Exploration-Guided Reward Shaping for Reinforcement Learning under Sparse Rewards

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the problem of reward shaping to accelerate the training process of a reinforcement learning agent. Existing works have considered a number of different reward shaping formulations; however, they either require external domain knowledge or fail in environments with extremely sparse rewards. In this paper, our goal is to develop a reward shaping framework that can operate in a fully self-supervised manner and can accelerate an agent's learning even in sparse-reward environments. We propose a novel framework, Exploration-Guided Reward Shaping, that combines exploration-based intrinsic bonuses along with extrinsic reward signals. We theoretically showcase the utility of our reward shaping framework in contrast to existing approaches. Experimental results on several environments with sparse/noisy reward signals demonstrate the effectiveness of our framework.

1 Introduction

Training reinforcement learning (RL) agents in environments with extremely sparse or distracting rewards is challenging. Existing works have studied several approaches to design informative rewards that speed up the agent's convergence [1–7]. One well-studied line of work is potential-based reward shaping, where the potential function is specified by an expert or obtained via transfer learning techniques (see [3, 8–17]). Another popular approach is to learn rewards via Inverse-RL using expert demonstrations [18]. Alternatively, one could also consider a manual specification of rewards, e.g., using distance-based metrics [19]. However, these reward design techniques typically rely on high-quality domain knowledge and may fail in practice. In fact, the RL agents can easily exploit poorly designed rewards and get stuck in local optima. This naturally leads to the fundamental question of how to do online reward shaping without relying on expert domain knowledge. More concretely, *can we design informative rewards that will accelerate the agent's training process by leveraging experience gained online during the agent's training lifetime itself?* [20–24]

To tackle this question, recent works [24–26] have explored fully self-supervised learning of parametric intrinsic rewards that can improve the performance of RL agents. In particular, these methods alternate between intrinsic reward parameter learning and the agent's policy optimization w.r.t. the learned reward. For instance, the Learning Intrinsic Rewards for Policy Gradient (LIPRG) technique [25] updates the intrinsic reward parameters to maximize the extrinsic rewards received by the policy from the environment. Self-supervised Online Reward Shaping (SORS) technique [26] infers an intrinsic reward using a classification-based reward inference algorithm, TREX [27]. However, these fully self-supervised reward shaping techniques might fail to produce meaningful agent behavior in environments with extremely sparse rewards (called *hard-exploration* domains) as they lack an explicit explorative component. Intuitively, these techniques will not be able to make updates to parameters of their intrinsic reward functions, without receiving a non-zero extrinsic reward signal.

In a parallel line of work, several techniques have been proposed to specifically tackle the challenges of extreme sparsity and exploration. One such line of work is to add more stochasticity in the agent's

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

behavior (e.g., [28–30]); however such techniques typically succeed in tasks with already well-shaped rewards. Another important line of work, relevant to our proposed framework, is bonus-driven exploration techniques for tackling hard-exploration domains – these techniques typically augment the extrinsic rewards with additional intrinsic bonus signals to encourage extra exploration [31]. The idea behind count-based bonuses is that it encourages RL agents to experience infrequently visited states [32, 32–34]. For instance, [34] proposed a simple generalization of the classic exploration method with count-based intrinsic bonuses [35–38]; in particular, they discretize the state space with a hash function. Another category of intrinsic bonuses is providing rewards for improving the agent's knowledge about the environment [39–44]. However, simply relying on these bonus-driven signals can mislead the agent towards sub-optimal or bad behaviors — for instance, in *noisy-distractive* domains such as the "noisy TV problem" [45], unpredictable random noise outputs would attract the agent's attention forever.

An important research question that we seek to address is: *How can we design an online intrinsic reward function, without any domain knowledge or external supervision, that can speed up the agent's learning process even in environments with extreme sparsity and noisy distractions?* To this end, we propose a novel framework, Exploration-Guided Reward Shaping, that combines both exploration-focused and exploitation-focused intrinsic reward components. Our framework of online reward shaping alternates between reward learning and policy optimization. Similar to SORS [26], our method is compatible with any existing RL algorithm, and not only policy-gradient style learners as considered in [25]. Our main contributions are:

I. We propose a novel framework, Exploration-Guided Reward Shaping (ExpRS), that appropriately balances exploration (via an intrinsic bonus component) and exploitation (via an intrinsic reward component) of extrinsic signals (Sections 2 and 3.1).

II. We derive intuitive meta-gradients for updating the intrinsic reward component that enables our method to be broadly applicable to any RL agent. We theoretically showcase the utility of ExpRS in a chain environment (Sections 3.2 and 3.3).

III. We empirically demonstrate the effectiveness of ExpRS on several environments with sparse and noisy reward signals (Section 4).

2 Problem Setup

2.1 Preliminaries

An environment is defined as a Markov Decision Process (MDP) $M := (\mathcal{S}, \mathcal{A}, T, P_0, \gamma, R)$, where the state and action spaces are denoted by $\mathcal{S}$ and $\mathcal{A}$ respectively. $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ captures the state transition dynamics, i.e., $T(s' \mid s, a)$ denotes the probability of landing in state s' by taking action a from state s. γ is the discounting factor, and P_0 is the initial state distribution. The reward function is given by $R : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$, for some $R_{\max} > 0$. We denote the true underlying extrinsic reward function by $\overline{R}$ and the designed reward function by $\widehat{R}$.

We denote a stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ as a mapping from a state to a probability distribution over actions, and a deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ as a mapping from a state to an action. For any trajectory $\xi = \{(s_t, a_t)\}_{t=0,1,\ldots,H}$, we define its cumulative return w.r.t. reward function R as $J(\xi, R) := \sum_{t=0}^{H} \gamma^t \cdot R(s_t, a_t)$. Then, the expected cumulative return (value) of a policy π w.r.t. R is defined as $J(\pi, R) := \mathbb{E}\left[J(\xi, R) | P_0, T, \pi\right]$, where $s_0 \sim P_0(\cdot)$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim T(\cdot|s_t, a_t)$. The learner seeks to find a policy that has maximum value w.r.t. the extrinsic reward function $\overline{R}$, i.e., $\max_\pi J(\pi, \overline{R})$.

