Explicable Reward Design for Reinforcement Learning Agents

Rati Devidze\textsuperscript{1} Goran Radanovic\textsuperscript{1} Parameswaran Kamalaruban\textsuperscript{2} Adish Singla\textsuperscript{1}

rdevidze@mpi-sws.org gradanovic@mpi-sws.org kparameswaran@turing.ac.uk adishs@mpi-sws.org

\textsuperscript{1}Max Planck Institute for Software Systems (MPI-SWS), Saarbrucken, Germany
\textsuperscript{2}The Alan Turing Institute, London, UK

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 as a proxy for ease of interpretability of the rewards. 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 to consider is sparseness as a proxy for ease of interpretability of the rewards. There are several practical settings where sparseness and interpretability of rewards are important, as discussed next. The first motivating application is when
We denote a stochastic policy $\pi^Q$ and from a state to an action. For any policy $\pi$ a state to a probability distribution over actions, and a deterministic policy $\pi^{\ell}$ the $R$ represent the reward function by a vector $R : S \times S \times A \rightarrow [0,1]$. An environment is defined as a Markov Decision Process (MDP) $M := (S, A, T, \gamma, R)$, where the set of states and actions are denoted by $S$ and $A$ respectively. $T : S \times S \times A \rightarrow [0,1]$ 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, $\gamma$ is the discounting factor. The underlying reward function is given by $R : S \times A \rightarrow [-R_{\max}, R_{\max}]$, for some $R_{\max} > 0$. We interchangeably represent the reward function by a vector $R \in \mathbb{R}^{\mathbb{S} \cdot |A|}$, whose $(s,|A|+a)$-th entry is given by $R(s, a)$. We define the support of $R$ as $\text{supp}(R) := \{s : s \in S, R(s,a) \neq 0 \text{ for some } a \in A\}$, and the $l_0$-norm of $R$ as $\|R\|_0 := |\text{supp}(R)|$.

**Environment.** An environment is defined as a Markov Decision Process (MDP) $M := (S, A, T, \gamma, R)$, where the set of states and actions are denoted by $S$ and $A$ respectively. $T : S \times S \times A \rightarrow [0,1]$ 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, $\gamma$ is the discounting factor. The underlying reward function is given by $R : S \times A \rightarrow [-R_{\max}, R_{\max}]$, for some $R_{\max} > 0$. We interchangeably represent the reward function by a vector $R \in \mathbb{R}^{\mathbb{S} \cdot |A|}$, whose $(s,|A|+a)$-th entry is given by $R(s, a)$. We define the support of $R$ as $\text{supp}(R) := \{s : s \in S, R(s,a) \neq 0 \text{ for some } a \in A\}$, and the $l_0$-norm of $R$ as $\|R\|_0 := |\text{supp}(R)|$.

**Preliminaries and definitions.** We denote a stochastic policy $\pi : S \rightarrow \Delta(A)$ as a mapping from a state to a probability distribution over actions, and a deterministic policy $\pi : S \rightarrow A$ as a mapping from a state to an action. For any policy $\pi$, the state value function $V^\pi_\infty$ and the action value function $Q^\pi_\infty$ in the MDP $M$ are defined as follows respectively: $V^\pi_\infty(s) := \mathbb{E}_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, T, \pi$ and $Q^\pi_\infty(s, a) := \mathbb{E}_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, T, \pi$. 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 distinct policy $\pi^* : S \rightarrow \Delta(A)$ such that $Q^*_{\infty}(s, a) = V^*_{\infty}(s)$.
deterministic stationary policy \( \pi \) that achieves the optimal value function simultaneously for all \( s \in S \) \([7, 20]\), and we denote all such deterministic optimal policies by the set \( \Pi^* := \{ \pi : S \to A \operatorname{s.t.} V^*_{\infty}(s) = V^*_{\infty}(s), \forall s \in S \} \). From here onwards, we focus on deterministic policies unless stated otherwise. For any \( \pi \) and \( R \), we define the following quantities that capture the \( \infty \)-step (global) optimality gap and the \( 0 \)-step (myopic) optimality gap of action \( a \) at state \( s \), respectively:

