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, we propose a novel framework, Exploration-Guided Reward Shaping (ExpRS), that operates in a fully self-supervised manner and can accelerate an agent's learning even in sparse-reward environments. The key idea of ExpRS is to learn an intrinsic reward function in combination with exploration-based bonuses to maximize the agent's utility w.r.t. extrinsic rewards. We theoretically showcase the utility of ExpRS in a special family of MDPs. 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 (LIRPG) 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 3.4).

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

2 Problem Setup

2.1 Online Reward Shaping

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., $\max_\pi J(\pi, \overline{R})$.

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

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.

2.2 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 $\Phi : \mathcal{S} \to \mathbb{R}$ that quantifies "goodness" of the states, the reward function $\widehat{R}^{\text{PBRS}}$ produced by the PBRS framework is given by: $\widehat{R}^{\text{PBRS}}(s,a) := \overline{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s,a) \cdot \Phi(s') - \Phi(s)$. 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. However, 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 state space even after obtaining extrinsic reward signals.

Reward shaping in fully self-supervised way: LIRPG [25] and PGRD [24]. The Learning Intrinsic Rewards for Policy Gradient (LIRPG) technique [25] considers a parametric reward function of the form $\widehat{R}^{\text{LIRPG}}(s,a) = \overline{R}(s,a) + R_\phi(s,a)$, and learns the parameter ϕ of the intrinsic reward function R_ϕ in a fully self-supervised manner. LIRPG method alternates between learning intrinsic reward parameter ϕ and the agent's policy optimization w.r.t. the learned reward $\widehat{R}^{\text{LIRPG}}(s,a) = \overline{R}(s,a) + R_\phi(s,a)$. At round k, for fixed π_k, LIRPG updates the parameter ϕ_{k-1} to ϕ_k by considering the effect such a change would have on the expected cumulative return (w.r.t. $\overline{R}$) of the learner through the change in the policy π_k, i.e., update ϕ using the gradient $\left[\nabla_\phi J(L(\pi_k, \widehat{R}^{\text{LIRPG}}), \overline{R})\right]_{\phi_{k-1}}$. In order to develop a concrete update rule for ϕ, LIRPG considers policy gradient style learning algorithm L with parametric policies $\left\{\pi_\theta : \theta \in \mathbb{R}^{d_\theta}\right\}$. For any parameter θ_k s.t. $\pi_k = \pi_{\theta_k}$, and for any ϕ, we update $\theta(\phi) = \theta_k + \alpha \cdot \left[\nabla_\theta J(\pi_\theta, \widehat{R}^{\text{LIRPG}})\right]_{\theta_k}$. Further, we set $L(\pi_k, \widehat{R}^{\text{LIRPG}}) = \pi_{\theta(\phi)}$. The LIRPG update, at round k, is given as follows: $\phi_k = \phi_{k-1} + \eta \cdot \left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}} \cdot \left[\nabla_{\theta(\phi)} J(\pi_{\theta(\phi)}, \overline{R})\right]_{\phi_{k-1}}$, where η is the learning rate. This form allows LIRPG to be applicable only to policy-based or policy-gradient style RL methods. There is a predecessor work to LIRPG, called Policy Gradient for Reward Design (PGRD [24]), which aims to design optimal rewards for RL agents with computational limitations resulting from small bounds on the depth of planning. To this end, they specifically consider a planning agent with h-step/depth Q-values in the tabular setting. In [24], the authors have empirically found that when the planning depth h is small, the role of the intrinsic reward component R_ϕ becomes more significant in $\widehat{R}^{\text{PGRD}}(s,a)$, where $\widehat{R}^{\text{PGRD}}(s,a) = \overline{R}(s,a) + R_\phi(s,a)$.

