL)

Require: MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, RL algorithm $\mathcal{L}$, heuristic f, mixing coefficients $\{\lambda_n\}$.

1: **for** $n = 1, \ldots, N$ **do**
2: $\quad \mathcal{D}_n \leftarrow \mathcal{L}$.CollectData($\mathcal{M}$)
3: $\quad$ Get λ_n from $\{\lambda_n\}$ and set $\widetilde{\mathcal{M}}_n = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \widetilde{r}_n, \widetilde{\gamma}_n)$ according to (2) using f and λ_n
4: $\quad \pi_n \leftarrow \mathcal{L}$.Train($\mathcal{D}_n$, $\widetilde{\mathcal{M}}_n$)
5: **end for**
6: **return** π_N

2.2 Setup: Reinforcement Learning with Heuristics

We consider RL with prior knowledge expressed in the form of a heuristic. The goal is to find a policy π that has high return through interactions with an unknown MDP $\mathcal{M}$, i.e., $\max_\pi V^\pi(d_0)$. While the agent here does not fully know $\mathcal{M}$, we suppose that, before interactions start the agent is provided with a heuristic $f : \mathcal{S} \to \mathbb{R}$ which the agent can query throughout learning.

The heuristic f represents a prior guess of the optimal value function V^* of $\mathcal{M}$. Common sources of heuristics are domain knowledge as typically employed in planning, and logged data collected by exploratory or by expert behavioral policies. In the latter, a heuristic guess of V^* can be computed from the data by offline RL algorithms. For instance, when we have trajectories of an expert behavioral policy, Monte-Carlo regression estimate of the observed returns may be a good guess of V^*.

Using heuristics to solve MDP problems has been popular in planning and control, but its usage is rather limited in RL. The closest provable technique in RL is potential-based reward reshaping (PBRS) [27], where the reward is modified into $\overline{r}(s, a) \coloneqq r(s, a) + \gamma \mathbb{E}_{s'|s,a}[f(s')] - f(s)$. It can be shown that this transformation does not introduce bias into the policy ordering, and therefore solving the new MDP $\overline{\mathcal{M}} \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{P}, \overline{r}, \gamma)$ would yield the same optimal policy π^* of $\mathcal{M}$. However, PBRS does not fully leverage information provided by the heuristic because it still requires the learning agent to solve a long horizon problem, which is the main source of difficulty in RL. Conceptually when the heuristic is the optimal value function $f = V^*$, the agent should be able to find the optimal policy π^* of $\mathcal{M}$ by acting myopically, as V^* already contains all necessary long-term information for good decision making. However RL with PBRS does not enable such a solution concept.

3 Heuristic-Guided Reinforcement Learning

We propose a general framework, HuRL, for leveraging heuristics to accelerate RL. In contrast to tabula rasa RL algorithms that attempt to directly solve the long-horizon MDP $\mathcal{M}$, HuRL uses a heuristic to guide the agent to solve a sequence of short-horizon MDPs so as to amortize the complexity of long-term credit assignment. In effect, HuRL creates a heuristic-based learning curriculum to help the agent learn faster.

3.1 Algorithm

HuRL takes a reduction approach to realize the idea of heuristic guidance. As summarized in Algorithm 1, HuRL takes a heuristic $f : \mathcal{S} \to \mathbb{R}$ and a base RL algorithm $\mathcal{L}$ as input and outputs an approximately optimal policy for the MDP $\mathcal{M}$. During training, HuRL iteratively runs the base algorithm $\mathcal{L}$ to collect data from the MDP $\mathcal{M}$ and then uses the heuristic f to reshape the agent's experiences. In iteration n, the agent collects data in the original MDP $\mathcal{M}$ and saves the raw transition tuples[2] $\mathcal{D}_n = \{(s, a, r, s')\}$ (line 2). Then HuRL defines a reshaped MDP $\widetilde{\mathcal{M}}_n \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{P}, \widetilde{r}_n, \widetilde{\gamma}_n)$ (line 3) by reshaping the reward and further lowering the discount

$$\widetilde{r}_n(s, a) \coloneqq r(s, a) + (1 - \lambda_n)\gamma \mathbb{E}_{s'|s,a}[f(s')] \qquad \text{and} \qquad \widetilde{\gamma}_n \coloneqq \lambda_n \gamma, \tag{2}$$

where $\lambda_n \in [0, 1]$ is the mixing coefficient. The new discount $\widetilde{\gamma}_n$ makes $\widetilde{\mathcal{M}}_n$ have a shorter horizon, while the heuristic f is blended into the new reward in (2) proportional to $1 - \lambda_n$ to account for the missing long-term information. We call $\widetilde{\gamma}_n = \lambda_n \gamma$ in (2) the *guidance discount* to be consistent with prior literature [19] which effectively uses a constant heuristic in view of our framework. In the

[2] If $\mathcal{L}$ learns only with trajectories, we transform each tuple and assemble them to get the modified trajectory.

last step (line 4), HuRL calls the base algorithm $\mathcal{L}$ to perform updates with respect to the reshaped MDP $\widetilde{\mathcal{M}}_n$. This is realized by *1)* setting the discount factor used in $\mathcal{L}$ as $\widetilde{\gamma}_n$, and *2)* setting the sampled reward as $r + (\gamma - \widetilde{\gamma}_n) f(s')$ for a transition tuple (s, a, r, s') of $\mathcal{M}$. We remark that the base algorithm $\mathcal{L}$ in line 2 always collect of trajectories of lengths proportional to the original discount γ, while internally the optimization is done with a lower discount $\widetilde{\gamma}_n$ in line 4.

