Explicable Reward Design for Reinforcement Learning Agents

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the design of *explicable* reward functions for a reinforcement learning agent while guaranteeing that an optimal policy induced by the function belongs to a set of target policies. By being explicable, we seek to capture two properties: (a) *informativeness* so that the rewards speed up the agent's convergence, and (b) *sparseness* so that the rewards are easy to interpret and debug. The key challenge is that higher informativeness typically requires dense rewards for many learning tasks, and existing techniques do not allow one to balance these two properties appropriately. In this paper, we investigate the problem from the perspective of discrete optimization and introduce a novel framework, ExpRD, to design explicable reward functions. ExpRD builds upon an informativeness criterion that captures the (sub-)optimality of target policies at different time horizons in terms of actions taken from any given starting state. We provide a mathematical analysis of ExpRD, and show its connections to existing reward design techniques including potential-based reward shaping. Experimental results on two navigation tasks demonstrate the effectiveness of ExpRD in designing explicable reward functions.

1 Introduction

A reward function plays the central role during the learning/training process of a reinforcement learning (RL) agent. Given a "task" the agent is expected to perform (i.e., the desired learning outcome), there are typically many different reward specifications under which an optimal policy has the same performance guarantees on the task. This freedom in choosing the reward function, in turn, leads to the fundamental question of reward design: *What are different criteria that one should consider in designing a reward function for the agent, apart from the agent's final output policy?* [1–3].

One of the important criteria is *informativeness*, capturing that the rewards should speed up the agent's convergence [1–6]. For instance, a major challenge faced by an RL agent is because of delayed rewards during training; in the worst-case, the agent's convergence is slowed down exponentially w.r.t. the time horizon of delay [7]. In this case, we seek to design a new reward function that reduces this time horizon of delay while guaranteeing that any optimal policy induced by the designed function is also optimal under the original reward function [3]. The classical technique of potential-based reward shaping (when applied with appropriate state potentials) indeed allows us to reduce this time horizon of delay to 1; see [3, 8] and Section 2. With 1, it means that globally optimal actions for any state are also myopically optimal, thereby making the agent's learning process trivial.

While informativeness is an important criterion, it is not the only criterion to consider when designing rewards for many practical applications. Another natural criterion is *sparseness* of the reward function, ensuring that the rewards are easy to interpret and debug. There are several important practical settings where this criterion is crucial: (i) when rewards are designed for human agents who are learning to perform sequential tasks, for instance, in pedagogical applications such as educational games [9] and virtual reality-based training simulators [10, 11], (ii) when rewards are designed for practitioners

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

who then "program" these rewards into software, for instance, in robotics applications [1, 3], and (iii) when investigating/developing optimal reward-poisoning attacks for tasks that are expected to have sparse rewards [12–16]. Beyond these practical settings, many naturally occurring reward functions in real-life tasks are inherently sparse and interpretable, further motivating the need to distill these properties in the automated reward design process. The key challenge is that higher informativeness typically requires dense rewards for many learning tasks – for instance, the above-mentioned reward function that achieves a time horizon of 1 would require that most of the states be associated with some real-valued reward (see Sections 2 and 4). To this end, an important research question that we seek to address is: *How to balance these two criteria of informativeness and sparseness in the reward design process while guaranteeing an optimality criterion on policies induced by the reward function?*

In this paper, we formalize the problem of designing *explicable* reward functions, focusing on the criteria of informativeness and sparseness. We investigate this problem from an expert/teacher's point of view who has full domain knowledge (in this case, an original reward function along with optimal policies induced by the original function), and seeks to design a new reward function for the agent—see further discussion in Section 5 on expert-driven vs. agent-driven reward design. We tackle the problem from the perspective of discrete optimization and introduce a novel framework, ExpRD, to design reward functions. ExpRD allows us to appropriately balance informativeness and sparseness while guaranteeing that an optimal policy induced by the function belongs to a set of target policies. ExpRD builds upon an informativeness criterion that captures the (sub-)optimality of target policies at different time horizons from any given starting state. Our main results and contributions are summarized below:

I. We formulate the problem of explicable reward functions to balance the two important criteria of informativeness and sparseness in the reward design process. (Sections 2 and 3.1)

II. We propose a novel optimization framework, ExpRD, to design reward functions. As part of this framework, we introduce a new criterion capturing informativeness of reward functions that is amenable to optimization techniques and is of independent interest. (Sections 3.2 and 3.3)

III. We provide a detailed mathematical analysis of ExpRD and show its connections to popular techniques including potential-based reward shaping. (Sections 3.3 and 3.4)

IV. We provide a practical extension to apply our framework to large state spaces. We perform extensive experiments on two navigation tasks to demonstrate the effectiveness of ExpRD in designing explicable reward functions. (Sections 3.5 and 4)

2 Problem Setup

Environment. An environment is defined as a Markov Decision Process (MDP) $M := (\mathcal{S}, \mathcal{A}, T, \gamma, R)$, where the set of states and actions 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. Here, γ is the discounting factor. The underlying reward function is given by $R : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$, for some $R_{\max} > 0$. We interchangeably represent the reward function by a vector $R \in \mathbb{R}^{|\mathcal{S}| \cdot |\mathcal{A}|}$, whose $(s\,|\mathcal{A}| + a)$-th entry is given by $R(s, a)$. We define the support of R as $\mathrm{supp}(R) := \{s : s \in \mathcal{S}, R(s, a) \neq 0 \text{ for some } a \in \mathcal{A}\}$, and the ℓ_0-norm of R as $\|R\|_0 := |\mathrm{supp}(R)|$.

Preliminaries and definitions. 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 policy π, the state value function V^π_∞ and the action value function Q^π_∞ in the MDP M are defined as follows respectively: $V^\pi_\infty(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, T, \pi\right]$ and $Q^\pi_\infty(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, T, \pi\right]$. Further, the optimal value functions are given by $V^*_\infty(s) = \sup_\pi V^\pi_\infty(s)$ and $Q^*_\infty(s, a) = \sup_\pi Q^\pi_\infty(s, a)$. There always exists a deterministic stationary policy π that achieves the optimal value function simultaneously for all $s \in \mathcal{S}$ [7, 17], and we denote all such deterministic optimal policies by the set $\Pi^* := \{\pi : \mathcal{S} \to \mathcal{A} \text{ s.t. } V^\pi_\infty(s) = V^*_\infty(s), \forall s \in \mathcal{S}\}$. From here onwards, we focus on deterministic policies unless stated otherwise. For any policy π and reward R, we define the following quantities that capture the ∞-step (global) optimality gap and the 0-step (myopic) optimality gap of action a at state s, respectively:

$$\delta^\pi_\infty(s, a) := Q^\pi_\infty(s, \pi(s)) - Q^\pi_\infty(s, a), \text{ and } \delta^\pi_0(s, a) := Q^\pi_0(s, \pi(s)) - Q^\pi_0(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A},$$

where $Q^\pi_0(s, a) = R(s, a)$ is the 0-step action value function of policy π. The $\delta^\pi_\infty(s, a)$ values

are same for all $\pi \in \Pi^*$, and we denote it by $\delta^*_\infty(s,a) = V^*_\infty(s) - Q^*_\infty(s,a)$; whereas this is not the case with $\delta^\pi_0(s,a)$ values in general. For any state $s \in \mathcal{S}$ and a set of policies Π, we define $\Pi_s := \{a : a = \pi(s), \pi \in \Pi\}$. Then, we have that $\delta^*_\infty(s,a) = 0, \forall a \in \Pi^*_s, s \in \mathcal{S}$.