Reward shaping in fully self-supervised way: SORS [26]. The Self-supervised Online Reward Shaping (SORS) technique [26] considers a parametric reward function of the form $\widehat{R}^{\text{SORS}}(s,a) = R_\phi(s,a)$, and infers the intrinsic reward parameter ϕ using a classification-based reward inference algorithm, T-REX [27]. However, unlike T-REX that requires manual rankings over the trajectories, SORS uses the extrinsic reward $\overline{R}$ as a self-supervised learning signal to rank the trajectories generated by the agent during training. SORS can work with any RL method in contrast to LIRPG and PGRD techniques, which work only for specific types (policy-gradient based or planning-based) of RL agents. By design, SORS only enforces the relative (pair-wise) ordering

over the trajectories w.r.t. $\overline{R}$ when training R_ϕ. It ignores the weighting or importance of different trajectories w.r.t. the scaling of $\overline{R}$ at different parts of the state space. This makes training a policy challenging, when $\overline{R}$ has noisy/distractive signals.

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. The fully self-supervised reward shaping techniques (LIRPG, PGRD, and SORS) might fail to produce meaningful agent behavior in environments with extremely sparse rewards as they lack an explicit exploration 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. We overcome the limitations of these techniques by incorporating an explicit explorative component in our reward formulation. In contrast, the techniques that rely only on the intrinsic bonus [32–34] can mislead the agent towards sub-optimal or bad behaviors in noisy-distractive domains. We overcome the limitations of these techniques by appropriately exploiting the extrinsic reward signals. Further, we derive intuitive meta-gradients for updating the intrinsic reward component that enables our method to be broadly applicable, not just for specific RL agents as in LIRPG [25] and PGRD [24]. Similar to SORS [26], our technique is compatible with any RL agent as long we have access to their rollouts.

3 Exploration-Guided Reward Shaping

3.1 Our Reward Formulation

We consider the following parametric reward function for our Exploration-Guided Reward Shaping (ExpRS) technique (see Algorithm 1):

$$\widehat{R}^{\text{ExpRS}}(s,a) \;:=\; \overline{R}(s,a) + R_\phi^{\text{SelfRS}}(s,a) + B_w^{\text{Exp}}(s), \tag{1}$$

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. At round k of Algorithm 1, $\widehat{R}_{k-1}^{\text{ExpRS}}(s,a)$ 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 (ϕ_k, w_k). For the remainder of this section, we drop the superscripts in the reward/bonus terms of Eq. (1).

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 LIRPG [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 (inspired by PGRD [24]).

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 $N_w(s) = \left(\frac{\lambda}{B_{\max}}\right)^2 + w[\psi(s)]$ for some $B_{\max}, \lambda > 0$. 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 Parameter Updates for R_ϕ

In this section, we first obtain high-level meta-gradient updates for R_ϕ similar to LIRPG [25]. Then, we derive intuitive meta-gradient updates that would allow our Exploration-Guided Reward Shaping (ExpRS) technique to be compatible with any RL algorithm.

Notation. In the subscript of the expectations, let $\pi(a|s)$ mean $a \sim \pi(\cdot|s)$, $\mu^\pi(s,a)$ mean $s \sim d^\pi, a \sim \pi(\cdot|s)$, and $\mu^\pi(s)$ mean $s \sim d^\pi$. Further, we write $\mu_{s,a} = \mu^{\pi_{\theta_k}}(s,a)$, and $\mu_s = \mu^{\pi_{\theta_k}}(s)$.

High-level gradient updates for R_ϕ. We solve the bi-level optimization problem (P1.U)-(P1.L) of the intrinsic reward component in an 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:

$$\begin{aligned}\phi_k \;=\; \phi_{k-1} + \eta \cdot \left[\nabla_\phi J(\pi_{\theta(\phi)}, \overline{R})\right]_{\phi_{k-1}} &\overset{(a)}{=} \phi_{k-1} + \eta \cdot \left[\nabla_\phi \theta(\phi) \cdot \nabla_{\theta(\phi)} J(\pi_{\theta(\phi)}, \overline{R})\right]_{\phi_{k-1}} \\ &\overset{(b)}{\approx} \phi_{k-1} + \eta \cdot \underbrace{\left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}}}_{①} \cdot \underbrace{\left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k}}_{②},\end{aligned}$$