2.2 Online Reward Shaping

A general framework of online reward shaping for RL agents is given in Algorithm 1. A natural objective here is to design informative rewards $\widehat{R}_k$ at each round k so that the resulting final policy π_K performs better (i.e., has high value w.r.t. $\overline{R}$) compared to the corresponding policy obtained via the standard training with $\widehat{R}_k = \overline{R}$. Note that, we consider a single lifetime training setting for an RL agent, on a single task, i.e., between rounds we do not *reset* the policy.

Algorithm 1 Online Reward Shaping

1: **Input:** Extrinsic reward $\overline{R}$, and RL algorithm L
2: **Initialization:** π_0, $\widehat{R}_0$
3: **for** $k = 1, 2, \ldots, K$ **do**
4: update policy $\pi_k \leftarrow L(\pi_{k-1}, \widehat{R}_{k-1})$
5: update reward $\widehat{R}_k$ using $\widehat{R}_{k-1}$ and π_k
6: **Output:** π_K

2.3 Existing Techniques and Issues

Reward shaping based on expert domain knowledge. A popular technique for reward shaping is potential-based reward shaping (PBRS) which guarantees that any optimal policy induced by the designed function is also optimal under the extrinsic reward function [3]. Given a potential function $\Phi : \mathcal{S} \rightarrow \mathbb{R}$ that quantifies "goodness" of the states, the reward function $\widehat{R}_k$ produced by the PBRS framework is given by:

$$\widehat{R}_k\left(s, a\right) \;:=\; \overline{R}\left(s, a\right) + \gamma \sum_{s' \in \mathcal{S}} T\left(s' \mid s, a\right) \cdot \Phi\left(s'\right) - \Phi\left(s\right).$$

For the above reward function to be helpful for training the RL agent, we need to have access to good potential functions based on expert domain knowledge. It has already been noted that using the wrong potential functions could drastically slow down the learning process [46]. The authors in [12, 47] have extended the PBRS framework to incorporate dynamic potential functions that are learned online, alongside agent's training. As noted in [26], the effects of dynamic PBRS can be "learned away" over time, as they are equivalent to value function initialization [48].

Reward shaping based on exploration bonuses. In the bonus-driven exploration framework [32–34], a count-based intrinsic bonus $B_k(s)$ is given to the agent to encourage exploration. The bonus $B_k(s)$ measures the "novelty" of a state s given the history of all transitions up to round k. The authors in [34] extend the classic exploration methods with count-based intrinsic bonuses [35–38] to high-dimensional, continuous state spaces. The states are mapped to hash codes, which allows to count their occurrences with a hash table. These counts are then used to compute an intrinsic bonus according to the classic bonus-driven exploration theory. At round k, the agent receives the reward $\widehat{R}_k(s, a) = \overline{R}(s, a) + B_k(s)$. These "exploration-only" reward shaping methods do not appropriately combine the successful extrinsic reward signals received from the environment. When there are distractive zones in the state space, these methods will keep on exploring the space even after obtaining extrinsic reward signals.

Reward shaping in fully self-supervised way. The self-supervised reward shaping methods [24–26] learn the parameters ϕ of an intrinsic reward function R_ϕ by exploiting the successful extrinsic reward signals received from the environment. These parameters are learned online to speed up the agent's learning during its lifetime. At round k, the LIPRG [25] technique provides the reward $\widehat{R}_k(s, a) = \overline{R}(s, a) + R_{\phi_k}(s, a)$ and the SORS [26] technique provides the reward $\widehat{R}_k(s, a) = R_{\phi_k}(s, a)$ to the agent. These self-supervised reward shaping methods lack an explicit explorative component. When $\overline{R}$ is extremely sparse, the learner would not get any extrinsic signal to update the parameters of R_ϕ, until a first successful rollout is realized from π_k.

In this paper, we are interested in developing an online reward shaping technique that can succeed in environments with extremely sparse and distractive rewards, without any expert guidance.

3 Exploration-Guided Reward Shaping

In Sections 3.1 and 3.2, we propose an exploration-guided reward shaping technique, ExpRS, to accelerate an RL agent's training process. In Section 3.3, we theoretically compare our reward shaping technique against baseline techniques in a chain environment.

3.1 Our Reward Formulation

We design the following parametric reward function for Algorithm 1:

$$\widehat{R}^{\text{ExpRS}}(s, a) \;:=\; \overline{R}(s, a) + R_\phi^{\text{SelfRS}}(s, a) + B_w^{\text{Exp}}(s),$$

where $\phi \in \mathbb{R}^{d_\phi}$ and $w \in \mathbb{R}^{d_w}$. Here, R_ϕ^{SELFRS} corresponds to the intrinsic rewards in self-supervised reward shaping techniques, and B_w^{EXP} corresponds to the intrinsic bonuses in exploration-only reward shaping techniques. For the remainder of the section, we drop the superscripts in the reward/bonus terms. At round k of Algorithm 1, $\widehat{R}_{k-1}(s,a)$ is designed with parameters (ϕ_{k-1}, w_{k-1}). Then, given updated policy π_k, we update the parameters (ϕ_{k-1}, w_{k-1}) to new values (ϕ_k, w_k).

Intrinsic reward R_ϕ. We model the intrinsic reward R_ϕ using any general parameterized function. At round k, for fixed π_k and w_{k-1}, we update the parameter ϕ_{k-1} to ϕ_k by considering the effect such a change would have on the the expected cumulative return (w.r.t. $\overline{R}$) of the learner through the change in the policy π_k (as in LIPRG [25]). In particular, we update ϕ using the gradient $\left[\nabla_\phi J(L(\pi_k, \widehat{R}), \overline{R})\right]_{\phi_{k-1}}$, where $\widehat{R}(s,a) = \overline{R}(s,a) + R_\phi(s,a) + B_{w_{k-1}}(s)$. However, we may not have access to the learning algorithm L directly or it might be too complex to analyze the impact of the change in ϕ considering L. Since our end goal is to speed up the learning process of any RL agent, it is actually useful to consider a simple learning algorithm with parametric policies $\left\{\pi_\theta : \theta \in \mathbb{R}^{d_\theta}\right\}$. To this end, we consider a learning algorithm $\widetilde{L}$ that does 1-step vanilla policy gradient update with h-depth planning for Q-values. In particular, we map the policy π_k to a parameter $\theta_k \in \mathbb{R}^{d_\theta}$ and define:

$$\widetilde{L}(\theta_k, \widehat{R}) \;:=\; \theta_k + \alpha \cdot \left[\nabla_\theta J(\pi_\theta, \widehat{R})\right]_{\theta_k} \;=\; \theta_k + \alpha \cdot \mathbb{E}_{\mu^{\pi_{\theta_k}}(s,a)}\left[\left[\nabla_\theta \log \pi_\theta(a|s)\right]_{\theta_k} \cdot Q^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right],$$

where $Q^{\pi_{\theta_k}}_{\widehat{R},h}(s,a) \;=\; \mathbb{E}\left[\sum_{t=0}^{h} \gamma^t \cdot \widehat{R}(s_t,a_t) | s_0 = s, a_0 = a, T, \pi_{\theta_k}\right]$ and α is the learning rate. Then, we update the parameter ϕ using the following bi-level optimization:

$$\arg\max_{\phi} \quad J(\pi_{\theta(\phi)}, \overline{R}) \tag{P1.U}$$

$$\text{subject to} \quad \theta(\phi) \leftarrow \widetilde{L}(\theta_k, \widehat{R}) \tag{P1.L}$$

$$\text{where } \widehat{R}(s,a) := \overline{R}(s,a) + R_\phi(s,a) + B_{w_{k-1}}(s).$$

Note that the above bi-level formulation explicitly accounts for the requirement of accelerating the training process of an RL agent, with h-depth planning. Further, we note that the LIPRG [25] technique is specifically designed for policy-gradient style RL agents, and their technique is incompatible with value-based RL agents.

Intrinsic bonus B_w. Given a state abstraction $\psi : \mathcal{S} \to \mathcal{X}_\psi$ (with $|\mathcal{X}_\psi| = d_w$), we maintain the visitation count of the abstracted states in w, i.e., $w[x]$ corresponds to the visitation counts of the states $\{s \in \mathcal{S} : \psi(s) = x\}$. This allows us to implicitly maintain pseudo-counts $N_w(s)$ of visiting states $s \in \mathcal{S}$. In particular, we set $N_w(s) = \left(\frac{\lambda}{\overline{R}_{\max}}\right)^2 + w[\psi(s)]$ for some $\lambda > 0$. Then, we define the intrinsic bonus as follows: $B_w(s) = \frac{\lambda}{\sqrt{N_w(s)}}$. We update w based on the rollouts in round k [32–34].

3.2 Proposed Algorithm

Parameter updates for R_ϕ. We solve the bi-level optimization problem (P1.U)-(P1.L) of the intrinsic reward component in iterative manner using the gradient updates that we derive below. At round k, for fixed π_k and w_{k-1}, we update the parameter ϕ_{k-1} to ϕ_k as follows:

$$\begin{aligned}
\phi_k \;&=\; \phi_{k-1} + \eta \cdot \left[\nabla_\phi J(\pi_{\theta(\phi)}, \overline{R})\right]_{\phi_{k-1}} \\
&\overset{(a)}{=}\; \phi_{k-1} + \eta \cdot \left[\nabla_\phi \theta(\phi) \cdot \nabla_{\theta(\phi)} J(\pi_{\theta(\phi)}, \overline{R})\right]_{\phi_{k-1}} \\
&\overset{(b)}{\approx}\; \phi_{k-1} + \eta \cdot \left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}} \cdot \left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k},
\end{aligned}$$

where η is the learning rate, (a) is due to chain rule, and (b) is due to the following smoothness condition: $\left\|\left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta(\phi_{k-1})} - \left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k}\right\|_2 \le \|\theta(\phi_{k-1}) - \theta_k\|_2$. Next, we analyze the two gradient terms, $\left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}} \in \mathbb{R}^{d_\phi \times d_\theta}$ and $\left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k} \in \mathbb{R}^{d_\theta}$, separately. Below, we use the following notation: $\mu^\pi(s,a)$ means $s \sim d^\pi, a \sim \pi(\cdot|s)$; similarly, $\mu^\pi(s)$ means $s \sim d^\pi$. By using the policy gradient theorem, we have:

$$\left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k} \;=\; \mathbb{E}_{\mu^{\pi_{\theta_k}}(s,a)}\left[\left[\nabla_\theta \log \pi_\theta(a|s)\right]_{\theta_k} \cdot Q^{\pi_{\theta_k}}_{\overline{R}}(s,a)\right].$$

We obtain the meta-gradient [49–51] as follows:

$$\left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}} \;=\; \alpha \cdot \mathbb{E}_{\mu^{\pi_{\theta_k}}(s,a)}\Big[\left[\nabla_\phi Q^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \left[\nabla_\theta \log \pi_\theta(a|s)\right]^\top_{\theta_k}\Big],$$

where $\widehat{R}(s,a) := \overline{R}(s,a) + R_\phi(s,a) + B_{w_{k-1}}(s)$.

Here, we aim to derive intuitive simplifications of the above two gradient terms that enable our method to be applicable to any RL agent, not just policy-gradient style agents (in comparison to LIPRG [25]). For the derivation purposes, we further simplify the learner model using tabular representation. However, the algorithm and updates do not require this. In particular, we consider $\theta \in \mathbb{R}^{S \cdot A}$ (where $S = |\mathcal{S}|$ and $A = |\mathcal{A}|$), and define $\pi_\theta(a|s) := \frac{\exp\left(\theta(s,a)\right)}{\sum_b \exp\left(\theta(s,b)\right)}, \forall s \in \mathcal{S}, a \in \mathcal{A}$. Note that the reward function R_ϕ is still represented by a parametric function. For any $s \in \mathcal{S}, a \in \mathcal{A}$, let $\mathbf{1}_{s,a} \in \mathbb{R}^{S \cdot A}$ denote a vector that has one in the (s,a)-th entry and zero else where. We define $A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a) := Q^{\pi_{\theta_k}}_{\widehat{R},h}(s,a) - V^{\pi_{\theta_k}}_{\widehat{R},h}(s)$ and $A^{\pi_{\theta_k}}_{\overline{R}}(s,a) := Q^{\pi_{\theta_k}}_{\overline{R}}(s,a) - V^{\pi_{\theta_k}}_{\overline{R}}(s)$. Then, we write the above two gradient terms as follows (with $\mu = \mu^{\pi_{\theta_k}}(s,a)$):