\[
\delta_{\infty}^*(s, a) := Q^*_\infty(s, \pi(s)) - Q^*_\infty(s, a), \quad \text{and} \quad \delta_0^*(s, a) := Q^*_0(s, \pi(s)) - Q^*_0(s, a), \forall s \in S, a \in A,
\]

where \( Q^*_\infty(s, a) = R(s, a) \) is the \( 0 \)-step action value function of policy \( \pi \). The \( \delta_{\infty}^*(s, a) \) values are same for all \( \pi \in \Pi^* \), and we denote it by \( \delta_{\infty}(s, a) = V^*_{\infty}(s) - Q^*_\infty(s, a) \); however, this is not the case with \( \delta_0^*(s, a) \) values in general. For any state \( s \in \mathcal{S} \) and a set of policies \( \Pi \), we define \( \Pi_s := \{ a : a = \pi(s), \pi \in \Pi \} \). Then, we have that \( \delta_{\infty}(s, a) = 0, \forall s \in \mathcal{S}, a \in \Pi_s^* \).

**Explicable reward design.** Figure 1 presents an illustration of the explicable reward design problem that we formalize below. A task is specified as an MDP \( M \) with a given goal-based reward function \( \mathcal{R} \) where \( \mathcal{R} \) has non-zero rewards only on goal states \( \mathcal{G} \subseteq \mathcal{S} \), i.e., \( \mathcal{R}(s, a) = 0, \forall s \in \mathcal{S}\setminus \mathcal{G}, a \in A \). Many naturally occurring tasks (see Section 1 for motivating applications) are goal-based and challenging for learning an optimal policy when the state space \( \mathcal{S} \) is very large. In this paper, we study the following explicable reward design problem from an expert/teacher’s point of view: Given \( \mathcal{R} \) and the corresponding optimal policy set \( \Pi^* \) w.r.t. \( \mathcal{R} \) as the input, the teacher designs a new reward function \( \hat{\mathcal{R}} \) with criteria of informativeness and sparseness while guaranteeing an \( \infty \)-step requirement (these properties are formalized in Section 3). Informally, the \( \infty \)-step requirement is that any optimal policy learned using the new reward \( \hat{\mathcal{R}} \) belongs to the optimal policy set \( \hat{\Pi}^* \) induced by \( \hat{\mathcal{R}}^* \).

**Typical techniques for reward design and issues.** Given a set of important states (subgoals) in the environment, one could design a handcrafted reward function \( \hat{\mathcal{R}}_{\text{Craft}} \) by assigning non-zero reward values only to these states. Even though this simple approach produces a reward function with a specified sparsity level, it often fails to satisfy the \( \infty \)-step 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 \( V^*_\infty \) under \( \mathcal{R} \) as the potential function is defined as follows:

\[
\hat{\mathcal{R}}_{\text{PBRS}}(s, a) := \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, a) \cdot V^*_\infty(s') - V^*_\infty(s) .
\]  

The set of optimal policies \( \hat{\Pi}^* \) induced by \( \hat{\mathcal{R}}_{\text{PBRS}} \) is exactly equal to the set of optimal policies \( \hat{\Pi}^* \) induced by \( \mathcal{R} \) since \( \hat{\delta}_{\infty}^*(s, a) = \delta_{\infty}^*(s, a) \) for all \( \pi \in \Pi^* \) \([3]\). In addition, for any state \( s \in \mathcal{S} \), globally optimal actions \( \Pi_s^* \subseteq A \) under \( \mathcal{R} \) are also myopically optimal under \( \hat{\mathcal{R}}_{\text{PBRS}} \) since \( \hat{\delta}_{\infty}^*(s, a) = \delta_{\infty}^*(s, a) \) for all \( \pi \in \hat{\Pi}^* \) \([3, 8]\) – this leads to a dramatic speed-up in the learning process. However, the potential-based reward shaping produces dense reward function which is less interpretable (see Section 4).