where η is the learning rate, (a) is due to chain rule, and (b) is due to the following smoothness condition: $\left\|\left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta(\phi_{k-1})} - \left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k}\right\|_2 \leq \|\theta(\phi_{k-1}) - \theta_k\|_2$. By using the meta-gradient derivations in [49–51], we write the term ① as follows: $\left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}} = \alpha \cdot \mathbb{E}_{\mu_{s,a}}\left[\left[\nabla_\phi Q^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}} \cdot \left[\nabla_\theta \log \pi_\theta(a|s)\right]^\top_{\theta_k}\right]$, where $\widehat{R}(s,a) := \overline{R}(s,a) + R_\phi(s,a) + B_{w_{k-1}}(s)$. By using the policy gradient theorem [52], we write the term ② as follows: $\left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k} = \mathbb{E}_{\mu_{s,a}}\left[\left[\nabla_\theta \log \pi_\theta(a|s)\right]_{\theta_k} \cdot Q^{\pi_{\theta_k}}_{\overline{R}}(s,a)\right]$. The above gradient update of ϕ_k, involving the terms ① and ②, resembles the LIRPG [25] update. However, both the terms ① and ② require computing the gradient of the policy network, i.e., $\nabla_\theta \log \pi_\theta(a|s)$. This requirement makes the above (LIRPG) update applicable only for policy-gradient based agents. Below, we aim to derive intuitive simplifications of the above two terms, ① and ②, that would enable our method to be applicable to any RL agent, and not just policy-gradient based agents.

Intuitive gradient updates for R_ϕ. In order to obtain intuitive forms of the terms ① and ②, we consider a simplified learner model. However, we note that our final algorithm does not rely on this simplified model, i.e., our final algorithm is applicable to any RL agent. In particular, we consider a learner model using tabular representation $\theta \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$, 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, reward function R_ϕ is still represented by a parametric function. For any $s \in \mathcal{S}, a \in \mathcal{A}$, let $\mathbf{1}_{s,a} \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ denote a vector with 1 in the (s,a)-th entry and 0 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 $A^{\pi_{\theta_k}}_{\overline{R}}(s,a) := Q^{\pi_{\theta_k}}_{\overline{R}}(s,a) - V^{\pi_{\theta_k}}_{\overline{R}}(s)$.

Proposition 1. *For the learner model described above, we have the following:*

$$\left[\nabla_\phi \theta(\phi)\right]_{\phi_{k-1}} \cdot \left[\nabla_\theta J(\pi_\theta, \overline{R})\right]_{\theta_k} \;=\; \alpha \cdot \mathbb{E}_{\mu^{\pi_{\theta_k}}(s,a)}\left[\mu^{\pi_{\theta_k}}(s,a) \cdot A^{\pi_{\theta_k}}_{\overline{R}}(s,a) \cdot \left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}}\right].$$

Finally, we get the expectation-based update of ϕ as follows (with $\eta' = \eta \cdot \alpha$):

$$\phi_k \;\approx\; \phi_{k-1} + \eta' \cdot \mathbb{E}_{\mu^{\pi_{\theta_k}}(s,a)}\left[\mu^{\pi_{\theta_k}}(s) \cdot \pi_{\theta_k}(a|s) \cdot A^{\pi_{\theta_k}}_{\overline{R}}(s,a) \cdot \left[\nabla_\phi A^{\pi_{\theta_k}}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}}\right]. \tag{2}$$

Remark. Compared to the high-level gradient update of ϕ as in LIRPG [25], the above intuitive update with a simplified learner model does not require computing the gradient of the policy network. This allows the above update to be applicable to any RL agent. Given the current policy π_k (possibly without any differentiable parameterization), we can write the above gradient update of ϕ as follows:

$$\phi_k \;\approx\; \phi_{k-1} + \eta' \cdot \mathbb{E}_{\mu^{\pi_k}(s,a)}\left[\mu^{\pi_k}(s) \cdot \pi_k(a|s) \cdot A^{\pi_k}_{\overline{R}}(s,a) \cdot \left[\nabla_\phi A^{\pi_k}_{\widehat{R},h}(s,a)\right]_{\phi_{k-1}}\right],$$

which only requires black-box access to (or sample rollouts from) policy π_k. In particular, we use $h = 1$ depth planning for Q-values in $\widetilde{L}$. Then, in the above gradient update of ϕ, we have: $\left[\nabla_\phi A^{\pi_k}_{\widehat{R},1}(s,a)\right]_{\phi_{k-1}} = \left[\nabla_\phi \left(R_\phi(s,a) - \mathbb{E}_{\pi_k(b|s)}[R_\phi(s,b)]\right)\right]_{\phi_{k-1}}$.