$$\begin{aligned}
\mathbb{E}_\mu\Big[\left[\nabla_\phi Q^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \left[\nabla_\theta \log \pi_\theta(a|s)\right]^\top_{\theta_k}\Big] \;&=\; \mathbb{E}_\mu\Big[\left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \mathbf{1}^\top_{s,a}\Big] \\
\mathbb{E}_\mu\Big[\left[\nabla_\theta \log \pi_\theta(a|s)\right]_{\theta_k} \cdot Q^{\pi_{\theta_k}}_{\overline{R}}(s,a)\Big] \;&=\; \mathbb{E}_\mu\Big[A^{\pi_{\theta_k}}_{\overline{R}}(s,a) \cdot \mathbf{1}_{s,a}\Big].
\end{aligned}$$

Finally, we get the full update as follows (with $\eta' = \eta \cdot \alpha$, $\mu = \mu^{\pi_{\theta_k}}(s,a)$ and $\mu' = \mu^{\pi_{\theta_k}}(s',a')$):

$$\begin{aligned}
\phi_k \;&\approx\; \phi_{k-1} + \eta' \cdot \mathbb{E}_\mu\Big[\left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \mathbf{1}^\top_{s,a}\Big] \cdot \mathbb{E}_{\mu'}\Big[A^{\pi_{\theta_k}}_{\overline{R}}(s',a') \cdot \mathbf{1}_{s',a'}\Big] \\
&=\; \phi_{k-1} + \eta' \cdot \mathbb{E}_\mu\Big[\left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \mathbf{1}^\top_{s,a} \cdot \mathbb{E}_{\mu'}\Big[A^{\pi_{\theta_k}}_{\overline{R}}(s',a') \cdot \mathbf{1}_{s',a'}\Big]\Big] \\
&=\; \phi_{k-1} + \eta' \cdot \mathbb{E}_\mu\Big[\left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \mathbf{1}^\top_{s,a} \cdot \mu \cdot A^{\pi_{\theta_k}}_{\overline{R}}(s,a) \cdot \mathbf{1}_{s,a}\Big] \\
&=\; \phi_{k-1} + \eta' \cdot \mathbb{E}_\mu\Big[\mu \cdot A^{\pi_{\theta_k}}_{\overline{R}}(s,a) \cdot \left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}}\Big] \\
&=\; \phi_{k-1} + \eta' \cdot \mathbb{E}_\mu\Big[\mu^{\pi_{\theta_k}}(s) \cdot \pi_{\theta_k}(a|s) \cdot A^{\pi_{\theta_k}}_{\overline{R}}(s,a) \cdot \left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}}\Big]. \qquad (1)
\end{aligned}$$

Our algorithm with empirical updates. First, we obtain empirical update rules for intrinsic reward and bonus components. To this end, we rely on the rollout data $\mathcal{D}_k$ collected by executing the learner's current policy π_k in the MDP M. Then, based on Eq. (1), we update the parameter ϕ of the intrinsic reward, for fixed w_{k-1}, as follows:

$$\begin{aligned}
\phi_k \;&\approx\; \phi_{k-1} + \eta' \cdot \frac{1}{|\mathcal{D}_k|} \sum_{(s_i,a_i,\xi_i) \in \mathcal{D}_k} \mu^{\pi_k}(s_i) \cdot \pi_k(a_i|s_i) \cdot A^{\pi_k}_{\overline{R}}(s_i,a_i) \cdot \left[\nabla_\phi A^{\pi_k}_{\widehat{R},h}(s_i,a_i)\right]_{\phi_{k-1}} \\
&\approx\; \phi_{k-1} + \eta'' \cdot \sum_{(s_i,a_i,\xi_i) \in \mathcal{D}_k} \pi_k(a_i|s_i) \cdot A^{\pi_k}_{\overline{R}}(s_i,a_i) \cdot \left[\nabla_\phi A^{\pi_k}_{\widehat{R},h}(s_i,a_i)\right]_{\phi_{k-1}}, \qquad (2)
\end{aligned}$$

where we absorbed $\frac{1}{|\mathcal{D}_k|}$ into η'', and ignored the term $\mu^{\pi_k}(s_i)$. Note that we can compute: (i) $\pi_k(a_i|s_i)$ using the learner's policy π_k, (ii) $A^{\pi_k}_{\overline{R}}(s_i,a_i)$ using $J(\xi_i, \overline{R})$ or a critic corresponding to $\overline{R}$, and (iii) $A^{\pi_k}_{\widehat{R},h}(s_i,a_i)$ using $J(\xi_i, \widehat{R})$ or a critic corresponding to $\widehat{R}$. When we use 1-depth planning for Q-values in $\widetilde{L}$, we have: $\left[\nabla_\phi A^{\pi_k}_{\widehat{R},1}(s_i,a_i)\right]_{\phi_{k-1}} = \left[\nabla_\phi R_\phi(s_i,a_i)\right]_{\phi_{k-1}} - \left[\sum_{b \in \mathcal{A}} \pi_k(b|s_i) \cdot \nabla_\phi R_\phi(s_i,b)\right]_{\phi_{k-1}}$. Now, we update the parameter w of the intrinsic bonus, for fixed ϕ_{k-1}, as follows (component-wise):

$$w_k[x] \;=\; w_{k-1}[x] + \sum_{s_i \in \mathcal{D}_k} \mathbf{1}\left\{\psi(s_i) = x\right\}. \qquad (3)$$

Then, we have $B_{w_k} = \frac{\lambda}{\sqrt{N_{w_k}(s)}}$, where $N_{w_k}(s) = \left(\frac{\lambda}{\overline{R}_{\max}}\right)^2 + w_k[\psi(s)]$. Finally, we obtain our designed reward function at round k as follows: $\widehat{R}_k(s,a) = \overline{R}(s,a) + R_{\phi_k}(s,a) + B_{w_k}(s)$. The complete procedure of our exploration-guided reward shaping is given in Algorithm 2.