\footnote{In the rest of the paper, the quantities defined corresponding to \( \mathcal{R} := \mathcal{R} \) are denoted by an overline, e.g., the optimal policy set by \( \overline{\Pi}^* \) and the \( \infty \)-step optimality gaps by \( \overline{\delta}_{\infty}^* \); the quantities defined corresponding to \( \hat{\mathcal{R}} := \hat{\mathcal{R}} \) are denoted by a widehat, e.g., the optimal policy set by \( \hat{\Pi}^* \).}
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 $\bar{R}$ and the corresponding optimal policy set $\bar{\Pi}$, we systematically develop a discrete optimization framework (EXPRD) to design an explicable reward function $\hat{R}$ (see Figure 1).

Spariness, informativeness, and invariance. The sparseness of the reward function $\hat{R}$ is captured by $\text{supp}(\hat{R})$. In Section 3.2, we formalize an informativeness criterion $I(\hat{R})$ of $\hat{R}$ that captures how hard/easy it is to learn an optimal behavior induced by $\hat{R}$. We explicitly enforce the invariance requirement (see Section 2) for the new reward $\hat{R}$ by choosing a set of candidate policies $\Pi^1 \subseteq \Pi^*$, and satisfying the following (Bellman-optimality) conditions:

$$Q^\pi_{\infty}(s,a) = \hat{R}(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) \cdot Q^\pi_{\infty}(s',\pi^\dagger(s')) \quad \forall \pi^\dagger \in \Pi^1 \quad (C.1)$$

$$Q^\pi_{\infty}(s,\pi^\dagger(s)) \geq Q^\pi_{\infty}(s,a) + \delta^\pi_{\infty}(s) \quad \forall \pi^\dagger \in \Pi^1 \subseteq \Pi^* \quad (C.2)$$

where $\delta^\pi_{\infty}(s) := \min_{a \in \mathcal{A}\setminus \Pi^\dagger} \delta^\pi_{\infty}(s,a)$, $\forall s \in S$. The above conditions guarantee that any optimal policy induced by $\hat{R}$ is also optimal under $\hat{R}$, i.e., $\Pi^1 \subseteq \hat{\Pi} \subseteq \Pi^*$. Here, the set $\Pi^1 \subseteq \Pi^*$ is used to reduce the number of constraints. Note that for the potential-based shaped reward $R_{PBRS}$, we have $\hat{\Pi} = \Pi^*$.

Maximizing informativeness for a given set of important states. When a domain expert provides us a set of important states (subgoals) in the environment [21–24], we want to use this set in a principled way to design a reward $\hat{R}$, while avoiding the “reward bugs” that can arise from hand-crafted rewards $\hat{R}_{CRAFT}$. To this end, for any given set of subgoals $Z \subseteq S \setminus G$, we optimize the informativeness criterion $I(R)$ while satisfying the invariance requirement:

$$g(Z) := \max_{R: \text{supp}(R) \subseteq Z \cup G} I(R)$$

subject to conditions (C.1) – (C.2) with $\hat{R}$ replaced by $R$ hold

$$|R(s,a)| \leq R_{\max}, \forall s \in S, a \in \mathcal{A}. \quad (P1)$$

Let $R(Z)$ denote the $R$ that maximizes $g(Z)$. Let $\mathcal{R} \subseteq \mathbb{R}^{[S] \setminus [A]}$ 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:

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

We can incorporate prior knowledge about the quality of subgoals using a set function $D : 2^S \rightarrow \mathbb{R}$ (we assume $D$ to be a submodular function [25]). Finally, the full EXPRD formulation is given by:

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

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

Note that the true action values $Q^\pi_{\infty}$ are used in the conditions (C.1) – (C.2) to obtain the terms $\delta^\pi_{\infty}(s,a)$, $\mathcal{A}\setminus \Pi^\dagger$, and $\Pi^1$. However, when we only have an approximate estimate of $Q^\pi_{\infty}$, we can adapt (C.1) – (C.2) appropriately with approximate versions of $\delta^\pi_{\infty}(s,a)$, $\mathcal{A}\setminus \Pi^\dagger$, and $\Pi^1$. 