Over the course of training, HuRL repeats the above steps with a sequence of increasing mixing coefficients $\{\lambda_n\}$. From (2), as the agent interacts with the environment, we see that the effects of the heuristic in MDP reshaping decrease and the effective horizon of the reshaped MDP increases.

3.2 HuRL as Horizon-based Regularization

We can think of HuRL as introducing a horizon-based *regularization* to RL, where the regularization center is defined by the heuristic and its strength diminishes as the mixing coefficient increases. As the agent gains more experiences, HuRL gradually removes the effects the regularization and the agent would eventually solve the original MDP.

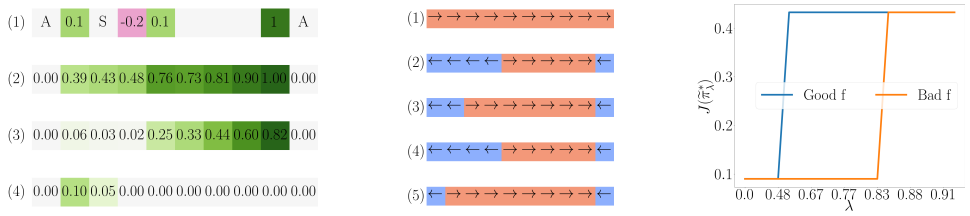
The regularization of HuRL is designed to reduce the learning variance, similar to the role of regularization in supervised learning. Unlike the typical weight decay imposed on function approximators (such as the agent's policy or value networks), our proposed regularization leverages the spatial structure of MDPs to regulate the complexity of the MDP the agent solves, as the complexity of an MDP scales with its discount factor (or equivalently the horizon).

When the guidance discount $\widetilde{\gamma}_n$ is lower than the original discount γ (i.e. $\lambda_n < 1$), the reshaped MDP $\widetilde{\mathcal{M}}_n$ given by (2) has a shorter horizon and requires less samples to solve. However, the reduced complexity comes at the cost of bias, because the agent is now incentivized toward maximizing the performance with respect to the heuristic rather than the long-term value of the original MDP $\mathcal{M}$. In the extreme where $\lambda_n = 0$, the reshaped MDP can be viewed as a zero-horizon contextual bandit problem with contexts sampled from the states visited by the agent in the original MDP $\mathcal{M}$.

3.3 A Toy Example

We visually demonstrate this idea in a chain MDP environment in Fig. 1. $\mathcal{S} = \{1, \ldots, 10\}$, the agent starts at $s_0 = 3$, and states 1 and 10 are absorbing states. Actions $\mathcal{A} = \{\leftarrow, \rightarrow\}$ move the agent left or right in the chain unless the agent is in an absorbing state. Rewards on the chain are illustrated in Fig. 1a-(1) and $\gamma = 0.9$. The optimal policy π^* for this MDP always picks action $\rightarrow$, as illustrated in Fig. 1b-(1), giving the optimal value V^* in Fig. 1a-(2). Suppose we used a smaller guidance discount $\widetilde{\gamma} = 0.5\gamma$ to accelerate learning. This is equivalent to HuRL with a zero heuristic $f = 0$ and $\lambda = 0.5$. Solving this reshaped MDP yields a policy $\widetilde{\pi}^*$ that acts very myopically in the original MDP, as shown in Fig. 1b-(2); the value function of $\widetilde{\pi}^*$ in the original MDP is visualized in Fig. 1a-(4).

Now, suppose we use Fig. 1a-(4) as a heuristic in HuRL instead of $f = 0$. This is a bad choice of heuristic (Bad f) as it introduces a large bias with respect to V^* (cf. Fig. 1a-(2)). On the other hand, we can rollout a random policy in the original MDP and use its value function as the heuristic (Good f), shown in Fig. 1a-(3). Though the random policy has an even *lower* return at the initial state $s = 3$, it gives a *better* heuristic because this heuristic shares the same trend as V^* in Fig. 1a-(1). HuRL run with Good f, Bad f yields policies in Fig. 1b-(3,4), and the quality of the resulting solutions in the original MDP, $J(\widetilde{\pi}^*_\lambda) := V^{\widetilde{\pi}^*}(d_0)$, is reported in Fig. 1c for different λ. Observe that HuRL with a



(a) Heatmap of different values. (b) Different policy behaviors. (c) HuRL with different f and λ.

Figure 1: Toy Example of HuRL in a chain MDP. Using good f even a small λ can find π^*. In (a), (1) r, (2) V^*, (3) Good f : V(rand π), (4) Bad f : V(myopic π). In (b), (1) π^*,(2) ($f = 0, \lambda = 0.5$), (3) (Good $f, \lambda = 0.5$), (4) (Bad $f, \lambda = 0.5$), (5) (Bad $f, \lambda = 1$).

good heuristic is able to achieve $V^*(d_0)$ with a substantially smaller horizon $\lambda \leq 0.5$. Bad f does not find π^* when $\lambda = 0.5$ (Fig. 1b-(4)) but is guaranteed to do so when λ converges to 1 (Fig. 1b-(5)).