Algorithm 2 Exploration-Guided Reward Shaping (ExpRS)

Input: learning rates $\{\eta''\}$, parameter λ, and reward update frequency f
Initialization: intrinsic parameters (ϕ_0, w_0), and policy π_0
collect rollouts using π_0 in M and store them in $\mathcal{D}_0$
for $k = 1, 2, \ldots, K$ **do**
  define reward $\widehat{R}_{k-1}(s,a) := \overline{R}(s,a) + R_{\phi_{k-1}}(s,a) + B_{w_{k-1}}(s)$
  update policy $\pi_k \leftarrow L(\pi_{k-1}, \widehat{R}_{k-1})$ using $\mathcal{D}_{k-1}$
  collect rollouts using π_k in M and store them in $\mathcal{D}_k$
  if $k\%f = 0$ **then**
    update ϕ_k according to Eq. (2) using $\mathcal{D}_k$
    update w_k according to Eq. (3) using $\mathcal{D}_k$
  else
    $\phi_k \leftarrow \phi_{k-1}$
    $w_k \leftarrow w_{k-1}$
Output: policy π_K

3.3 Theoretical Analysis

In this section, we theoretically showcase the utility of our exploration-guided reward shaping over fully self-supervised and exploration-only shaping techniques. In particular, we show that our shaping technique helps to learn an optimal policy in a sample efficient manner in an environment with both extremely-sparse and distractive rewards.

Environment. We consider a chain environment $M = \left(\mathcal{S}, \mathcal{A}, T, P_0, \gamma, \overline{R}\right)$ of length $n_1 + n_2 + 1$. Let the state space be $\mathcal{S} = \left\{x_{-n_2}, x_{-(n_2-1)}, \ldots, x_{-1}, x_0, x_1, \ldots, x_{(n_1-1)}, x_{n_1}\right\}$, and the action space be $\mathcal{A} = \{\leftarrow, \rightarrow\}$. We always start (or reset) in the state x_0, i.e., the initial state distribution is $P_0(x_0) = 1$. The transition dynamics is deterministic and given by: $T(x_{i+1}|x_i, \rightarrow) = 1$ for $-n_2 \leq i \leq n_1 - 1$, $T(x_{i-1}|x_i, \leftarrow) = 1$ for $-(n_2 - 1) \leq i \leq n_1$, $T(\texttt{terminal}|x_{n_1}, \rightarrow) = 1$, and $T(\texttt{terminal}|x_{-n_2}, \leftarrow) = 1$. The reward function is defined as follows: $\overline{R}(x_i, \rightarrow) = 0$ for $-n_2 \leq i \leq n_1 - 1$, $\overline{R}(x_{n_1}, \rightarrow) = 1$, and $\overline{R}(x_i, \leftarrow) = 0$ for $-n_2 \leq i \leq n_1$. We consider an infinite horizon setting with discounted returns ($H \rightarrow \infty$, and $\gamma < 1$).

Learning algorithm. For our theoretical analysis of different reward shaping techniques, we consider a simplified Q-learning style RL algorithm L given in Algorithm 3. The algorithm L takes two Boolean flags, `no-self` and `no-exp`, as input that correspond to the intrinsic reward and bonus components, R and B, respectively. For example, $L(\texttt{true}, \texttt{true})$ corresponds to the setting of learning with extrinsic reward $\overline{R}$ only.

The following theorem compares the sample complexity of different reward shaping techniques to learn an optimal policy in the chain environment.

Theorem 1. *Consider the chain environment M defined above, and the learning algorithm L given in Algorithm 3. Let cost$(L(\texttt{no-self}, \texttt{no-exp}))$ denote the number time steps required for $L(\texttt{no-self}, \texttt{no-exp})$ to learn an optimal policy in M. Then, we have the following:*

1. *without any reward shaping, $\mathbb{E}\left[\textit{cost}(L(\texttt{true}, \texttt{true}))\right] \geq \sum_{i=1}^{n_1} 2^{n_1 - i}$*
2. *with exploration-only reward shaping, cost$(L(\texttt{true}, \texttt{false})) = n_1 \cdot (n_1 + n_2 + 2)$*
3. *with fully self-supervised reward shaping, $\mathbb{E}\left[\textit{cost}(L(\texttt{false}, \texttt{true}))\right] \geq 2^{n_1 - 1}$*
4. *with exploration-guided reward shaping, cost$(L(\texttt{false}, \texttt{false})) \leq n_1 + n_2 + 2$*

Algorithm 3 Simplified RL algorithm L for theoretical analysis

Input: Boolean flags `no-self` and `no-exp`
Initialize: $V_0(s) = 0$; $R(s,a) = 0$, $B(s) = 1$, $\forall s \in \mathcal{S}, a \in \mathcal{A}$; $\lambda \in (0,1)$
$s_1 = x_0$; $B(s_1) = \lambda$
for each $t = 1, 2, \ldots$ **do**
  if `no-exp` $=$ `true` **then**
    $B(s) = 0, \forall s \in \mathcal{S}$
  $a_t = \arg\max_{a'} \overline{R}(s_t, a') + R(s_t, a') + B(T(s_t, a')) + \gamma \cdot V_{t-1}(T(s_t, a'))$
  $s_{t+1} = T(s_t, a_t)$
  $V_t(s_t) = \overline{R}(s_t, a_t) + R(s_t, a_t) + \gamma \cdot V_{t-1}(s_{t+1})$
  if $s_{t+1} =$ `terminal` **then**
    if $\overline{R}(s_t, a_t) = 1$ and `no-self` $=$ `false` **then**
      $\phi(s) = 0, \forall s \in \mathcal{S}$
      for all the states in the current rollout update $\phi(\cdot)$ as the discounted return
      $R(s,a) = \gamma \cdot \phi(T(s,a)) - \phi(s), \forall s \in \mathcal{S}, a \in \mathcal{A}$
      $V_t(s) = 0, \forall s \in \mathcal{S}$
    reset $s_{t+1} = x_0$
  $B(s_{t+1}) = \lambda \cdot B(s_{t+1})$
Output: policy π_t