3.3 Our Algorithm with Empirical Updates

In this section, we present a concrete pseudocode of our Exploration-Guided Reward Shaping (ExpRS) technique. We closely follow the algorithmic structure of SORS [26]. Further, we focus on rollouts (each round corresponds to a single rollout), instead of environment steps, as in SORS. Below, we discuss the empirical updates for intrinsic reward and bonus components of our reward $\widehat{R}^{\text{ExpRS}}$.

Empirical updates for intrinsic reward R_ϕ. We translate the final expectation-based update of ϕ_k in Section 3.2 to its empirical counterpart using the rollout data $\mathcal{D}$ collected by executing the current policy π_k (or the recent policies) in the MDP M. In particular, we update the parameter ϕ as follows:

$$\phi_{k+1} \leftarrow \phi_k + \eta^\phi_k \sum_{(s^i_t, a^i_t, \xi^i_t)\in\mathcal{D}} \pi_k(a^i_t|s^i_t) \cdot \left(J(\xi^i_t, \overline{R}) - V_{\overline{R},\widetilde{\phi}_k}(s^i_t)\right) \cdot \left[\nabla_\phi A^{\pi_k}_{\widehat{R},1}(s^i_t, a^i_t)\right]_{\phi_k}, \quad (3)$$

where we absorb $\frac{1}{|\mathcal{D}|}$ into η^ϕ_k, ignore the term $\mu^{\pi_k}(s^i_t)$, denote $\xi^i_t = (s^i_t, a^i_t, \ldots, s^i_H)$, and set $\left[\nabla_\phi A^{\pi_k}_{\widehat{R},1}(s^i_t, a^i_t)\right]_{\phi_k} = \left[\nabla_\phi \left(R_\phi(s^i_t, a^i_t) - \mathbb{E}_{\pi_k(b|s^i_t)}\left[R_\phi(s^i_t, b)\right]\right)\right]_{\phi_k}$. Similar to LIRPG [25], we also maintain a critic $V_{\overline{R},\widetilde{\phi}_k}(\cdot)$ to approximate $V^{\pi_k}_{\overline{R}}(\cdot)$. We update the parameters of the critic, $\widetilde{\phi}_k$ to $\widetilde{\phi}_{k+1}$, using the same rollout data $\mathcal{D}$ and learning rate $\eta^{\widetilde{\phi}}_k$.

Empirical updates for intrinsic bonus B_w. In comparison to the intrinsic reward, we update the intrinsic bonus at a more fine-grained level, i.e., we update B_w at each environment/rollout step t within each round k. However, for clear presentation, we write the B_w update at the level of round k, not at the level of environment step t. We update B_w based on the history of all the states visited up to round k. Similar to #Exploration [34], we use the count-based intrinsic bonuses with a state abstraction $\psi : \mathcal{S} \to \mathcal{X}_\psi$. We maintain the visitation count of the abstracted states in w. For each rollout ξ^k, we update the parameter w of the intrinsic bonus as follows (component-wise):

$$w_{k+1}[x] = w_k[x] + \sum_{s^k_t \in \xi^k} \mathbf{1}\left\{\psi(s^k_t) = x\right\}, \quad \forall x \in \mathcal{X}_\psi. \quad (4)$$

Based on the above empirical updates for intrinsic reward and bonus components, we derive our final algorithm given in Algorithm 2. For stability reasons, we update the policy more frequently compared to the intrinsic reward, i.e., $N_\pi < N_r$. Similar to the existing count-based exploration techniques [32–34], we use a lookahead step when incorporating the bonus term (see line 14 in Algorithm 2).

3.4 Theoretical Analysis

In this section, we theoretically showcase the utility of our exploration-guided reward shaping. 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 zones in the state space.

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 ($H \to \infty$, and $\gamma < 1$).