4 Theoretical Analysis

When can HuRL accelerate learning? Similar to typical regularization techniques, the horizon-based regularization of HuRL leads to a bias-variance decomposition that can be optimized for better finite-sample performance, compared with directly solving the original MDP. However, a non-trivial trade-off is possible only when the regularization can bias the learning toward a good direction. In our case, this is determined by the heuristic which resembles the prior we encode into learning.

We provide the theoretical foundation for HuRL in this section. In the following, we first describe the bias-variance trade-off made by HuRL. We show how the suboptimality in solving the reshaped MDP is translated into the performance of the original MDP as well as the assumptions HuRL requires from the base RL algorithm. In addition we characterize the quality of heuristics, and provide sufficient conditions to construct good heuristics from batch data using offline RL algorithms.

For clarity, we will focus on the reshaped MDP $\widetilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \widetilde{r}, \widetilde{\gamma})$ for a fixed $\lambda \in [0, 1]$, where $\widetilde{r}, \widetilde{\gamma}$ are defined in (1). We can view this reshaped MDP as the one used in an iteration of HuRL. For a policy π, we denote its state value function of $\widetilde{\mathcal{M}}$ as $\widetilde{V}^\pi$, and the optimal policy and value function of $\widetilde{\mathcal{M}}$ as $\widetilde{\pi}^*$ and $\widetilde{V}^*$, respectively. The missing proofs of this section can be found in Appendix A.

4.1 Short Horizon Reduction: Performance Decomposition

Our main result is a performance decomposition, which characterizes how the heuristic f and the suboptimality of solving the reshaped MDP $\widetilde{\mathcal{M}}$ relate to the performance in the original MDP $\mathcal{M}$.

Theorem 4.1. *For* any *policy π, heuristic $f : \mathcal{S} \to \mathbb{R}$ and mixing coefficient $\lambda \in [0, 1]$,*

$$V^*(d_0) - V^\pi(d_0) = \mathrm{Regret}(f, \lambda, \pi) + \mathrm{Bias}(f, \lambda, \pi)$$

where we define

$$\mathrm{Regret}(f, \lambda, \pi) \coloneqq \lambda \left(\widetilde{V}^*(d_0) - \widetilde{V}^\pi(d_0) \right) + \frac{1-\lambda}{1-\gamma} \left(\widetilde{V}^*(d^\pi) - \widetilde{V}^\pi(d^\pi) \right) \tag{3}$$

$$\mathrm{Bias}(f, \lambda, \pi) \coloneqq \left(V^*(d_0) - \widetilde{V}^*(d_0) \right) + \frac{\gamma(1-\lambda)}{1-\gamma} \mathbb{E}_{s,a \sim d^\pi} \mathbb{E}_{s'|s,a} \left[f(s') - \widetilde{V}^*(s') \right] \tag{4}$$

Furthermore, $\forall b \in \mathbb{R}$, $\mathrm{Bias}(f, \lambda, \pi) = \mathrm{Bias}(f + b, \lambda, \pi)$ *and* $\mathrm{Regret}(f, \lambda, \pi) = \mathrm{Regret}(f + b, \lambda, \pi)$.

The theorem shows that the suboptimality of a policy π in the original MDP $\mathcal{M}$ can be decomposed into *1)* a *bias* term due to solving the reshaped MDP $\widetilde{\mathcal{M}}$ instead of original MDP $\mathcal{M}$, and *2)* a *regret* term (i.e. the learning variance) due to π being suboptimal in the reshaped MDP $\widetilde{\mathcal{M}}$. Moreover, it shows that heuristics are equivalent up to constant offsets. In other words, only the relative ordering between states that a heuristic induces matters in learning, not the absolute values.

Balancing the two terms trades off bias and variance in learning. Conceptually, using a smaller λ would replace the long-term information with the heuristic and make the horizon of the reshaped MDP $\widetilde{\mathcal{M}}$ shorter. Therefore, given a finite interaction budget, the regret term in (3) can be more easily minimized, though the bias term in (4) can potentially be large if the heuristic is bad. On the contrary, with $\lambda = 1$, the bias is completely removed, as the agent solves directly the original MDP $\mathcal{M}$. We discuss in details how these regret and bias terms interact with the base RL algorithm $\mathcal{L}$, the heuristic f, and the mixing coefficient λ below.

4.2 Regret and Algorithm Requirement

The regret term in (3) characterizes the performance gap due to π being suboptimal in terms of the reshaped MDP $\widetilde{\mathcal{M}}$, because $\mathrm{Regret}(f, \lambda, \widetilde{\pi}^*) = 0$ for all f and λ. For learning, we need the base RL algorithm $\mathcal{L}$ to find a policy π such that the regret term in (3) is small. By the definition in (3), the base RL algorithm $\mathcal{L}$ is required not only to find a policy π such that $\widetilde{V}^*(s) - \widetilde{V}^\pi(s)$ is small for states from d_0, but also for states π visits when rolling out in the original MDP $\mathcal{M}$. In other words, it is insufficient for the base RL algorithm to only optimize for $\widetilde{V}^\pi(d_0)$ (the performance of

the reshaped MDP with respect to the initial state distribution). To see this more clearly, suppose $\lambda = 0$ and the initial state is uniquely s_0. Then maximizing $\widetilde{V}^\pi(d_0)$ alone would only optimize $\pi(\cdot|s_0)$ and the policy π need not know how to act in other parts of the state space.