Explicable reward design. Consider an MDP M with a sparse reward function $\overline{R}$ that has non-zero rewards only on a small number of states $\mathcal{G} \subseteq \mathcal{S}$, i.e., $\overline{R}(s,a) = 0, \forall s \in \mathcal{S} \backslash \mathcal{G}, a \in \mathcal{A}$. The quantities defined above corresponding to $R := \overline{R}$ are denoted by an overline, e.g., the optimal policy set by $\overline{\Pi}^*$, and the ∞-step optimality gaps by $\overline{\delta}^*_\infty$. When the state space $\mathcal{S}$ is very large, learning an optimal policy induced by the sparse reward $\overline{R}$ is challenging due to high sample complexity. We study the explicable reward design problem: given $\overline{R}$ and the corresponding optimal policy set $\overline{\Pi}^*$ as the input, an expert designs a new reward function $\widehat{R}$ with some *preferred characteristics*, while guaranteeing certain *invariance* requirement. We consider informativeness and sparseness (see Section 1) as the preferred characteristics of $\widehat{R}$. The invariance requirement is that any optimal policy learned using the new reward $\widehat{R}$ belongs to the optimal policy set $\overline{\Pi}^*$ induced by $\overline{R}$. The quantities defined above corresponding to $R := \widehat{R}$ are denoted by a widehat, e.g., the optimal policy set by $\widehat{\Pi}^*$.

Typical techniques for reward design and issues. Given a set of important states (subgoals) in the environment, one could design a handcrafted reward function $\widehat{R}_{\text{CRAFT}}$ by assigning non-zero reward values only to these states. Even though this simple approach produces a sufficiently sparse and interpretable reward function, it often fails to satisfy the invariance requirement. In particular, there are some well-known "reward bugs" that can arise in this approach and mislead the agent into learning sub-optimal policies (see [2, 3]). In the seminal work [3], the authors introduced the potential-based reward shaping (PBRS) method to alleviate this issue. The reward function produced by the PBRS method with optimal value function $\overline{V}^*_\infty$ under $\overline{R}$ as the potential function is defined as follows:

$$\widehat{R}_{\text{PBRS}}(s,a) \;:=\; \overline{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s,a) \cdot \overline{V}^*_\infty(s') - \overline{V}^*_\infty(s). \tag{1}$$

The set of optimal policies $\widehat{\Pi}^*$ induced by $\widehat{R}_{\text{PBRS}}$ is exactly equal to the set of optimal policies $\overline{\Pi}^*$ induced by $\overline{R}$ since $\widehat{\delta}^\pi_\infty(s,a) = \overline{\delta}^*_\infty(s,a)$ for all $\pi \in \overline{\Pi}^*$ [3]. In addition, for any state $s \in \mathcal{S}$, globally optimal actions $\overline{\Pi}^*_s$ under $\overline{R}$ are also myopically optimal under $\widehat{R}_{\text{PBRS}}$ since $\widehat{\delta}^\pi_0(s,a) = \overline{\delta}^*_\infty(s,a)$ for all $\pi \in \overline{\Pi}^*$ [3, 8] — this leads to a dramatic speed-up in the learning process of optimal behavior. However, the key issue with the potential-based reward shaping is that it produces a very dense reward which is less interpretable (see Section 1).

3 Our Reward Design Framework ExpRD

In Sections 3.1, 3.2 and 3.3, we propose an optimization formulation and a greedy solution for the explicable reward design problem. In Section 3.4, we provide a theoretical analysis of our greedy solution. In Section 3.5, we provide a practical extension to apply our framework to large state spaces.

3.1 Discrete Optimization Formulation

Given $\overline{R}$ and the corresponding optimal policy set $\overline{\Pi}^*$, we systematically develop a discrete optimization framework (ExpRD) to design an explicable reward function $\widehat{R}$ (see Section 2).

Sparseness, informativeness, and invariance. The sparseness of the reward $\widehat{R}$ is captured by $\text{supp}(\widehat{R})$. In Section 3.2, we formalize an informativeness criterion $I(\widehat{R})$ of $\widehat{R}$ that captures how hard/easy it is to learn an optimal behavior induced by $\widehat{R}$. We explicitly enforce the invariance requirement (see Section 2) for the new reward $\widehat{R}$ by choosing a set of candidate policies $\Pi^\dagger \subseteq \overline{\Pi}^*$, and satisfying the following (Bellman-optimality) conditions:

$$Q^{\pi^\dagger}_\infty(s,a) \;=\; \widehat{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s,a) Q^{\pi^\dagger}_\infty(s', \pi^\dagger(s')), \quad \forall a \in \mathcal{A}, s \in \mathcal{S}, \pi^\dagger \in \Pi^\dagger \tag{C.1}$$

$$Q^{\pi^\dagger}_\infty(s, \pi^\dagger(s)) \;\geq\; Q^{\pi^\dagger}_\infty(s,a) + \overline{\delta}^*_\infty(s), \quad \forall a \in \mathcal{A} \backslash \overline{\Pi}^*_s, s \in \mathcal{S}, \pi^\dagger \in \Pi^\dagger, \tag{C.2}$$

where $\overline{\delta}^*_\infty(s) := \min_{a \in \mathcal{A} \backslash \overline{\Pi}^*_s} \overline{\delta}^*_\infty(s,a), \forall s \in \mathcal{S}$. The above conditions guarantee that any optimal policy induced by $\widehat{R}$ is also optimal under $\overline{R}$, i.e., $\Pi^\dagger \subseteq \widehat{\Pi}^* \subseteq \overline{\Pi}^*$. Here, the set $\Pi^\dagger$ is used to reduce the number of constraints. Note that for the potential-based shaped reward $\widehat{R}_{\text{PBRS}}$, we have: $\widehat{\Pi}^* = \overline{\Pi}^*$.