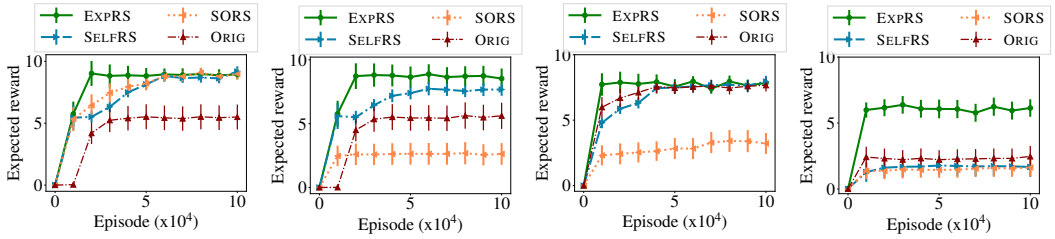
4 Experimental Evaluation

In this section, we evaluate our reward shaping framework on three environments: CHAIN (Section 4.1), ROOM (Section 4.2), and LINEK (Section 4.3). CHAIN corresponds to a navigation task in a chain, adapted from the environment used for theoretical analysis in Section 3.3; this is a canonical environment used for studying extremely sparse-reward settings [7]. ROOM corresponds to a navigation task in a grid-world where the agent has to learn a policy to quickly reach the goal location in one of four rooms, starting from an initial location. Even though this environment has a small state/action space, it provides a very rich and intuitive problem setting to validate different reward shaping techniques. In fact, variants of ROOM have been used extensively in the literature [10, 11, 14, 17, 52–55]—the environment used in our experiments is adapted from [55]. LINEK corresponds to a navigation task in a one-dimensional space where the agent has to first pick the key and then reach the goal. The agent's location is represented as a point on a line segment. This environment is inspired by variants of navigation tasks in the literature where an agent needs to perform subtasks [3, 55, 56]—the environment used in our experiments is adapted from [55]. We give an overview of main results here, and provide a more detailed description of the setup and additional results in the supplementary material.

4.1 Evaluation on CHAIN

CHAIN (Figure 1). We represent the chain environment of length $n_1 + n_2 + 1$ as an MDP with state-space $\mathcal{S}$ consisting of an initial location x_0 (shown as "blue-circle"), n_1 nodes to the right of x_0, and n_2 nodes to the left of x_0. The rightmost node of the chain is the "goal" state (shown as "green-star"). In the left part of the chain, there can be a "distractor" state (shown as "green-plus"). The agent can take two actions given by $\mathcal{A} := \{\text{"left"}, \text{"right"}\}$. An action takes the agent to the neighboring node represented by the direction of the action. However, taking "left" action at the leftmost node (shown as "thick-red-circle") leads to termination, and "right" action at the rightmost node (goal) keeps the agent at the current location. Furthermore, when an agent takes an action $a \in \mathcal{A}$, there is p_{rand} probability that an action $a' \in \mathcal{A} \setminus \{a\}$ will be executed instead of a. The agent receives a reward of: $R_{\max}$ for the "right" action at the goal state, R_{dis} for the "left" action at the distractor state, and 0 for all other state-action pairs. There is a discount factor γ and the environment resets after a horizon of H. In our evaluation, we set $p_{\text{rand}} = 0.05$, $R_{\max} = 1$, $R_{\text{dis}} = 0$ or 0.01, $H = 30$, and $\gamma = 0.99$. We obtain different variants of the chain environment by changing the values of $(n_1, n_2, R_{\text{dis}})$. In particular, CHAIN^0 denotes a set of two environments with the following configurations: $\{(15, 20, 0), (20, 20, 0)\}$. Similarly, CHAIN$^+$ denotes a set of two environments with the following configurations: $\{(15, 20, 0.01), (20, 20, 0.01)\}$.

Figure 1: CHAIN^0 / CHAIN$^+$



(a) Chain^0, REINFORCE (b) Chain$^+$, REINFORCE (c) Chain^0, Q-learning (d) Chain$^+$, Q-learning

Figure 2: Results for Chain environment. These plots show convergence in performance of the agent w.r.t. training episodes. **(a, b)** show results for REINFORCE agent on Chain^0 (i.e., Chain variant without any distractor state) and Chain$^+$ (i.e., Chain variant with a distractor state). **(c, d)** show results for Q-learning agent on Chain^0 and Chain$^+$. See Section 4.1 for details.

Evaluation setup. We conduct our experiments with two different types of RL agents [7]: First, we consider tabular REINFORCE agent that maintain scores $\theta[s, a]$ for each state-action pair and applies soft-max operation over the scores to obtain the policy; for the agent's updates, we set learning rate $\alpha = 0.1$. Second, we consider Q-learning agent with learning rate $\alpha = 0.1$, and exploration factor $\epsilon = 0.05$. We compare the performance of the following reward shaping techniques: (i) $\widehat{R}^{\text{Orig}} := \overline{R}$, which represents the extrinsic reward function; (ii) $\widehat{R}^{\text{SelfRS}} := \overline{R} + R_\phi^{\text{SelfRS}}$, where R_ϕ^{SelfRS} is learned with a tabular parameterization ϕ along with a critic w.r.t. $\overline{R}$ (see Section 3.1); (iii) $\widehat{R}^{\text{ExpRS}} := \overline{R} + R_\phi^{\text{SelfRS}} + B_w^{\text{Exp}}$, where B_w^{Exp} is a count-based bonus (see Section 3.1); (iv) $\widehat{R}^{\text{SORS}}$ is obtained via the SORS technique [26]. Next, we discuss details of the full training process. We maintain a rollout buffer (first-in-first-out) $\mathcal{D}$ of size 10. For every 5 rounds (corresponding to 5 rollouts in the MDP), we update the intrinsic reward using all rollouts in the buffer $\mathcal{D}$. For every 2 rounds, we update the agent's policy using the latest 5 rollouts in $\mathcal{D}$. We update the intrinsic bonuses at every environment step during a rollout. Note that, for stability, we update the policy more frequently than the intrinsic reward and at a higher learning rate, as considered in the work of [25, 26]. In the supplementary material, we have provided full details of technique-specific hyperparameters and learning rates.