Learning algorithm and reward shaping techniques. For our theoretical analysis of different reward shaping techniques, we consider a simplified TD-style RL algorithm L given in Appendix C. The algorithm L takes two binary flags, `no-self` and `no-exp`, as input that correspond to the intrinsic reward and bonus components, R and B, respectively. For example, $L(1, 1)$ corresponds to the setting of learning with extrinsic reward $\overline{R}$ only. The following theorem compares the sample complexity of algorithm L under different reward shaping techniques to learn an optimal policy.

Algorithm 2 Exploration-Guided Reward Shaping (EXPRS)

Inputs and hyperparameters: RL algorithm L; first-in-first-out buffer $\mathcal{D}$ with size D_{max}; abstraction ψ; learning rates $\{\eta_k^{\phi}\}$, $\{\eta_k^{\widetilde{\phi}}\}$; bonus parameters B_{max}, λ; update frequencies N_r, N_π
Initialization: initialize the intrinsic reward/bonus parameters $(\phi_0, \widetilde{\phi}_0, w_0)$, and the policy π_0
for $k = 1, 2, \ldots, K$ **do**
  // data collection
  Rollout the policy π_k in the MDP M to obtain a trajectory $\xi^k = \left(s_0^k, a_0^k, s_1^k, a_1^k, \ldots, s_H^k\right)$
  Store ξ^k in the buffer $\mathcal{D}.\texttt{add}(\xi^k)$; if the buffer $\mathcal{D}$ is full, remove the oldest trajectory
  // intrinsic bonus update
  Update w_k according to Eq. (4) using the states visited in the trajectory ξ^k
  Define bonus $B_{w_{k+1}}(s) = \frac{\lambda}{\sqrt{N_{w_{k+1}}(s)}}$, where $N_{w_{k+1}}(s) = \left(\frac{\lambda}{B_{\text{max}}}\right)^2 + w_{k+1}[\psi(s)]$
  // intrinsic reward update
  if $k\%N_r = 0$ **then**
    Update ϕ_k according to Eq. (3) using full data in $\mathcal{D}$
    Update critic $V_{\overline{R},\widetilde{\phi}_k}$ to $V_{\overline{R},\widetilde{\phi}_{k+1}}$ using full data in $\mathcal{D}$ and the learning rate $\eta_k^{\widetilde{\phi}}$
  else
    Maintain the previous values $\phi_{k+1} \leftarrow \phi_k$ and $\widetilde{\phi}_{k+1} \leftarrow \widetilde{\phi}_k$
  // policy update
  if $k\%N_\pi = 0$ **then**
    Define reward $\widehat{R}_{k+1}(s, a, s') := \overline{R}(s, a) + R_{\phi_{k+1}}(s, a) + B_{w_{k+1}}(s')$
    Update policy $\pi_{k+1} \leftarrow L(\pi_k, \widehat{R}_{k+1})$ using the latest rollouts in $\mathcal{D}$
  else
    Maintain the previous value $\pi_{k+1} \leftarrow \pi_k$
Output: Policy π_K

Theorem 1. *Consider the chain environment M defined above, and the learning algorithm L given in Appendix C. 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, with appropriate choice of hyperparameters, we have the following: (i) without any reward shaping, $\mathbb{E}\left[\text{cost}(L(1,1))\right] \geq 2^{n_1-1}$; (ii) with exploration-only reward shaping, cost$(L(1,0)) = n_1 \cdot (n_1 + n_2 + 2)$; (iii) with fully self-supervised reward shaping, $\mathbb{E}\left[\text{cost}(L(0,1))\right] \geq 2^{n_1-1}$; (iv) with exploration-guided reward shaping, cost$(L(0,0)) \leq n_1 + n_2 + 2$.*