To use HuRL, we need the base algorithm to learn a policy π that has small *action gaps* of the reshaped MDP $\widetilde{\mathcal{M}}$ but along trajectories in the original MDP $\mathcal{M}$, as we show below. This property is satisfied by off-policy RL algorithms such as Q-learning [28].

Proposition 4.1. *For* any *policy π, heuristic $f : \mathcal{S} \to \mathbb{R}$ and mixing coefficient $\lambda \in [0, 1]$,*

$$\mathrm{Regret}(f, \lambda, \pi) = -\mathbb{E}_{\rho^\pi(d_0)}\left[\sum_{t=0}^{\infty} \gamma^t \widetilde{A}^*(s, a)\right]$$

where $\rho^\pi(d_0)$ denotes the trajectory distribution of running π from d_0, and $\widetilde{A}^*(s, a) = \widetilde{r}(s, a) + \widetilde{\gamma}\mathbb{E}_{s'|s,a}[\widetilde{V}(s')] - \widetilde{V}^*(s) \leq 0$ is the action gap with respect to the optimal policy $\widetilde{\pi}^*$ of $\widetilde{\mathcal{M}}$.

Another way to comprehend the regret term is through studying its dependency on λ. When $\lambda = 1$, $\mathrm{Regret}(f, 0, \pi) = V^*(d_0) - V^\pi(d_0)$, which is identical to the policy regret of $\mathcal{M}$ for the *fixed* initial distribution d_0. On the other hand, when $\lambda = 0$, $\mathrm{Regret}(f, 0, \pi) = \max_{\pi'} \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^\pi}[\widetilde{r}(s, \pi') - \widetilde{r}(s, \pi)]$, which is the regret of a *non-stationary* contextual bandit problem, where the context distribution is d^π (the average state distribution of π). In general, with $\lambda \in (0, 1)$, it mixes a short-horizon non-stationary problem and a long-horizon stationary problem.

One natural question is whether the reshaped MDP $\widetilde{\mathcal{M}}$ has a more complicated and larger value landscape than the original MDP $\mathcal{M}$, because these characteristics may affect the regret rate of a base algorithm. We show that $\widetilde{\mathcal{M}}$ preserves the value bounds and linearity of the original MDP.

Proposition 4.2. *Reshaping the MDP as* (1) *preserves the following characteristics: 1) If $f(s) \in [0, \frac{1}{1-\gamma}]$, then $\widetilde{V}^\pi(s) \in [0, \frac{1}{1-\gamma}]$, for all π and $s \in \mathcal{S}$. 2) If $\widetilde{\mathcal{M}}$ is a linear MDP with feature $\phi(s, a)$ (i.e. $r(s, a)$ and $\mathbb{E}_{s'|s,a}[g(s')]$, for any g, can be linearly parametrized in $\phi(s, a)$), then $\widetilde{\mathcal{M}}$ is also a linear MDP with feature $\phi(s, a)$.*

4.3 Bias and Heuristic Quality

The bias term in (4) characterizes the suboptimality due to using a heuristic f in place of the long-term value of $\mathcal{M}$. What is the best heuristic in this case? From the definition of the bias term in (4), we see tha the ideal heuristic is the optimal value V^*, as $\mathrm{Bias}(V^*, \lambda, \pi) = 0$ for all $\lambda \in [0, 1]$. By continuity, we can expect that if f deviates from V^* a little, then the bias is small.

Corollary 4.1. *If $\inf_{b\in\mathbb{R}} \|f + b - V^*\|_\infty \leq \epsilon$, then $\mathrm{Bias}(f, \lambda, \pi) \leq \epsilon\left(\frac{2(1-\lambda)\gamma}{1-\gamma} + \frac{(1-\lambda)^2\gamma^2}{(1-\gamma)^2}\right)$*

To better understand how the heuristic f affects the bias, we derive an upper bound on the bias, by replacing the first term in (4) with an upper bound that depends only on π^*.

Proposition 4.3. *For $g : \mathcal{S} \to \mathbb{R}$ and $\eta \in [0, 1]$, define $\mathcal{C}(\pi, g, \eta) \coloneqq \mathbb{E}_{\rho^\pi(d_0)}\left[\sum_{t=1}^{\infty} \eta^{t-1} g(s_t)\right]$.*

$$\mathrm{Bias}(f, \lambda, \pi) \leq (1 - \lambda)\gamma\left(\mathcal{C}(\pi^*, V^* - f, \lambda\gamma) + \mathcal{C}(\pi, f - \widetilde{V}^*, \gamma)\right)$$

In Proposition 4.3, the first term is the underestimation error of the heuristic f under the states visited by optimal policy π^* in the original MDP $\mathcal{M}$. Therefore, to minimize the first term in the bias, we would want the heuristic f to be large along the path π^* generates.