Maximizing informativeness for a given set of important states. When a domain expert provides us a set of important states (subgoals) in the environment [18–21], we want to use this set in a principled way to design a reward $\widehat{R}$, while avoiding the "reward bugs" that can arise from hand-crafted rewards $\widehat{R}_{\text{CRAFT}}$. To this end, for any given set of subgoals $\mathcal{Z} \subseteq \mathcal{S}$, we optimize the informativeness criterion $I(R)$ while satisfying the invariance requirement:

$$\begin{aligned} g(\mathcal{Z}) \;:=\; & \max_{R:\text{supp}(R)\subseteq\mathcal{Z}\cup\mathcal{G}} \quad I(R) \\ & \text{subject to} \quad \text{conditions (C.1)} - \text{(C.2) hold with } \widehat{R} = R \\ & \qquad\qquad\quad |R(s,a)| \;\leq\; R_{\max},\, \forall s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned} \tag{P1}$$

Let $R^{(\mathcal{Z})}$ denote the R that maximizes $g(\mathcal{Z})$. Let $\mathcal{R} \subseteq \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ be a constraint set on R that captures only the conditions (C.1)-(C.2), and the $R_{\max}$ bound.

Jointly finding subgoals along with maximizing informativeness. Based on (P1), we propose the following discrete optimization formulation that allows us to select a set of important states (of size B) and design a reward function that maximizes informativeness automatically:

$$\max_{\mathcal{Z}:\mathcal{Z}\subseteq\mathcal{S}\setminus\mathcal{G},|\mathcal{Z}|\leq B} \; g(\mathcal{Z}). \tag{P2}$$

We can incorporate prior knowledge about the quality of subgoals using a set function $D : 2^{\mathcal{S}} \to \mathbb{R}$ (we assume D to be a submodular function [22]). Finally, the full ExpRD formulation is given by:

$$\max_{\mathcal{Z}:\mathcal{Z}\subseteq\mathcal{S}\setminus\mathcal{G},|\mathcal{Z}|\leq B} \; g(\mathcal{Z}) + \lambda D(\mathcal{Z}\cup\mathcal{G}), \;\; \text{for some } \lambda \geq 0. \tag{P3}$$

We study the problems (P1), (P2), and (P3) in the following subsections.

3.2 Informativeness Criterion

Understanding the informativeness of a reward function is an important problem, and several works have investigated it [4, 5, 23]. Our goal is to define an informativeness criterion that is amenable to optimization techniques. As noted in Section 2, for any policy $\pi \in \overline{\Pi}^*$, 0-step and ∞-step optimality gaps induced by $\widehat{R}_{\text{PBRS}}$, and ∞-step optimality gaps induced by $\overline{R}$ are all equal, i.e., $\widehat{\delta}_0^{\pi}(s,a) = \widehat{\delta}_{\infty}^{\pi}(s,a) = \overline{\delta}_{\infty}^{*}(s,a)$. For any reward function R, one could ask how much these two quantities could differ, and even consider the intermediate cases between 0-step and ∞-step optimality. Inspired by the h-step optimality notions studied in [4, 23], we define the h-step action value function of any policy π as $Q_h^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{h} \gamma^t R(s_t,a_t) | s_0 = s, a_0 = a, T, \pi\right]$, and it satisfies the following recursive relationship: $Q_h^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} T(s'|s,a) \cdot Q_{h-1}^{\pi}(s',\pi(s'))$. Let $\mathcal{H}$ be a set of horizons for which we want to maximize informativeness. For any policy π and reward R, we define the following quantity that captures the h-step optimality gap of action a at state s: $\delta_h^{\pi}(s,a) := Q_h^{\pi}(s,\pi(s)) - Q_h^{\pi}(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}, h \in \mathcal{H}$. Later, in the proof of Proposition 2, we show that $\widehat{\delta}_h^{\pi}(s,a)$ is linear in R, i.e., $\widehat{\delta}_h^{\pi}(s,a) = \langle w_h(s,a), R\rangle$ for some vector $w_h(s,a) \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$. Interestingly, the following proposition states that, for any policy $\pi \in \overline{\Pi}^*$ and any h, the h-step optimality gap induced by $\widehat{R}_{\text{PBRS}}$ is equal to the ∞-step optimality gap induced by $\overline{R}$:

Proposition 1. *The original reward function $\overline{R}$, and the potential-based shaped reward function $\widehat{R}_{\text{PBRS}}$ given in* (1) *satisfy the following: $\widehat{\delta}_h^{\pi}(s,a) = \overline{\delta}_{\infty}^{*}(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}, \pi \in \overline{\Pi}^*, h \in \mathcal{H}$.*

Let $\ell : \mathbb{R} \to \mathbb{R}$ be a monotonically non-decreasing concave function. Then, based on the h-step optimality gaps, we define the informativeness criterion of the reward R as follows:

$$I_{\ell}(R) \;:=\; \sum_{\pi^{\dagger}\in\Pi^{\dagger}} \sum_{h\in\mathcal{H}} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}\setminus\overline{\Pi}_s^*} \ell(\delta_h^{\pi^{\dagger}}(s,a)).$$

From here onwards, we let I be I_{ℓ} in the problem (P1). As an example for ℓ, we consider the negated hinge loss given by $\ell_{\text{hg}}(\delta(s,a)) := -\max(0, \overline{\delta}_{\infty}^{*}(s,a) - \delta(s,a))$. By Proposition 1, we have that $I_{\ell_{\text{hg}}}(\widehat{R}_{\text{PBRS}}) = 0$, and for any other R, $I_{\ell_{\text{hg}}}(R) \leq 0$, i.e., $\widehat{R}_{\text{PBRS}}$ achieves the maximum value of $I_{\ell_{\text{hg}}}$.

3.3 Iterative Greedy Algorithm

First, we show that the problem (P1) can be efficiently solved using the standard concave optimization methods to find $R^{(\mathcal{Z})}$ for any given $\mathcal{Z} \subseteq \mathcal{S}$:

Proposition 2. *For any given $\mathcal{Z} \subseteq \mathcal{S}$, the problem* (P1) *is a concave optimization problem in $R \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ with linear constraints. Further, the feasible set of the problem* (P1) *is non-empty.*

Then, inspired by the Forward Stepwise Selection method from [24], we propose an iterative greedy solution (see Algorithm 1) to solve the problems (P2) and (P3). To compute the incremental gain at each step, we would need to solve the concave optimization problem (P1) for different values of $\mathcal{Z}$. The problem (P1) has $|\mathcal{S}| \cdot |\mathcal{A}|$ optimization variables, and $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{A}| \cdot \left|\Pi^{\dagger}\right| \cdot |\mathcal{H}|)$ constraints.