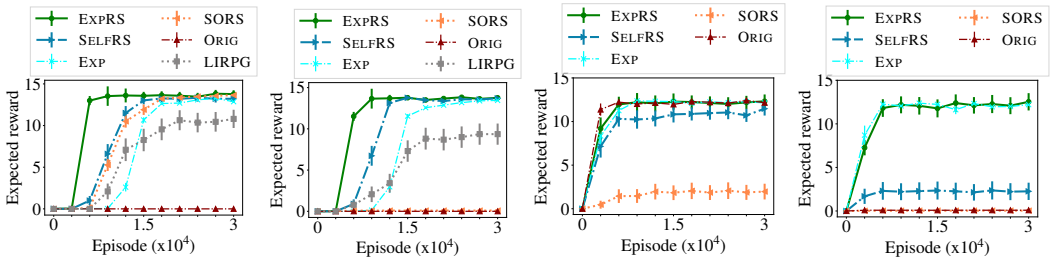
4 Experimental Evaluation

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 3.4; 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, 53–56]—the environment used in our experiments is adapted from [56]. LINEK corresponds to a navigation task in a one-dimensional space where the agent has to first pick the correct 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, 56, 57]—the environment used in our experiments is adapted from [56]. 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

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.

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 intended neighboring node with probability of $(1 - p_{\text{rand}})$ for $p_{\text{rand}} = 0.05$. The agent receives a reward of $R_{\max} = 1$ 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 $\gamma = 0.99$ and the environment resets after a horizon of $H = 40$. We consider two different variants of the chain environment: (i) CHAIN^0 with $(n_1 = 20, n_2 = 40, R_{\text{dis}} = 0)$; (ii) CHAIN$^+$ with $(n_1 = 20, n_2 = 40, R_{\text{dis}} = 0.01)$. We defer the full environment details to appendices.

Evaluation setup and results. We conduct our experiments with two different types of RL agents for CHAIN: tabular REINFORCE agent [7] and tabular Q-learning agent [7]. Algorithm 2 provides a sketch of the overall training process, and shows how agent's training interleaves with reward shaping techniques. We compare the performance of the following reward shaping techniques: (i) $\widehat{R}^{\text{ORIG}} := \overline{R}$ is a default baseline without any shaping; (ii) $\widehat{R}^{\text{SORS}}$ is obtained via the SORS technique [26] (see Section 2.2); (iii) $\widehat{R}^{\text{LIRPG}}$ is obtained via adapting the LIRPG technique [25] to our training pipeline (see Algorithm 2, Sections 2.2 and 3.2)—note that $\widehat{R}^{\text{LIRPG}}$ is not applicable to Q-learning agent; (iv) $\widehat{R}^{\text{EXP}} := \overline{R} + B_w^{\text{EXP}}$ uses only the intrinsic bonuses; (v) $\widehat{R}^{\text{SELFRS}} := \overline{R} + R_\phi^{\text{SELFRS}}$ uses only the intrinsic rewards; (vi) $\widehat{R}^{\text{EXPRS}} := \overline{R} + R_\phi^{\text{SELFRS}} + B_w^{\text{EXP}}$ combines intrinsic bonuses with intrinsic rewards. We provide full details of hyperparameters in appendices. 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. These results demonstrate the effectiveness of our exploration-guided reward shaping framework ($\widehat{R}^{\text{EXPRS}}$), in comparison to baselines ($\widehat{R}^{\text{ORIG}}$, $\widehat{R}^{\text{SORS}}$, $\widehat{R}^{\text{LIRPG}}$, $\widehat{R}^{\text{EXP}}$, $\widehat{R}^{\text{SELFRS}}$). We provide a more detailed analysis of these results in appendices.

4.2 Evaluation on ROOM

ROOM (Figure 3). This environment is based on the work of [56]; 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: (i) ROOM^0 has $R_{\text{dis}} = 0$ at the distractor state shown as "green-plus" (equivalently, there is no distractor state); (ii) 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 full details to appendices.

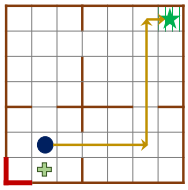
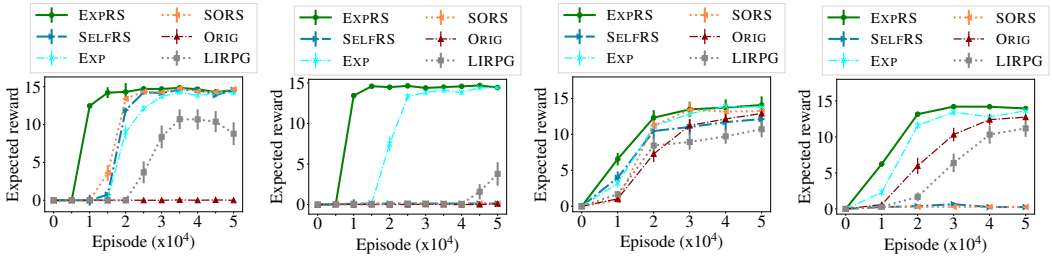