However, Proposition 4.3 also limits the heuristic from being arbitrarily large, because the second term in the bias in (4) (equivalently the second term in Proposition 4.3) desires the heuristic to underestimate the optimal value of the reshaped MDP $\widetilde{V}^*$. More precisely, the second term requires the heuristic to obey some form of spatial consistency. A quick intuition is the observation that if $f(s) = V^{\pi'}(s)$ for some π' or $f(s) = 0$, then $f(s) \leq \widetilde{V}^*(s)$ for all $s \in \mathcal{S}$. More generally, we show that if the heuristic is *improvable* with respect to the original MDP $\mathcal{M}$ (i.e. the heuristic value is lower than that of the max of Bellman backup), then $f(s) \leq \widetilde{V}^*(s)$. By Proposition 4.3, learning with an improvable heuristic in HuRL has a much smaller bias.

Definition 4.1. Define the Bellman operator $(\mathcal{B}f)(s, a) \coloneqq r(s, a) + \gamma\mathbb{E}_{s'|s,a}[f(s')]$. A heuristic function $f : \mathcal{S} \to \mathbb{R}$ is said to be *improvable* with respect to an MDP $\mathcal{M}$ if $\max_a(\mathcal{B}f)(s, a) \geq f(s)$.

Proposition 4.4. *If f is improvable with respect to $\mathcal{M}$, then $\widetilde{V}^*(s) \geq f(s)$, for all $\lambda \in [0, 1]$.*

4.4 Pessimistic Heuristics are Good Heuristics

While Corollary 4.1 shows that HuRL can handle an imperfect heuristic, this result is not ideal. The corollary depends on the ℓ_∞ approximation error, which can be difficult to control when state space is large. Here we provide a more refined characterization of good heuristics. We show that the concept of *pessimism* in the face of uncertainty provides a finer mechanism to control the approximation error of a heuristic and would allow us to remove the ℓ_∞-type error. This result is useful for constructing heuristics from data that does not have sufficient support.

By Proposition 4.3, we see that the source of the ℓ_∞ error is due to the second term in the bias upper bound, as it depends on the states that the agent's policy visits which can change during learning. To remove this dependency, we can use improvable heuristics (see Proposition 4.4), as they satisfy $f(s) \leq \widetilde{V}^*(s)$. Below we show that Bellman-consistent pessimism yields improvable heuristics.

Proposition 4.5. *Suppose $f(s) = Q(s, \pi')$ for some policy π' and function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that $Q(s, a) \leq (\mathcal{B}f)(s, a)$, $\forall s \in \mathcal{S}$, $a \in \mathcal{A}$. Then f is improvable and $f(s) \leq V^{\pi'}(s)$ for all $s \in \mathcal{S}$.*

The Bellman consistent pessimism in (4.5) essentially says that f is pessimistic with respect to the Bellman backup. This condition has been used as the foundation to design pessimistic off-policy RL algorithms, such as pessimistic value iteration [23] and algorithms based on pessimistic absorbing MDPs [24]. In other words, these pessimistic algorithms can be used to construct good heuristics with small bias in Proposition 4.3 from offline data. With such a heuristic, the bias upper bound would be simply $\mathrm{Bias}(f, \lambda, \pi) \leq (1 - \lambda)\gamma\mathcal{C}(\pi^*, V^* - f, \lambda\gamma)$. Therefore, as long as enough batch data are sampled from a distribution that covers states that π^* visits, these pessimistic algorithms can construct good heuristics with nearly zero bias for HuRL with high probability.

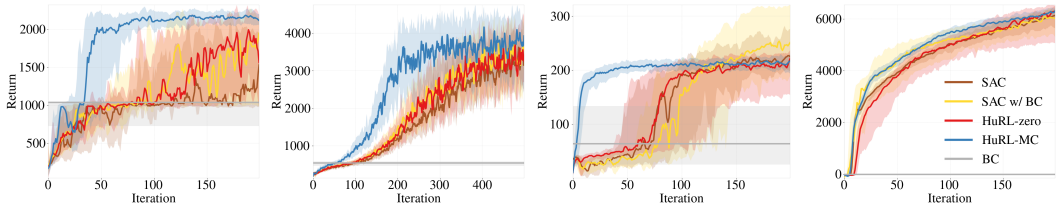
5 Experiments

We validate our framework HuRL experimentally in MuJoCo (commercial license) [25] robotics control problems and Procgen games (MIT License) [26], where soft actor critic (SAC) [29] and proximal policy optimization (PPO) [30] were used as the base RL algorithms, respectively. The goal is to study whether HuRL can accelerate learning by shortening the horizon with heuristics. In particular, we conduct studies to investigate the effects of different heuristics and mixing coefficients. Since the main focus here is on the possibility of leveraging a *given* heuristic to accelerate RL algorithms, in these experiments we used vanilla techniques to construct heuristics for HuRL. Experimentally studying the design of heuristics for a domain or a batch of data is beyond the scope of the current paper but are important future research directions; we show preliminary results with more advanced heuristic constructions in Appendix C. For space limitation, here we report only the results of the MuJoCo experiments. The results of Procgen games along with other experimental details can also be found in Appendix C.