Algorithm 1 Iterative Greedy Algorithm for ExpRD

1: **Input:** original MDP $\overline{M}$, $\overline{\delta}^{*}_{\infty}(s)$ values, sets $\overline{\Pi}^{*}, \overline{\Pi}^{\dagger}, \mathcal{G}, \mathcal{H}$, sparsity budget B
2: **Initialize:** $\mathcal{Z}_0 \leftarrow \mathcal{G}$
3: **for** $k = 1, 2, \ldots, B$ **do**
4: $\quad z_k \leftarrow \arg\max_{z \in \mathcal{S} \setminus \mathcal{Z}_{k-1}} g(\mathcal{Z}_{k-1} \cup \{z\}) + \lambda D(\mathcal{Z}_{k-1} \cup \mathcal{G} \cup \{z\}) - g(\mathcal{Z}_{k-1}) - \lambda D(\mathcal{Z}_{k-1} \cup \mathcal{G})$
5: $\quad \mathcal{Z}_k \leftarrow \mathcal{Z}_{k-1} \cup \{z_k\}$
6: **Output:** $\mathcal{Z}_B$, and the corresponding optimal reward function $R^{(\mathcal{Z}_B)}$.

3.4 Theoretical Analysis

Here, we provide guarantees for the solution returned by our Algorithm 1. Below, we give an overview of the main technical ideas, and leave a detailed discussion along with proofs in Appendix of the supplementary material. We define a normalized set function $\bar{f} : 2^{\mathcal{S}} \to \mathbb{R}$ as follows:

$$\bar{f}(\mathcal{Z}) \;=\; \max_{R:\mathrm{supp}(R) \subseteq \mathcal{Z} \cup \mathcal{G}, R \in \mathcal{R}} \left(I_\ell(R) - I_\ell(R^{(\mathcal{G})})\right) + \lambda(D(\mathcal{Z} \cup \mathcal{G}) - D(\mathcal{G})), \tag{2}$$

where $R^{(\mathcal{G})} = \arg\max_{R:\mathrm{supp}(R) \subseteq \mathcal{G}, R \in \mathcal{R}} I_\ell(R)$. Note that the optimization problem (P3) is equivalent to $\max_{\mathcal{Z}:\mathcal{Z} \subseteq \mathcal{S} \setminus \mathcal{G}, |\mathcal{Z}| \leq B} \bar{f}(\mathcal{Z})$. For a given sparsity budget B, let $\mathcal{Z}_B^{\mathrm{Greedy}}$ be the set selected by our Algorithm 1, and $\mathcal{Z}_B^{\mathrm{OPT}}$ be the optimal set that maximizes (P3). The corresponding $\bar{f}$ values of these sets are denoted by $\bar{f}_B^{\mathrm{Greedy}}$ and $\bar{f}_B^{\mathrm{OPT}}$ respectively; in the following, we are interested in comparing these two values. The problem (P3) is closely related to the subset selection problem studied in [24] with a twist of an additional constraint set $\mathcal{R}$ (see the discussion after (P1)), making the theoretical analysis more challenging. Inspired by the analysis in [24], we need to prove a weak form of submodularity [22, 25] for $\bar{f}$ (especially when $\lambda = 0$). To this end, we require the informativeness criterion I_ℓ to satisfy certain structural assumptions. First, we define the restricted strongly concavity, and restricted smoothness notions of a function that are used in our analysis.

Definition 1 (Restricted Strong Concavity, Restricted Smoothness [26])**.** *A function $\mathcal{L} : \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|} \to \mathbb{R}$ is said to be restricted strong concave with parameter m_Ω, and restricted smooth with parameter M_Ω on a domain $\Omega \subset \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|} \times \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ if for all $(x, y) \in \Omega$:*

$$-\frac{m_\Omega}{2} \|y - x\|_2^2 \;\geq\; \mathcal{L}(y) - \mathcal{L}(x) - \langle \nabla \mathcal{L}(x), y - x \rangle \;\geq\; -\frac{M_\Omega}{2} \|y - x\|_2^2 .$$

For any integer k, we define the sets: $\Omega_k = \{(x, y) : \|x\|_0 \leq k, \|y\|_0 \leq k, \|x - y\|_0 \leq k, x, y \in \mathcal{R}\}$, and $\tilde{\Omega}_k := \{(x, y) : \|x\|_0 \leq k, \|y\|_0 \leq k, \|x - y\|_0 \leq 1, x, y \in \mathcal{R}\}$. Let $m_k := m_{\Omega_k}$, and $M_k := M_{\Omega_k}$ (similarly we define $\tilde{m}_k$ and $\tilde{M}_k$).

When there is no $R \in \mathcal{R}$ constraint in (2), the following assumption on the informativeness criterion is sufficient to prove the weak submodularity of $\bar{f}$ [24]:

Assumption 1. *I_ℓ is m_{2B}-restricted strongly concave, and M_{2B}-restricted smooth on Ω_{2B}.*

However, due to additional $R \in \mathcal{R}$ constraint, for any given $\mathcal{Z} \subseteq \mathcal{S}$, we need to ensure that (i) the components of the gradient $\nabla I_\ell(R^{(\mathcal{Z})})$ of the informativeness criterion at the optimal reward $R^{(\mathcal{Z})}$ are bounded, and (ii) the components of the optimal reward $R^{(\mathcal{Z})}$ outside $\mathcal{Z}$ do not lie in the boundary of $\mathcal{R}$. These requirements are formally captured in the following assumption:

Assumption 2. *For any $x \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$, and $\mathcal{Z} \subseteq \mathcal{S}$, $x_{\mathcal{Z}}$ is defined as $x_{\mathcal{Z}}(s, a) = x(s, a), \forall s \in \mathcal{Z}, a \in \mathcal{A}$, and $x_{\mathcal{Z}}(s, a) = 0$ otherwise. The vector $e_j \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ is defined as $e_j(j, a) = 1, \forall a \in \mathcal{A}$, and $e_j(s, a) = 0$ otherwise. The informativeness criterion I_ℓ satisfies the following: (i) $\left\|\nabla I_\ell(R^{(\mathcal{Z})})_{\mathcal{Z}}\right\|_2 \leq d_{\max}^{\mathrm{opt}}, \forall \mathcal{Z} \subseteq \mathcal{S}$, and $d_{\min}^{\mathrm{non}} \leq \left|\nabla I_\ell(R^{(\mathcal{Z})})(s, a)\right| \leq d_{\max}^{\mathrm{non}}, \forall s \in \mathcal{S} \setminus \mathcal{Z}, \mathcal{Z} \subseteq \mathcal{S}, a \in \mathcal{A}$, and (ii) $\exists \kappa \leq 1$ such that $\forall \mathcal{Z} \subseteq \mathcal{S}, j \in \mathcal{S} \setminus \mathcal{Z} : R^{(\mathcal{Z})} \pm \kappa \cdot \frac{d_{\max}^{\mathrm{non}}}{\tilde{M}_{|\mathcal{Z}|+1}} \cdot e_j \in \mathcal{R}$.*