Results. During training, the agent receives rewards based on $\widehat{R}$ and is evaluated based on $\overline{R}$. Figure 2 illustrates results for both variants of Chain environment; the reported results are averaged over 30 runs and convergence plots show the mean performance with standard error bars. The convergence behavior in Figure 2 demonstrates the effectiveness of our exploration-guided reward shaping framework. In particular, the RL agents (both REINFORCE and Q-learning) trained with $\widehat{R}^{\text{ExpRS}}$ outperforms all other techniques in both Chain^0 and Chain$^+$ environments. Specifically, we attribute the following reasons for the poor performance of SORS technique: (i) it is not specifically designed for environments with extremely-sparse rewards; (ii) it is agnostic to the scale of the extrinsic reward function as it only uses the information about pairwise labels.

4.2 Evaluation on Room

Room (Figure 3). The environment used in our experiments is based on the work of [55]; however, we adapted it to have a "distractor" state (shown as "green-plus") that provides a small reward. Similar to the two variants of Chain, we have two variants of this environment: (a) Room^0 has $R_{\text{dis}} = 0$ at the distractor state shown as "green-plus" (equivalently, there is no distractor state); (b) Room$^+$ has $R_{\text{dis}} = 0.01$ at the distractor state. The environment-specific parameters (including p_{rand}, $R_{\max}$, H, γ) are kept same as in Section 4.1. We defer the full details of the environment to the supplementary material.

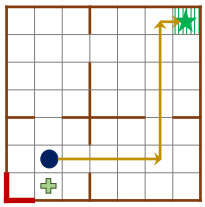
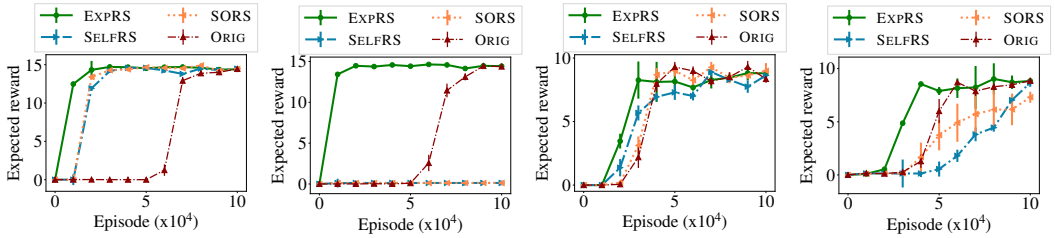


Figure 3: Room^0 / Room$^+$

Evaluation setup and results. Our evaluation setup for this environment is the same as the Chain environment (described in Section 4.1); here, we consider only the tabular REINFORCE agent. In particular, all the hyperparameters (related to the REINFORCE agent, reward shaping techniques, and training process) are the same as in Section 4.1. Figures 4a and 4b show the agent's performance for environments Room^0 and Room$^+$ (averaged over 20 runs). These results, along with results



(a) Room^0, REINFORCE (b) Room$^+$, REINFORCE (c) LineK^0, REINFORCE (d) LineK$^+$, REINFORCE

Figure 4: Results for Room and LineK environments. These plots show convergence in performance of the agent w.r.t. training episodes. **(a, b)** show results for REINFORCE agent on Room^0 (i.e., Room variant without any distractor state) and Room$^+$ (i.e., Room variant with a distractor state). **(c, d)** show results for REINFORCE agent on LineK^0 (i.e., LineK variant without any distractor state) and LineK$^+$ (i.e., LineK variant with a distractor state). See Sections 4.2 and 4.3 for details.

obtained in Figure 2, further demonstrate the effectiveness and robustness of $\widehat{R}^{\text{ExpRS}}$ across different environments, compared to baselines.

4.3 Evaluation on LineK

LineK (Figure 5). This environment corresponds to a navigation task in a one-dimensional space where the agent has to first pick the key and then reach the goal. The environment used in our experiments is adapted from the work of [55]. In particular, similar to Sections 4.1 and 4.2, we use two adaptations of the environment: (a) LineK^0 has $R_{\text{dis}} = 0$ and has no distractor state; (b) Room$^+$ has $R_{\text{dis}} = 0.01$ at the distractor state, i.e., the agent can go to the goal area ("green region" marked with "vertical lines") without possessing the key and achieve a small reward of R_{dis}. We represent the environment as an MDP with $\mathcal{S}$ states corresponding to the agent's status comprising of the current location (shown as "blue-circle") and a binary flag indicating whether the agent has acquired a key (shown as "cyan-bolt"); the agent can take three actions given by $\mathcal{A} := \{$"left", "right", , "pick"$\}$. Based on the setting considered in [55], we use an abstracted state space in our experiments obtained via discretization of the segment which is then passed in bitmap vector representation to neural networks. We defer full details of the environment to the supplementary material.

Figure 5: LineK^0 / LineK$^+$

Experimental setup and results. We conduct our experiments with a neural REINFORCE agent using a two-layered neural network architecture. As before, we compare the performance of four reward shaping techniques. As a crucial difference to experiments in Sections 4.1 and 4.2, here we use neural-network based reward functions for $\widehat{R}^{\text{SelfRS}}$, $\widehat{R}^{\text{ExpRS}}$, and $\widehat{R}^{\text{SORS}}$. These networks use a single non-linear layer that applies $tanh$-clipping for the output reward values (as considered by [25, 26].) Figures 4c and 4d show the agent's performance for environments Room^0 and Room$^+$ (averaged over 5 runs). These results showcase the effectiveness of $\widehat{R}^{\text{ExpRS}}$ in more complex learning settings, requiring neural representations for reward functions. We provide additional results in the supplementary material.