4
3.2 Informativeness Criterion

Understanding the informativeness of a reward function is an important problem, and several works have investigated it [4, 5, 26–28]. Our goal is to define an informativeness criterion that is amenable to optimization techniques. As noted in Section 2, for any policy \( \pi \in \Pi^* \), 0-step and \( \infty \)-step optimality gaps induced by \( R_{\text{PBR}} \) are all equal to \( \infty \)-step optimality gaps induced by \( R \), i.e., \( \delta_0^R(s, a) = \delta_{\infty}^R(s, a) = R_{\infty}(s, a) \). 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 \( \infty \)-step optimality.

Inspired by the \( h \)-step optimality notions studied in [4, 26], we define the \( h \)-step action value function of any policy \( \pi \) as \( Q_h^R(s, a) = \mathbb{E} \left[ \sum_{t=0}^h \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a, T, \pi \right] \), and it satisfies the following recursive relationship: \( Q_h^R(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a) \cdot Q_{h-1}^R(s', \pi(s')) \).

Let \( \mathcal{H} \) be a set of horizons for which we want to maximize informativeness. For any policy \( \pi \) and reward function \( R \), we define the following quantity that captures the optimality gaps induced by \( \hat{R}_{\text{PBR}} \) given in (1) is equal to the \( \infty \)-step optimality gap induced by \( R \):

**Proposition 1.** The goal-based reward function \( R \), and the potential-based shaped reward function \( \hat{R}_{\text{PBR}} \) given in (1) satisfy the following:

\[
\delta_h^R(s, a) = \delta_{\infty}(s, a), \forall s \in S, a \in A, \pi \in \Pi^*, h \in \mathcal{H}.
\]

Let \( \ell : \mathbb{R} \rightarrow \mathbb{R} \) 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:

\[
I_\ell(R) := \sum_{\pi \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s,a \in A} \sum_{\pi \in \Pi^*} \ell(\delta_h^{\pi}(s, a)).
\]

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

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 \( \mathcal{Z} \subseteq S \setminus \mathcal{G} \):

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

Then, inspired by the Forward Stepwise Selection method from [29], 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 \( |S| \cdot |A| \) optimization variables and \( \mathcal{O}(|S| \cdot |A| \cdot |\Pi| \cdot |\mathcal{H}|) \) constraints.

**Algorithm 1** Iterative Greedy Algorithm for EXPRD

1: **Input:** MDP \( \mathcal{M} := (S, A, T, \gamma, R) \), \( \delta_{\infty}(s, a) \) values, sets \( \Pi^*, \Pi^\dagger, \mathcal{G}, \mathcal{H} \), sparsity budget \( B \)
2: **Initialize:** \( \mathcal{Z}_0 \leftarrow \emptyset \)
3: for \( k = 1, 2, \ldots, B \) do
4: \( z_k \leftarrow \arg \max_{z \in S \setminus \mathcal{Z}_{k-1}} g(\mathcal{Z}_{k-1} \cup \{z\}) + \lambda \cdot D(\mathcal{Z}_{k-1} \cup \{z\}) - g(\mathcal{Z}_{k-1}) - \lambda \cdot D(\mathcal{Z}_{k-1} \cup \mathcal{G}) \)
5: \( \mathcal{Z}_k \leftarrow \mathcal{Z}_{k-1} \cup \{z_k\} \)
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 the
Appendix. For some \( \mu \geq 0 \), let \( I^\text{reg}_\ell (R) := I_\ell (R) - \mu \| R \|_2^2 \) be the regulated informativeness criterion. We define a normalized set function \( f : 2^S \to \mathbb{R} \) as follows:

\[
f(Z) = \max_{R : \text{supp}(R) \subseteq Z \cup G, R \in \mathcal{R}} \left( I^\text{reg}_\ell (R) - I^\text{reg}_\ell (R^{(0)}) \right) + \lambda \cdot (D(Z \cup G) - D(G)),
\]

where \( R^{(0)} = \arg \max_{R : \text{supp}(R) \subseteq G, R \in \mathcal{R}} I^\text{reg}_\ell (R) \). Note that the regularized variant (\( I_\ell \) replaced by \( I^\text{reg}_\ell \)) of the optimization problem (P3) is equivalent to \( \max_{Z : Z \subseteq S \setminus \{G \} \subseteq \mathcal{R}} f(Z) \). For a given sparsity budget \( B \), let \( Z_B^{\text{Greedy}} \) be the set selected by our Algorithm 1 and \( Z_B^{\text{OPT}} \) be the optimal set that maximizes the regularized variant of problem (P3). The corresponding \( f \) values of these sets are denoted by \( f_{B}^{\text{Greedy}} \) and \( f_{B}^{\text{OPT}} \) 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 [29] 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 [29], we need to prove a weak form of submodularity [25, 30] for \( f \) (since \( D \) is already a submodular function, we need to prove this for the case when \( \lambda = 0 \)). To this end, we require the regularized informativeness criterion \( I^\text{reg}_\ell \) 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 [31]). A function \( \mathcal{L} : \mathbb{R}^{|S|+|A|} \to \mathbb{R} \) is said to be restricted strongly concave with parameter \( m_\Omega \) and restricted smooth with parameter \( M_\Omega \) on a domain \( \Omega \subset \mathbb{R}^{|S|+|A|} \times \mathbb{R}^{|S|+|A|} \) 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 following two sets: \( \Omega_k := \{(x, y) : \|x\|_0 \leq k, \|y\|_0 \leq k, \|x - y\|_0 \leq k, x, y \in \mathcal{R}\} \), and \( \Omega_k := \{(x, y) : \|x\|_0 \leq k, \|y\|_0 \leq k, \|x - y\|_0 \leq 1, x, y \in \mathcal{R}\} \). Let \( m_k := m_{\Omega_k} \) and \( M_k := M_{\Omega_k} \) (similarly we define \( m_{\text{non}} \) and \( M_{\text{non}} \)).

When there is no \( R \in \mathcal{R} \) constraint in (2), the following assumption on the regularized informativeness criterion is sufficient to prove the weak submodularity of \( f \) [29]:

**Assumption 1.** The regularized informativeness criterion \( I^\text{reg}_\ell \) is restricted strongly concave and \( M_{2B+|G|} \cdot \text{restricted smooth on } \Omega_{2B+|G|} \).

However, due to the additional \( R \in \mathcal{R} \) constraint, we need to enforce further requirements on \( I^\text{reg}_\ell \) formally captured in Assumption 2 provided in the Appendix; here, we discuss these requirements informally. Let \( Z \) be any set such that \( Z \subseteq S \setminus \mathcal{G} \), and \( \nabla I^\text{reg}_\ell (R(Z)) \) be the gradient of the regularized informativeness criterion at the optimal reward \( R(Z) \). Then, we need to ensure the following: (i) the \( \ell_2 \)-norm of the projection of \( \nabla I^\text{reg}_\ell (R(Z)) \) on \( (Z \cup \mathcal{G}) \) is upper-bounded, captured by \( d_{\text{max}} \); (ii) the \( \ell_2 \)-norm of the projection of \( \nabla I^\text{reg}_\ell (R(Z)) \) on any \( j \in S \setminus (Z \cup \mathcal{G}) \) is lower-bounded, captured by \( d_{\text{non}} \); and (iii) the components of the optimal reward \( R(Z) \) outside \( Z \cup \mathcal{G} \) do not lie in the boundary of \( \mathcal{R} \), captured by \( \kappa \). Then, by using Assumption 1 and Assumption 2 (see Appendix), we prove the weak submodularity of \( f \). Finally, by applying Theorem 3 from [29], we obtain the following theorem:

**Theorem 1.** Let \( I^\text{reg}_\ell \) satisfies Assumption 1 and Assumption 2 requirements. Then, we have \( f_{B}^{\text{Greedy}} \geq (1 - e^{-\gamma}) f_{B}^{\text{OPT}} \), where \( \gamma = \frac{\kappa \cdot m_{2B+|G|}}{M_{2B+|G|}} \cdot \frac{(d_{\text{max}}^2)}{(d_{\text{non}})} \).

We provide Assumption 2 and a detailed proof of the theorem in the Appendix.

### 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 [32–34]. We use an abstraction \( \phi : S \to X_\phi \), which is a mapping from high-dimensional state space \( S \) to a low-dimensional latent space \( X_\phi \). Let \( \phi^{-1}(x) := \{s \in S : \phi(s) = x\} \), \( \forall x \in X_\phi \), and \( \mathcal{M} := (S, \mathcal{A}, T, \gamma, R) \). We propose the following pipeline:

1. By using \( \overline{M} \) and \( \phi \), we construct an abstract MDP \( \overline{M}_\phi = (X_\phi, \mathcal{A}, T_\phi, \gamma, \overline{R}_\phi) \) as follows,

\[
\forall x, x' \in X_\phi, a \in \mathcal{A}: T_\phi(x', x, a) = \frac{1}{|\phi^{-1}(x)|} \sum_{s \in \phi^{-1}(x)} \sum_{s' \in \phi^{-1}(x')} T(s', a), \text{ and } \overline{R}_\phi(x, a) = \frac{1}{|\phi^{-1}(x)|} \sum_{s \in \phi^{-1}(x)} R(s, a). \]

We compute the set of optimal policies \( \overline{\Pi}_\phi \) for the MDP \( \overline{M}_\phi \).

6
2. We run our EXPRD framework on $\hat{M}_\phi$ with $\Pi^\dagger = \Pi^*_\phi$, and the resulting reward is denoted $\hat{R}_\phi$.

3. We define the reward function $\hat{R}$ on the state space $\mathcal{S}$ as follows: $\hat{R}(s, a) = \hat{R}_\phi(\phi(s), a)$.

By assuming certain structural conditions on $\phi$ formalized in the Appendix, we can show that any optimal policy induced by the above reward $\hat{R}$ acts nearly optimal w.r.t. $\hat{R}$. This pipeline can be extended to continuous state space as well, similar to [34–36]. We provide more details in the Appendix.

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 [14, 21, 22, 37–40]. 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, 41]. We give an overview of main results here, and provide a more detailed description of the setup and additional results in the Appendix.

4.1 Evaluation on ROOMSNAVENV

ROOMSNAVENV (Figure 2). 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 $p_{\text{rand}}$ 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 $R_\text{max}$ 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 $\gamma$. This MDP has $|\mathcal{S}| = 49$ and $|\mathcal{A}| = 4$; we set $p_{\text{rand}} = 0.1$, $R_\text{max} = 10$, and $\gamma = 0.95$ in our evaluation.

Techniques evaluated. We consider the following baselines: (i) $\hat{R}_\text{ORIG} := \hat{R}$, which simply represents default reward function, (ii) $\hat{R}_\text{PBRS}$ obtained via the PBRS technique with the optimal value function $V^\pi_\infty$ w.r.t. $\hat{R}$ (see Section 2), (iii) $\hat{R}_\text{RAFT}$ that we design manually (see Section 2 and description below), and (iv) $\hat{R}_\text{PBRS-CRAFT}(B=5)$ obtained via the PBRS technique with the optimal value function w.r.t. $\hat{R}_\text{RAFT}$ instead of $V^\pi_\infty$ [42]. To design $\hat{R}_\text{RAFT}$, 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 [21–24]—see full details about $D$ in the Appendix. 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, we use $B = 5$ and denote the function as $\hat{R}_\text{RAFT}(B=5)$. Note that apart from $B$ states, $\hat{R}_\text{RAFT}(B=5)$ also has a reward assigned for the goal state taken from $\hat{R}$.