5.1 Setup

We consider four MuJoCo environments (Hopper-v2, HalfCheetah-v2, Humanoid-v2, and Swimmer-v2). In these experiments, we suppose that a batch of data collected by multiple behavioral policies are available before learning, and a heuristic is constructed by an offline policy evaluation algorithm from the batch data; in the experiments, we generated these behavioral policies by running SAC from scratch and saved the intermediate policies generated in training. For simplicity, we use basic Monte Carlo regression to construct heuristics, where a least squares regression problem was used to fit a fully connected neural network to predict the empirical returns on the trajectories in the sampled batch of data. We also use behavior cloning (BC) to warm start RL agents based on the same batch dataset. The base RL algorithm here is SAC, which is based on the standard implementation of Garage (MIT License) [31]. The policy and the value networks are fully connected neural networks, independent of each other. The policy is Tanh-Gaussian and the value network has a linear head.

Algorithms We compare the performance of different algorithms below. *1)* BC *2)* SAC *3)* SAC with BC warm start (SAC w/ BC) *4)* HuRL with a zero heuristic and BC warm start (HuRL-zero) *5)* HuRL with the Monte-Carlo heuristic and BC warm start (HuRL-MC). For the HuRL algorithms, the mixing coefficient λ_n is scheduled as $\lambda_n = \lambda_0 + (1 - \lambda_0)\tanh(\omega(n - 1))$, for $n = 1, \ldots, N$, where $\lambda_0 \in [0, 1]$ is the initial λ and $\omega > 0$ controls the increasing rate. We chose these algorithms to illustrate the effect of each additional warm-start component (BC and heuristics) added on top of the



(a) Hopper-v2 (b) Humanoid-v2 (c) Swimmer-v2 (d) HalfCheetah-v2

Figure 2: Experimental results of different MuJoCo environments. The plots show the 25th, 50th, 75th percentiles of each algorithm's performance over 30 random seeds.

base algorithm SAC. HuRL-zero is SAC w/ BC but with an extra λ schedule described above that further lowers the discount, whereas SAC and SAC w/ BC keep a constant discount factor.

Evaluation and Hyperparameters In each iteration, the RL agent has a fixed sample budget for environment interactions, and its performance is measured in terms of the undiscounted accumulated rewards (estimated by 10 rollouts) of the deterministic mean policy extracted from SAC. The hyperparameters used in the algorithms above were selected as follows. First, the learning rates and the discount factor of the base RL algorithm, SAC, were tuned for each environment to maximize the performance. This tuned discount factor is used as the de factor discount factor γ of the original MDP $\mathcal{M}$. Fixing the hyperparameters above, λ_0 and ω for the λ schedule of HuRL were tuned for each environment and each heuristic. Finally, after all these hyperparameters were decided, we conducted additional testing runs with 30 different random seeds and report their statistics here. The randomness include the data collection process of the behavioral policies, training the heuristics from batch data, BC, and online RL, but the behavioral policies are fixed.

5.2 Summary Result

Fig. 2 shows the experimental results of the MuJoCo environments. Overall, we see that HuRL-MC is able to leverage a simple heuristic based on Monte-Carlo regression to significantly improve the learning efficiency. This trend is consistent across all four environments that we tested on, though the gain in HalfCheetah-v2 is minor. We also see that warm starting SAC using BC (i.e. SAC w/ BC) can improve the learning efficiency compared with the vanilla SAC, but using BC alone does not result in a good policy. Lastly, we see that using a zero heuristic (HuRL-zero) with additional λ-scheduling does not further improve the performance of SAC w/ BC. This comparison verifies that the learned Monte Carlo heuristic provides non-trivial information.

The results in Fig. 2 are quite encouraging and interesting. Because the datasets used in the experiments contain trajectories collected by a range of policies, it is likely that BC suffers from the disagreement in action-selection between different policies and results in the bad performance. Nonetheless, training a heuristic using a basic Monte-Carlo regression seems to be less sensitive to these conflicts and still results in a helpful heuristic for HuRL. One explanation can be that heuristics are only functions of states, not of states and actions, and therefore the conflicts are minor. Another plausible explanation is that HuRL only uses the heuristic to *guide* the learning, and does not completely rely on it to make decisions (as shown in Appendix C, all these algorithms selected a positive initial λ_0). Thus, HuRL can be more robust to the heuristic quality, or equivalently the quality of prior knowledge.

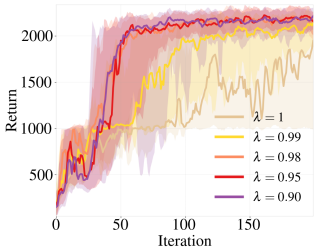


Figure 3: Ablation results of different initial λ_0 in Hopper-v2. The plots show the 25th, 50th, 75th percentiles of each algorithm's performance over 30 random seeds.