Figure 3: ROOM^0 / ROOM$^+$

Evaluation setup and results. Our evaluation setup for this environment is exactly same as that used for 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 obtained in Figure 2, further demonstrate the effectiveness and robustness of $\widehat{R}^{\text{EXPRS}}$ across different environments in comparison to baselines.



(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 distractor states). See Sections 4.2 and 4.3 for details.

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 correct key and then reach the goal. The environment used in our experiments is based on the work of [56]; however, we adapted it to have multiple keys (only one being correct) and "distractor" states that provide a small reward at goal locations even without the correct key. The environment comprises of the following main elements: (a) an agent whose current location (shown as "blue-circle") is a point x in $[0, 1]$; (b) goal (shown as "green-star") is available in locations on the segment $[0.9, 1]$; (c) a set of k keys that are available in locations on the segment $[0.0, 0.1]$, (d) among k keys, only 1 key is correct and the remaining $k - 1$ keys are wrong (i.e., irrelevant at the goal). Moreover, we consider the agent with two different actions related to picking a key: (a) "pickCorrect" makes the agent collect the correct key required at the goal; (b) "pickWrong" makes the agent collect one of the $k - 1$ wrong keys, chosen at random. Similar to Sections 4.1 and 4.2, we use two adaptations of the environment: (i) LineK^0 with $(k = 10, R_{\text{dis}} = 0)$; (ii) LineK$^+$ with $(k = 10, R_{\text{dis}} = 0.01)$. We defer full details to appendices.

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 (i.e., one fully connected hidden layer with 256 nodes and RELU activation) [7]. Similar to Section 4.1, we compare the performance of six techniques. As a crucial difference, here we use neural-network based reward functions for $\widehat{R}^{\text{SORS}}$, $\widehat{R}^{\text{LIRPG}}$, $\widehat{R}^{\text{SelfRS}}$, and $\widehat{R}^{\text{ExpRS}}$. Based on [25, 26], we use the same neural-network architecture for intrinsic reward functions as used for the agent's policy by applying appropriate transformations at the output layer (e.g., instead of using soft-max, use $tanh$-clipping to get output reward values for actions). Figures 4c and 4d show the agent's performance for environments LineK^0 and LineK$^+$ (averaged over 20 runs). In particular, Figure 4d shows results for LineK$^+$ – the performance of the three reward shaping techniques ($\widehat{R}^{\text{SORS}}$, $\widehat{R}^{\text{LIRPG}}$, $\widehat{R}^{\text{SelfRS}}$) is even worse compared to that of $\widehat{R}^{\text{Orig}}$. These results showcase the effectiveness of $\widehat{R}^{\text{ExpRS}}$ in more complex learning settings that require neural representations for reward functions. We provide a more in-depth analysis in appendices.

5 Concluding Discussions

We proposed a novel reward shaping framework, ExpRS, that operates in a fully self-supervised manner and could accelerate an agent's learning even in sparse-reward environments. Next, we discuss a few limitations of our work and outline a future plan to address them. First, the experimental evaluation is conducted on simpler environments to study the performance of techniques w.r.t. the three characteristics of (a) hard exploration, (b) local minima, and (c) noisy TV problem. It would be interesting to evaluate different reward design techniques in more complex environments (e.g., with continuous state/action spaces); this would also require designing benchmark environments that systematically capture the above three characteristics. Second, ExpRS combines the intrinsic rewards and intrinsic bonuses that allows it to overcome the limitations of state-of-the-art techniques. It would be interesting to develop more principled ways to combine these two signals. Third, it would be useful to provide rigorous analysis of ExpRS in terms of convergence speed and stability of an agent.

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] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 1999.

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

[54] 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.

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

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

[57] 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]