By using the Assumption 1 and 2, we prove prove the weak submodularity of $\bar{f}$. Then, by applying Theorem 3 from [24], we obtain the following theorem, which holds even when $\lambda = 0$:

Theorem 1. *Let the informativeness criterion I_ℓ satisfies the Assumption 1 and 2. Then, we have $\bar{f}_B^{\text{Greedy}} \geq (1 - e^{-\gamma})\, \bar{f}_B^{\text{OPT}}$, where $\gamma = \frac{\kappa \cdot m_{2B}}{M_{2B}} \cdot \frac{(d_{\min}^{\text{non}})^2}{\left(d_{\max}^{\text{opt}}\right)^2 + \left(d_{\min}^{\text{non}}\right)^2}$.*

We provide a detailed proof of the theorem in Appendix of the supplementary material.

3.5 Extension to Large State Spaces using State Abstractions

This section presents an extension of our ExpRD framework that is scalable to large state spaces by leveraging the techniques from state abstraction literature [27–29]. We use an abstraction $\phi : \mathcal{S} \to \mathcal{X}_\phi$, which is a mapping from high-dimensional state-space $\mathcal{S}$ to a low-dimensional latent space $\mathcal{X}_\phi$. Let $\phi^{-1}(x) := \{s \in \mathcal{S} : \phi(s) = x\}, \forall x \in \mathcal{X}_\phi$. We propose the following pipeline:

1. By using $\overline{M}$ and ϕ, we construct an abstract MDP $\overline{M}_\phi = \left(\mathcal{X}_\phi, \mathcal{A}, T_\phi, \gamma, P_0, \overline{R}_\phi\right)$ as follows, $\forall x, x' \in \mathcal{X}_\phi, a \in \mathcal{A}$: $T_\phi(x'|x,a) = \frac{1}{|\phi^{-1}(x)|} \sum_{s \in \phi^{-1}(x)} \sum_{s' \in \phi^{-1}(x')} T(s'|s,a)$, and $\overline{R}_\phi(x,a) = \frac{1}{|\phi^{-1}(x)|} \sum_{s \in \phi^{-1}(x)} \overline{R}(s,a)$. We compute the set of optimal policies $\overline{\Pi}_\phi^*$ for the MDP $\overline{M}_\phi$.
2. We run our ExpRD framework on $\overline{M}_\phi$ with $\Pi^\dagger = \overline{\Pi}_\phi^*$, and the resulting reward is denoted $\widehat{R}_\phi$.
3. We define the reward function $\widehat{R}$ on the state space $\mathcal{S}$ as follows: $\widehat{R}(s,a) = \widehat{R}_\phi(\phi(s), a)$.

By assuming some structural conditions on ϕ, we can show that any optimal policy induced by the above reward $\widehat{R}$ acts nearly optimal w.r.t. $\overline{R}$. This pipeline can be extended to continuous state space as well, similar to [29–31]. We provide more details in Appendix of the supplementary material.

4 Experimental Evaluation

In this section, we evaluate ExpRD on two environments: RoomsNavEnv (Section 4.1) and LineKeyNavEnv (Section 4.2). RoomsNavEnv 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 space, it provides a very rich and an intuitive problem setting to validate different reward design techniques, and variants of RoomsNavEnv have been used extensively in the literature [18, 19, 32–36]. LineKeyNavEnv 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 in this environment is represented as a point on a line segment. Given the large state space representation, it is computationally challenging to apply the reward design technique from Section 3.3 and we use the state abstraction-based extension of our framework from Section 3.5. This environment is inspired by variants of navigation tasks in the literature where an agent needs to perform sub-tasks [3, 37]. We give an overview of main results here, and provide a more detailed description of the setup, additional results, and implementation code in the supplementary material.

4.1 Evaluation on RoomsNavEnv

RoomsNavEnv (Figure 1). We represent the environment as an MDP with $\mathcal{S}$ states each corresponding to cells in the grid-world indicating the agent's current location (shown as "blue-circle"). Goal (shown as "green-star") is located at the top-right corner cell. The agent can take four actions given by $\mathcal{A} := \{$"up", "left", "down", "right"$\}$. An action takes the agent to the neighbouring cell represented by the direction of the action; however, if there is a wall (shown as "brown-segment"), the agent stays 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. In addition to these walls, there are a few terminal walls (shown as "thick-red-segment") that terminates the episode—at the bottom-left corner cell, "left" and "down" actions terminate; at the top-right corner cell, "right" action terminates. The agent gets a reward of $R_{\max}$ after it has navigated to the goal and then takes a "right" action (i.e., only one state-action pair has a reward); note that this action also terminates the episode. The reward is 0 for all other state-action pairs and there is a discount factor γ. This MDP has $|\mathcal{S}| = 49$ and $|\mathcal{A}| = 4$; we set $p_{\text{rand}} = 0.1$, $R_{\max} = 10$, and $\gamma = 0.95$ in our evaluation.

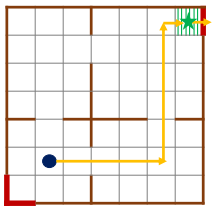
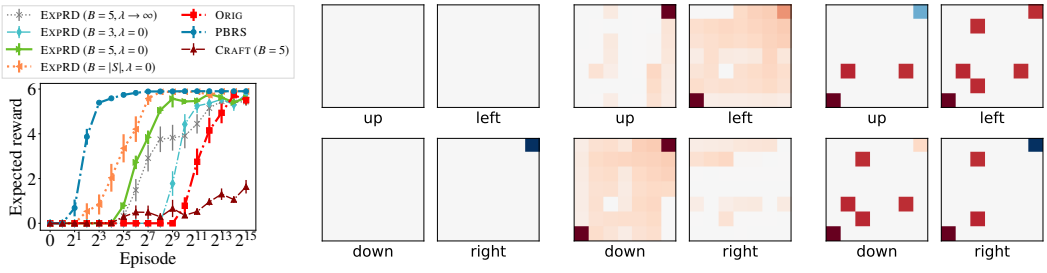