5.3 Ablation: Effects of Horizon Shortening

To further verify that the acceleration in Fig. 2 is indeed due to horizon shortening, we conducted an ablation study in Hopper-v2. Here we run HuRL-MC in Fig. 2a with a range of λ (Fig. 2a uses a constant $\lambda = 0.95$ throughout). We observe in Section 5.2 that increasing λ above 0.98 leads to a performance drop. Since using a large λ would decrease the bias and makes the reshaped MDP more similar to the original MDP, we conclude that the increased learning speed in Hopper-v2 is due to horizon shortening made by HuRL.

6 Related Work

HuRL combines discount regularization and reward-shaping to leverage a heuristic to accelerate RL. In particular, our reshaped MDP can be shown to be equivalent to another reshaped MDP with reward $\bar{r}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}[f(s')] - f(s)$ used in PBRS and a guidance discount $\gamma\lambda$. Therefore, HuRL can be viewed as incorporating guidance discount into PBRS.[3]

Discount regularization. The horizon-truncation idea can be traced back to Blackwell optimality in the known MDP setting [17]. Reducing the discount factor amounts to running HuRL with a trivial zero heuristic. [18–20] studies the known MDP setting; [21] studies POMDPs. [22] focuses on the impact of discount regularization on Temporal Difference (TD) methods, while [6] uses a logarithmic mapping for action values so as to use a lower discount for online RL.

Reward shaping. Reward shaping has a long history in RL, from the classic PBRS [27] to recent bilevel-optimization approaches [32]. [5] considers a complementary problem to HuRL: given a discount γ', they find a reshaped reward r' that preserves ordering of trajectories in the original MDP. Meanwhile there is a vast literature on bias-variance trade-off for online RL with horizon truncation. TD(λ) [33, 34] and Generalized Advantage Estimates [35] blend value estimates across many discount factors, while [36] uses the discount factor as an input to the value function estimator. TD(Δ) [37] computes differences between value functions across discount factors for stable estimation.

Heuristics in model-based methods. Classic uses of heuristics include A* [12], Monte-Carlo Tree Search (MCTS) [13], and Model Predictive Control (MPC) [38]. [14] shortens the horizon in MPC by a value function approximator. [15] additionally uses an estimate for the running cost to trade off solution quality with computation. [16] demonstrates heuristic-accelerated truncated horizon MPC on actual robots and tunes the value function throughout learning. [7] augments MPC with a terminal value heuristic and can be viewed as an instance of HuRL where the base algorithm is MPC.

Pessimistic extrapolation. Offline RL techniques employ pessimistic extrapolation for robust learning [23], and the value functions learned by these techniques can be used as heuristics in HuRL. [39] penalizes out-of-distribution actions in off-policy optimization while [24] additionally uses a variational auto-encoder to detect out-of-distribution states. We experimented with VAE-filtered pessimistic heuristics in Appendix C. Even pessimistic offline evaluation techniques [40] can be useful in HuRL since function approximation often induces extrapolation errors [41].

Other warm-starting techniques. HuRL is a new way to warm start online RL methods. [42] uses a heuristic policy to initialize agents' policies. [2, 10] train value and policy using batch IL and then used them as regularization in online RL. [9] uses off-policy RL on the batch data and fine-tune the learned policy. Recent approaches of hybrid IL-RL has a strong connections with HuRL [43–45]. In particular, [44] is a special case of HuRL with a max-aggregation heuristic. Finally [8] uses the several related tasks to learn a task-dependent heuristic and perform shorter horizon planning or RL.

7 Discussion and Limitation

This work is an early step towards theoretically understanding the role and potential of heuristics in guiding RL algorithms. We propose a framework, HuRL, that can accelerate RL when an informative heuristic is provided. HuRL induces a horizon-based regularization to RL, complementary to other existing warm-starting schemes. We provide sufficient conditions for the base algorithm and heuristics to speed up learning, and verify our theory empirically in multiple simulated RL problems. More broadly, we hope that our analysis will help counter some of AI's risks by making RL more controllable via incorporating prior information into the learning process.

We remark nonetheless that the effectiveness of HuRL depends on the available heuristic. While HuRL is robust to non-ideal heuristics in the sense that it eventually solves the original RL problem, using a bad heuristic can slow down the learning compared with not using one. Therefore, an important future research direction is to adaptively tune the mixing coefficient based on the heuristic quality with curriculum or meta learning techniques. In addition, while our theoretical analysis points out a strong connection between good heuristics for HuRL and pessimistic offline RL, we found that the existing techniques for the latter are not yet scalable and robust enough for high-dimensional problems. Further research on offline RL is needed to unlock the full potential of HuRL.

[3] We choose (2) over the PBRS version, because the latter does not preserve the properties in Proposition 4.2.

References

[1] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[2] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[3] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *NeurIPS*, pages 2818–2826, 2015.

[4] Aaron Sidford, Mengdi Wang, Xian Wu, Lin F Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *NeurIPS*, pages 5192–5202, 2018.

[5] Chen Tessler and Shie Mannor. Maximizing the total reward via reward tweaking. *arXiv preprint arXiv:2002.03327*, 2020.

[6] Harm Van Seijen, Mehdi Fatemi, and Arash Tavakoli. Using a logarithmic mapping to enable lower discount factors in reinforcement learning. In *NeurIPS*, volume 32. Curran Associates, Inc., 2019.