5 Concluding Discussions

We proposed a novel reward shaping framework that operates in a fully self-supervised manner and could accelerate an agent's learning even in sparse-reward environments. Our framework is based on combining exploration-based intrinsic bonuses along with extrinsic reward signals to accelerate the agent's training process. Experimental evaluation in different environments demonstrated the effectiveness of our reward shaping framework. Next, we discuss a few limitations of our work and outline a future plan on how our work can be extended to address them. First, it would be interesting to extend the experimental evaluation to more complex environments (e.g., with continuous state/action spaces). Second, it would be useful to provide rigorous analysis of our reward shaping framework in terms of convergence speed and stability of an agent. Third, it would be useful to systematically study our reward design framework across agents trained using different methods.

References

[1] Maja J. Mataric. Reward Functions for Accelerated Learning. In *ICML*, pages 181–189, 1994.

[2] Jette Randløv and Preben Alstrøm. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *ICML*, pages 463–471, 1998.

[3] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *ICML*, pages 278–287, 1999.

[4] Adam Laud and Gerald DeJong. The Influence of Reward on the Speed of Reinforcement Learning: An Analysis of Shaping. In *ICML*, pages 440–447, 2003.

[5] Falcon Z. Dai and Matthew R. Walter. Maximum Expected Hitting Cost of a Markov Decision Process and Informativeness of Rewards. In *NeurIPS*, pages 7677–7685, 2019.

[6] Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return Decomposition for Delayed Rewards. In *NeurIPS*, pages 13544–13555, 2019.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[8] Eric Wiewiora. Potential-Based Shaping and Q-Value Initialization are Equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

[9] Eric Wiewiora, Garrison W. Cottrell, and Charles Elkan. Principled Methods for Advising Reinforcement Learning Agents. In *ICML*, pages 792–799, 2003.

[10] John Asmuth, Michael L. Littman, and Robert Zinkov. Potential-based Shaping in Model-based Reinforcement Learning. In *AAAI*, pages 604–609, 2008.

[11] Marek Grzes and Daniel Kudenko. Plan-based Reward Shaping for Reinforcement Learning. In *International IEEE Conference on Intelligent Systems*, volume 2, pages 10–22, 2008.

[12] Sam Devlin and Daniel Kudenko. Dynamic Potential-based Reward Shaping. In *AAMAS*, pages 433–440, 2012.

[13] Marek Grzes. Reward Shaping in Episodic Reinforcement Learning. In *AAMAS*, pages 565–573, 2017.

[14] Alper Demir, Erkin Çilden, and Faruk Polat. Landmark Based Reward Shaping in Reinforcement Learning with Hidden States. In *AAMAS*, pages 1922–1924, 2019.

[15] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using natural language for reward shaping in reinforcement learning. In *IJCAI*, pages 2385–2391, 2019.

[16] Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Reward Shaping via Meta-Learning. *CoRR*, abs/1901.09330, 2019.

[17] Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-Logic-Based Reward Shaping for Continuing Reinforcement Learning Tasks. In *AAAI*, 2021.

[18] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1, 2004.

[19] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards. In *NeurIPS*, pages 10376–10386, 2019.

[20] Satinder P. Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically Motivated Reinforcement Learning. In *NeurIPS*, pages 1281–1288, 2004.

[21] Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pages 2601–2606. Cognitive Science Society, 2009.

[22] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[23] Jonathan Sorg, Satinder P Singh, and Richard L Lewis. Internal rewards mitigate agent boundedness. In *ICML*, 2010.

[24] Jonathan Sorg, Satinder P. Singh, and Richard L. Lewis. Reward Design via Online Gradient Ascent. In *NeurIPS*, pages 2190–2198, 2010.

[25] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. In *NeurIPS*, 2018.

[26] Farzan Memarian, Wonjoon Goo, Rudolf Lioutikov, Scott Niekum, and Ufuk Topcu. Self-supervised online reward shaping in sparse-reward environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2369–2375. IEEE, 2021.

[27] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *ICML*, pages 783–792. PMLR, 2019.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[29] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[30] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897. PMLR, 2015.

[31] Lilian Weng. Exploration strategies in deep reinforcement learning. *lilianweng.github.io*, 2020.

[32] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *NeurIPS*, 2016.

[33] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *ICML*, pages 2721–2730. PMLR, 2017.

[34] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *NeurIPS*, 2017.

[35] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[36] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[37] J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *ICML*, pages 513–520, 2009.

[38] Jonathan Sorg, Satinder Singh, and Richard L Lewis. Variance-based rewards for approximate bayesian reinforcement learning. In *UAI*, pages 564–571, 2010.

[39] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

[40] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[41] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[42] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[43] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, pages 2778–2787. PMLR, 2017.

[44] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *NeurIPS*, 2016.

[45] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[46] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. In *NeurIPS*, 2021.

[47] Marek Grześ and Daniel Kudenko. Online learning of shaping rewards in reinforcement learning. *Neural networks*, 23(4):541–550, 2010.

[48] Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

[49] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

[50] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850. PMLR, 2016.

[51] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[52] Amy McGovern and Andrew G. Barto. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *ICML*, pages 361–368, 2001.

[53] Özgür Simsek, Alicia P. Wolfe, and Andrew G. Barto. Identifying Useful Subgoals in Reinforcement Learning by Local Graph Partitioning. In *ICML*, volume 119, pages 816–823, 2005.

[54] Michael R. James and Satinder P. Singh. Sarsalandmark: An Algorithm for Learning in POMDPs with Landmarks. In *AAMAS*, pages 585–591, 2009.

[55] Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. Explicable reward design for reinforcement learning agents. In *NeurIPS*, 2021.

[56] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. In *ICML*, volume 80, pages 4254–4263, 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] The paper is organized according to the contributions listed at the end of the introduction section.
   (b) Did you describe the limitations of your work? [Yes] In Section 5, we discuss the limitations of our work and outline a future plan to address these limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This work presents a new online reward shaping technique for reinforcement learning agents. Given the algorithmic and empirical nature of our work applied to learning agents, we do not foresee any direct negative societal impacts of our work in the present form.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We confirm that our paper conforms with the ethics review guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All assumptions are stated either as part of the theorem statements or the proofs.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs of all theoretical results are included in the supplementary material.
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] The code and instructions are included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Details are provided in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Error bars are included in all the result graphs

(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] Details are provided in the supplementary material.

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? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (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]