Figure 1: RoomsNavEnv



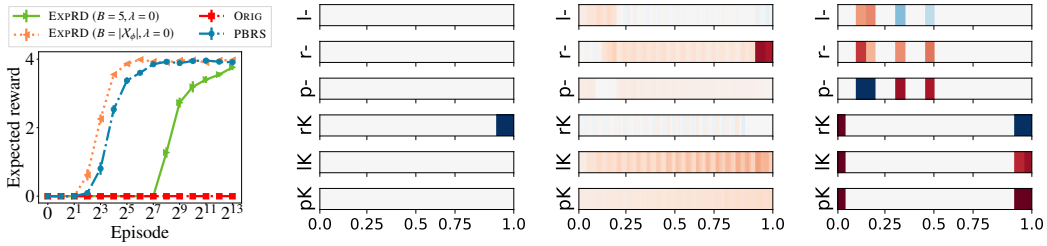
(a) Convergence (b) $\widehat{R}_{\text{ORIG}}$ (c) $\widehat{R}_{\text{PBRS}}$ (d) $\widehat{R}_{\text{ExpRD}(B=5,\lambda=0)}$

Figure 2: Results for RoomsNavEnv. **(a)** shows convergence in performance of the agent w.r.t. training episodes. Here, performance is measured as the expected reward per episode computed using $\overline{R}$; note that the x-axis is exponential in scale. **(b-d)** visualize the designed reward functions $\widehat{R}_{\text{ORIG}}$, $\widehat{R}_{\text{PBRS}}$, and $\widehat{R}_{\text{ExpRD}(B=5,\lambda=0)}$. These plots illustrate reward values for all combinations of $\mathcal{S} \times \mathcal{A}$ shown as four 7×7 grids corresponding to different actions. Blue color represents positive reward, red color represents negative reward, and the magnitude of the reward is indicated by color intensity. As an example, consider "right" action grid for $\widehat{R}_{\text{ORIG}}$ in **(b)** – the dark blue color in the corner indicates the goal. To increase the color contrast, we clipped rewards in the range $[-4, +4]$ for this visualization even though the designed rewards are in the range $[-10, +10]$. See Section 4.1 for details.

Techniques evaluated. We consider the following baselines: (i) $\widehat{R}_{\text{ORIG}} := \overline{R}$, which simply represents default reward function, (ii) $\widehat{R}_{\text{PBRS}}$ obtained via the PBRS technique (see Section 2), and (iii) $\widehat{R}_{\text{CRAFT}}$, which we designed manually (see Section 2). To design $\widehat{R}_{\text{CRAFT}}$, we first hand-crafted a set function D that assigns scores to the states in the MDP, e.g., the scores are higher for the four entry points in the rooms. In general, one could learn such D automatically using the techniques from [18–21]—see full details about D in the supplementary. Then, for a fixed budget B, we pick the top B states according to the scoring by D and assign a reward of $+1$ for optimal actions and -1 for others. For the evaluation in the main paper, we use $B = 5$ and denote the function as $\widehat{R}_{\text{CRAFT}}(B = 5)$. Note that apart from B states, $\widehat{R}_{\text{CRAFT}}(B = 5)$ also has a reward assigned for the goal state taken from $\overline{R}$.

The reward functions $\widehat{R}_{\text{ExpRD}}$ designed by our ExpRD framework are parameterized by budget B and hyperparameter λ. For λ, we consider two extreme settings: (a) $\lambda = 0$ where the problem (P3) reduces to (P2) corresponding to fully automated reward design, and (b) $\lambda \to \infty$ where the problem (P3) reduces to (P1) corresponding to the reward design with subgoals pre-selected by the function D. We use the same function D that we used for $\widehat{R}_{\text{CRAFT}}$ above. For budget B, we consider values from $\{3, 5, |S|\}$. In particular, we evaluate the following reward functions: $\widehat{R}_{\text{ExpRD}(B=5,\lambda\to\infty)}$, $\widehat{R}_{\text{ExpRD}(B=3,\lambda=0)}$, $\widehat{R}_{\text{ExpRD}(B=5,\lambda=0)}$, and $\widehat{R}_{\text{ExpRD}(B=|S|,\lambda=0)}$. For the evaluation in this section, we use the following parameter choices for ExpRD: $\mathcal{H} = \{1, 2, 4, 8, 16, 32\}$, ℓ is the negated hinge loss ℓ_{hg}, and $\Pi^{\dagger}$ contains only one policy from $\overline{\Pi}^{*}$.

Results. We use standard Q-learning method for the agent with a learning rate 0.5 and exploration factor 0.1 [7]. During training, the agent receives rewards based on $\widehat{R}$, however, is evaluated based on $\overline{R}$. A training episode ends when the maximum steps (set to 50) is reached or an agent's action terminates the episode. All the results are reported as average over 20 runs and convergence plots show mean with standard error bars. The convergence behavior in Figure 2a demonstrates the effectiveness of the reward functions designed by our ExpRD framework. Note that $\widehat{R}_{\text{CRAFT}}(B = 5)$ leads to sub-optimal behavior due to "reward bugs" (see Section 2), whereas $\widehat{R}_{\text{ExpRD}(B=5,\lambda\to\infty)}$ fixes this issue using the same set of subgoals. ExpRD leads to good performance even without domain knowledge (i.e., when $\lambda = 0$), e.g., the performance corresponding to $\widehat{R}_{\text{ExpRD}(B=3,\lambda=0)}$ is comparable to that of $\widehat{R}_{\text{ExpRD}(B=5,\lambda\to\infty)}$. The visualizations of $\widehat{R}_{\text{ORIG}}$, $\widehat{R}_{\text{PBRS}}$, and $\widehat{R}_{\text{ExpRD}(B=5,\lambda=0)}$ in Figures 2b, 2c, and 2d highlight the trade-offs in terms of sparseness and interpretability of the reward functions. The reward function $\widehat{R}_{\text{ExpRD}(B=5,\lambda=0)}$ designed by our ExpRD framework provides a good balance in terms of convergence performance while maintaining high sparseness. Additional visualizations and results when varying B and λ are provided in the supplementary material.



(a) Convergence (b) $\widehat{R}_{\text{ORIG}}$ (c) $\widehat{R}_{\text{PBRS}}$ (d) $\widehat{R}_{\text{EXPRD}(B=5,\lambda=0)}$

Figure 4: Results for LINEKEYNAVENV. **(a)** shows convergence in performance of the agent w.r.t. training episodes. Here, performance is measured as the expected reward per episode computed using $\overline{R}$. **(b-d)** visualize the designed reward functions $\widehat{R}_{\text{ORIG}}$, $\widehat{R}_{\text{PBRS}}$, and $\widehat{R}_{\text{EXPRD}(\lambda=0,B=5)}$. These plots illustrate reward values for all combination of triplets, i.e., agent's location on the segment $[0.0, 1.0]$ (shown as horizontal bar), agent's status whether it has acquired key or not (indicated as 'K' or '-'), and three actions (indicated as 'l' for "left", 'r' for "right", 'p' for "pick"). We use a color representation similar to Figure 2, and we clipped rewards in the range $[-3, +3]$ to increase the color contrast for this visualization. As an example, consider 'rK' bar for $\widehat{R}_{\text{ORIG}}$ in **(b)** – the dark blue color on the segment $[0.9, 1]$ indicate the locations with goal. See Section 4.2 for details.