[7] Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending mpc & value function approximation for efficient reinforcement learning. *arXiv preprint arXiv:2012.05909*, 2020.

[8] Amir-massoud Farahmand, Daniel N Nikovski, Yuji Igarashi, and Hiroki Konaka. Truncated approximate dynamic programming with task-dependent terminal value. In *AAAI*, pages 3123–3129, 2016.

[9] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[10] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018.

[11] Dylan Foster and Alexander Rakhlin. Beyond UCB: Optimal and efficient contextual bandits with regression oracles. In *ICML*, pages 3199–3210, 2020.

[12] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[13] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[14] Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *ADPRL*, pages 100–107. IEEE, 2013.

[15] David Hoeller, Farbod Farshidian, and Marco Hutter. Deep value model predictive control. In *CoRL*, pages 990–1004. PMLR, 2020.

[16] Wissam Bejjani, Rafael Papallas, Matteo Leonetti, and Mehmet R Dogar. Planning with a receding horizon for manipulation in clutter using a learned value function. In *Humanoids*, pages 1–9. IEEE, 2018.

[17] David Blackwell. Discrete dynamic programming. *The Annals of Mathematical Statistics*, pages 719–726, 1962.

[18] Marek Petrik and Bruno Scherrer. Biasing approximate dynamic programming with a lower discount factor. In *NeurIPS*, pages 1265–1272, 2008.

[19] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *AAMAS*, pages 1181–1189, 2015.

[20] Nan Jiang, Satinder Singh, and Ambuj Tewari. On structural properties of mdps that bound loss due to shallow planning. In *IJCAI*, pages 1640–1647, 2016.

[21] Yi-Chun Chen, Mykel J Kochenderfer, and Matthijs TJ Spaan. Improving offline value-function approximations for pomdps by reducing discount factors. In *IROS*, pages 3531–3536. IEEE, 2018.

[22] Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. In *ICML*, pages 269–278. PMLR, 2020.

[23] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? *arXiv preprint arXiv:2012.15085*, 2020.

[24] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch off-policy reinforcement learning without great exploration. *NeurIPS*, 33, 2020.

[25] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

[26] Sharada Mohanty, Jyotish Poonganam, Adrien Gaidon, Andrey Kolobov, Blake Wulfe, Dipam Chakraborty, Gražvydas Šemetulskis, João Schapke, Jonas Kubilius, Jurgis Pašukonis, et al. Measuring sample efficiency and generalization in reinforcement learning benchmarks: Neurips 2020 procgen benchmark. *arXiv preprint arXiv:2103.15332*, 2021.

[27] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[28] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *arXiv preprint arXiv:1807.03765*, 2018.

[29] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pages 1861–1870. PMLR, 2018.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] The garage contributors. Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage, 2019.

[32] Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *NeurIPS*, 33, 2020.

[33] Harm Seijen and Rich Sutton. True online td (lambda). In *International Conference on Machine Learning*, pages 692–700. PMLR, 2014.

[34] Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. Beyond the one-step greedy approach in reinforcement learning. In *ICML*, pages 1387–1396. PMLR, 2018.

[35] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[36] Craig Sherstan, Shibhansh Dohare, James MacGlashan, Johannes Günther, and Patrick M Pilarski. Gamma-nets: Generalizing value estimation over timescale. In *AAAI*, pages 5717–5725, 2020.

[37] Joshua Romoff, Peter Henderson, Ahmed Touati, Emma Brunskill, Joelle Pineau, and Yann Ollivier. Separating value functions across time-scales. In *ICML*, pages 5468–5477. PMLR, 2019.

[38] Jacques Richalet, André Rault, JL Testud, and J Papon. Model predictive heuristic control. *Automatica (journal of IFAC)*, 14(5):413–428, 1978.

[39] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[40] Caglar Gulcehre, Sergio Gómez Colmenarejo, Ziyu Wang, Jakub Sygnowski, Thomas Paine, Konrad Zolna, Yutian Chen, Matthew Hoffman, Razvan Pascanu, and Nando de Freitas. Regularized behavior value estimation. *arXiv preprint arXiv:2103.09575*, 2021.

[41] Tyler Lu, Dale Schuurmans, and Craig Boutilier. Non-delusional q-learning and value iteration. In *NeurIPS*, pages 9971–9981, 2018.

[42] Reinaldo AC Bianchi, Murilo F Martins, Carlos HC Ribeiro, and Anna HR Costa. Heuristically-accelerated multiagent reinforcement learning. *IEEE transactions on cybernetics*, 44(2):252–265, 2013.

[43] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International Conference on Machine Learning*, pages 3309–3318. PMLR, 2017.

[44] Ching-An Cheng, Andrey Kolobov, and Alekh Agarwal. Policy improvement via imitation of multiple oracles. *NeurIPS*, 33, 2020.

[45] Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. In *International Conference on Learning Representations*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The assumptions are in the theorem, proposition, and lemma statements throughout the paper.
   (b) Did you include complete proofs of all theoretical results? [Yes] Appendix A and Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Section 5 and Appendix C.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] References [31], [25], and [26] in Section 5.1.
   (b) Did you mention the license of the assets? [Yes] In Section 5.1.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Heuristic computation and scripts to run training.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]