4.2 Evaluation on LINEKEYNAVENV

LINEKEYNAVENV (Figure 3). 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 is a point x in $[0, 1]$) and a binary flag whether the agent has acquired a key (shown as "cyan-bolt"). Goal (shown as "green-star") is available in locations on the segment $[0.9, 1]$, and the key is available in locations on the segment $[0.1, 0.2]$. The agent can take three actions given by $\mathcal{A} := \{$"left", "right", , "pick"$\}$. "pick" action does not change the agent's location, however, when executed in locations with availability of the key, the agent acquires the key; if agent already had a key, the action does not affect the status. A move action of "left" or "right" takes the agent from the current location in the direction of move with the dynamics of the final location captured by two hyperparameters $(\Delta_{a,1}, \Delta_{a,2})$; for instance, with current location x and action "left", the new location x$'$ is sampled uniformly among locations from $(\text{x} - \Delta_{a,1} - \Delta_{a,2})$ to $(\text{x} - \Delta_{a,1} + \Delta_{a,2})$. Similar to ROOMSNAVENV, the agent's move action is not applied if the new location crosses the wall, and there is p_{rand} probability of a random action. The agent gets a reward of R_{max} after it has navigated to the goal locations after acquiring the key and then takes a "right" action; note that this action also terminates the episode. The reward is 0 elsewhere and there is a discount factor γ. We set $p_{\text{rand}} = 0.1$, $R_{\text{max}} = 10$, $\gamma = 0.95$, $\Delta_{a,1} = 0.075$ and $\Delta_{a,2} = 0.01$.

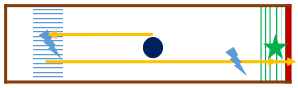


Figure 3: LINEKEYNAVENV

Techniques evaluated. The baseline $\widehat{R}_{\text{ORIG}} := \overline{R}$ represents the default reward function. We evaluate the variants of $\widehat{R}_{\text{PBRS}}$ and $\widehat{R}_{\text{EXPRD}}$ using an abstraction. For a given hyperparameter $\alpha \in (0, 1)$, the set of possible locations X are obtained by α-level discretization of the line segment from 0.0 to 1.0, leading to a $1/\alpha$ set of locations. For the abstraction ϕ associated with this discretization [38], the abstract MDP $\overline{M}_\phi$ (see Section 3.5) has $|\mathcal{X}_\phi| = 2/\alpha$ and $|\mathcal{A}| = 3$. We use $\alpha = 0.05$. We compute the optimal state value function in the abstract MDP $\overline{M}_\phi$, lift it to the original state space via ϕ, and use the lifted value function as the potential to design $\widehat{R}_{\text{PBRS}}$ [30]. We follow the pipeline in Section 3.5 to design $\widehat{R}_{\text{EXPRD}}$ – in the subroutine, we run EXPRD on $\overline{M}_\phi$ for a budget $B = 5$ and a full budget $B = |\mathcal{X}_\phi|$; we set $\lambda = 0$. For other parameters ($\mathcal{H}$, ℓ, and $\Pi^\dagger$), we use the same choices as in Section 4.1.

Results. The agent uses Q-learning method in the original MDP $\overline{M}$ by using a fine-grained discretization of the state space; rest of the method's parameters are same as in Section 4.1. All the results are reported as average over 20 runs and convergence plots show mean with standard error bars. Figure 4a demonstrates that all three designed reward functions—$\widehat{R}_{\text{PBRS}}$, $\widehat{R}_{\text{EXPRD}(B=5,\lambda=0)}$, $\widehat{R}_{\text{EXPRD}(B=|\mathcal{X}_\phi|,\lambda=0)}$—substantially improves the convergence, whereas the agent is not able to learn under $\widehat{R}_{\text{ORIG}}$. Based on the visualizations in Figures 4b, 4c, and 4d, $\widehat{R}_{\text{EXPRD}(B=5,\lambda=0)}$ provides a good balance between convergence and sparseness. Interestingly, $\widehat{R}_{\text{EXPRD}(B=5,\lambda=0)}$ assigned a high positive reward for the "pick" action when the agent is in the locations with key (see 'p-' bar in Figure 4d).

5 Related Work

Potential-based reward shaping. Introduced in the seminal work of [3], potential-based reward shaping is one of the most well-studied reward design technique (see [8, 32, 33, 35, 36, 39, 40, 40–43]). As we discussed in Section 2, the shaped reward function $\widehat{R}_{\text{PBRS}}$ is obtained by modifying $\overline{R}$ using a state-dependent potential function. The technique preserves a strong invariance property: a policy π is optimal under $\widehat{R}_{\text{PBRS}}$ *iff* it is optimal under $\overline{R}$. Furthermore, when using the optimal value-function $\overline{V}^*_\infty$ under $\overline{R}$ as the potential function, the shaped rewards achieve the maximum possible informativeness as per the notion we use in ExpRD. To balance informativeness and sparseness, our framework ExpRD can be seen as a relaxation of the potential-based shaping in the following ways: (i) ExpRD provides a guarantee on preserving a weaker invariance property whereby an optimal policy under $\widehat{R}_{\text{ExpRD}}$ is also optimal under $\overline{R}$; and (ii) ExpRD finds $\widehat{R}_{\text{ExpRD}}$ that maximizes informativeness under hard constraints of preserving this weaker policy-invariant property and a given spareness-level.

Optimization-based techniques for reward design. Beyond potential-based shaping, we can formulate reward design as an optimization problem [12–16]. In particular, optimization-based techniques for reward design are popularly used in data poisoning attacks where an attacker's goal is to minimally perturb the original reward function to force the agent into executing a target policy chosen by the attacker [14–16]. Our ExpRD framework builds on the optimization framework of [14–16]. The key novelty of ExpRD is that we optimize for informativeness of the reward function under a sparseness constraint, which makes our problem formulation much more challenging.

Agent-driven reward design. An important categorization of reward design techniques is based on who is designing the rewards and what domain knowledge is available. Agent-driven reward design techniques involve a reinforcement learning method where an agent self-designs its own rewards during the training process, with the objective of improving the exploration and speeding up the convergence [6, 44–48]. These agent-driven techniques use a wide-variety of ideas such as designing intrinsic rewards based on exploration bonus [44, 45, 49], designing rewards using some additional domain knowledge [46], and using credit assignment to create intermediate rewards [6, 47].

Expert-driven reward design. In contrast to agent-driven techniques, we have expert-driven reward design techniques where an expert/teacher with full domain knowledge can design a reward function for the agent [1, 3, 12–16, 36, 43]. Our ExpRD framework falls into the category of teacher-driven reward design. The above-mentioned techniques of potential-based reward shaping and optimization-based techniques can be seen as expert-driven reward design techniques; however, the distinction between expert-driven and agent-driven techniques can be blurry at times when one uses an expert-driven technique with minimal domain knowledge (e.g., when using approximate potentials [3]).

Reward automatas, landmark-based rewards, and subgoal discovery. Our ExpRD framework is also connected to techniques that specify rewards using higher-level abstract representations of the environment including symbolic automata and landmarks [32, 35, 36, 50–52]. In recent works [36, 50–52], potential-based reward shaping technique has been used with automata-based rewards to design interpretable and informative rewards. While similar in the overall objective, our work is technically quite different and our proposed optimization framework to reward design can be seen as complementary to these works. Another relevant line of work focuses on automatic discovery of subgoals in the environment [18–21] – these works are complementary and useful as subroutines in our framework by providing a prior knowledge about which states are important for assigning rewards.

6 Conclusions

We developed a novel optimization framework, ExpRD, to design explicable reward functions in which we can appropriately balance informativeness and sparseness in the reward design process. As part of the framework, we introduced a new criterion capturing informativeness of reward functions that is of independent interest. The mathematical analysis of ExpRD shows connections of our framework to the popular reward-design techniques, and provides theoretical underpinnings of expert-driven explicable reward design. We also provided a practical extension to apply our framework in environments with large state spaces via state abstractions. To make our framework more scalable, we plan to investigate alternate formulations of the reward design problem that avoids enumerating all the constraints explicitly (see Section 3). There are several promising directions for future work, including but not limited to the following: (a) using a combination of our optimization-based reward design technique with automata-driven rewards, (b) extending our framework for agent-driven reward design, and (c) investigating the usage of our informativeness criterion for discovering subgoals.

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

[9] Eleanor O'Rourke, Kyla Haimovitz, Christy Ballweber, Carol S. Dweck, and Zoran Popovic. Brain Points: A Growth Mindset Incentive Structure Boosts Persistence in an Educational Game. In *CHI*, pages 3339–3348, 2014.

[10] VirtaMed. Virtamed: Simulators for Medical Training and Education. `https://www.virtamed.com/en/`.

[11] Virtual Driver Interactive. `https://www.driverinteractive.com/`.

[12] Haoqi Zhang and David C. Parkes. Value-Based Policy Teaching with Active Indirect Elicitation. In *AAAI*, pages 208–214, 2008.

[13] Haoqi Zhang, David C. Parkes, and Yiling Chen. Policy Teaching through Reward Function Learning. In *EC*, pages 295–304, 2009.

[14] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy Poisoning in Batch Reinforcement Learning and Control. In *NeurIPS*, pages 14543–14553, 2019.

[15] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. In *ICML*, volume 119, pages 7974–7984, 2020.

[16] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy Teaching in Reinforcement Learning via Environment Poisoning Attacks. *CoRR*, abs/2011.10824, 2020.

[17] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

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

[19] 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.

[20] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. In *ICML*, volume 80, pages 1514–1523, 2018.

[21] Sujoy Paul, Jeroen van Baar, and Amit K. Roy-Chowdhury. Learning from Trajectories via Subgoal Discovery. In *NeurIPS*, pages 8409–8419, 2019.

[22] Andreas Krause and Daniel Golovin. Submodular Function Maximization. *Tractability*, 3:71–104, 2014.

[23] Michael J. Kearns, Yishay Mansour, and Andrew Y. Ng. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *Machine Learning*, 49(2-3):193–208, 2002.

[24] Ethan R Elenberg, Rajiv Khanna, Alexandros G Dimakis, and Sahand Negahban. Restricted Strong Convexity Implies Weak Submodularity. *Annals of Statistics*, 46(6B):3539–3568, 2018.

[25] Abhimanyu Das and David Kempe. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, pages 1057–1064, 2011.

[26] Sahand N Negahban, Pradeep Ravikumar, Martin J Wainwright, Bin Yu, et al. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical science*, 27(4):538–557, 2012.

[27] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[28] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4:5, 2006.

[29] David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

[30] Bhaskara Marthi. Automatic shaping and decomposition of reward functions. In *ICML*, pages 601–608, 2007.

[31] Parameswaran Kamalaruban, Rati Devidze, Volkan Cevher, and Adish Singla. Environment Shaping in Reinforcement Learning using State Abstraction. *CoRR*, abs/2006.13160, 2020.

[32] 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.

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

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

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

[36] 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.

[37] 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.

[38] John Burden and Daniel Kudenko. Uniform state abstraction for reinforcement learning. *arXiv preprint arXiv:2004.02919*, 2020.

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

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

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

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

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

[44] Andrew G. Barto. Intrinsic Motivation and Reinforcement Learning. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 17–47. 2013.

[45] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *NeurIPS*, pages 3675–3683, 2016.

[46] 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.

[47] Johan Ferret, Raphaël Marinier, Matthieu Geist, and Olivier Pietquin. Self-Attentional Credit Assignment for Transfer in Reinforcement Learning. In *IJCAI*, pages 2655–2661, 2020.

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

[49] Xuezhou Zhang, Yuzhe Ma, and Adish Singla. Task-Agnostic Exploration in Reinforcement Learning. In *NeurIPS*, 2020.

[50] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. Decision-Making with Non-Markovian Rewards: From LTL to Automata-based Reward Shaping. In *Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, pages 279–283, 2017.

[51] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A Composable Specification Language for Reinforcement Learning Tasks. In *NeurIPS*, pages 13021–13030, 2019.

[52] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *CoRR*, abs/2010.03950, 2020.

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 exactly according to the contributions listed at the end of the introduction section.
   (b) Did you describe the limitations of your work? [Yes] We discussed the scalability related limitations of our ExpRD framework in Sections 3.3 and 6. We also outlined a future plan to address these limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] As stated in Section 6, this work primarily presents the theoretical underpinnings of reward design in reinforcement learning. As such in the present form there are no direct negative societal impacts of our work. However, given the importance of reward design in RL, one needs to be cautious in the practical applications of these techniques in future.
   (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] For example, Assumption 1 and 2 gather the structural assumptions that the informativeness criterion needs to satisfy.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs of all theoretical results are included in the Appendix of 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] Training details are presented in the Appendix of the supplementary material and are also present in the code.
   (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, as can be seen in Figures 2a and 4a.
   (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] The details are provided in the Appendix of 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]