The reward shaping method in [42] is based on the PBRS technique and leads to dense reward functions. However, their method is more practical as it does not require solving the original task w.r.t. $\hat{R}$.
The reward functions $\hat{R}_{\text{EXP}}$ designed by our EXPRD framework are parameterized by budget $B$ and hyperparameter $\lambda$. For $\lambda$, we consider two extreme settings: (a) $\lambda = 0$ where the problem (P3) reduces to (P2), 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 $\hat{R}_{\text{RAFT}}$ above. For budget $B$, we consider values from $\{3, 5, |S|\}$. In particular, we evaluate the following reward functions: $\hat{R}_{\text{EXP}}(B=5, \lambda \to \infty)$, $\hat{R}_{\text{EXP}}(B=3, \lambda=0)$, $\hat{R}_{\text{EXP}}(B=5, \lambda=0)$, and $\hat{R}_{\text{EXP}}(B=|S|, \lambda=0)$. For the evaluation in this section, we use the following parameter choices for EXPRD: $H = \{1, 4, 8, 16, 32\}$, $\ell$ is the negated hinge loss $\ell_{\text{hinge}}$, and $\Pi^\dagger$ contains only one policy from $\Pi^\dagger$.

**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 $\hat{R}$, however, is evaluated based on $R$. 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 40 runs and convergence plots show mean with standard error bars. The convergence behavior in Figure 3a demonstrates the effectiveness of the reward functions designed by our EXPRD framework.\(^5\) Note that $\hat{R}_{\text{RAFT}}(B=5)$ leads to sub-optimal behavior due to “reward bugs” (see Section 2), whereas $\hat{R}_{\text{EXP}}(B=5, \lambda \to \infty)$ 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 $\hat{R}_{\text{EXP}}(B=3, \lambda=0)$ is comparable to that of $\hat{R}_{\text{EXP}}(B=5, \lambda=0)$ in Figures 3b, 3c, and 3d. The visualizations of $\hat{R}_{\text{RAFT}}$, $\hat{R}_{\text{PBRS}}$, and $\hat{R}_{\text{EXP}}(B=5, \lambda=0)$ provide a good balance in terms of convergence performance while maintaining high sparseness. Additional visualizations and results are provided in the Appendix.

### 4.2 Evaluation on LINEKeyNAVEnv

**LINEKeyNAVEnv (Figure 4).** We represent the environment as an MDP with $S$ 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 $[0.5, 1]$, and the key is available in locations on the segment $[0.1, 0.2]$. The agent can take three actions given by $A := \{\text{“left”}, \text{“right”}, \text{“pick”}\}$. “Pick” action does not change the agent’s location. Figure 4 shows partial results for $\hat{R}_{\text{PBRS}}$ designed using $\Pi^\dagger$ makes the agent’s learning process trivial.

---

\(^5\)As we discussed in Sections 1 and 2, $\hat{R}_{\text{PBRS}}$ designed using $\Pi^\dagger$ makes the agent’s learning process trivial.
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 $(x - \Delta_{a,1} - \Delta_{a,2})$ to $(x - \Delta_{a,1} + \Delta_{a,2})$. Similar to ROOMSNAVENV, the agent’s move action is not applied if the new location crosses the wall, and there is $p_{\text{rand}}$ probability of a random action. The agent gets a reward of $R_{\text{max}}$ 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 $\gamma$. We set $p_{\text{rand}} = 0.1, R_{\text{max}} = 10, \gamma = 0.95, \Delta_{a,1} = 0.075$ and $\Delta_{a,2} = 0.01$.  

Techniques evaluated. The baseline $\hat{R}_{\text{Orig}} := \bar{R}$ represents the default reward function. We evaluate the variants of $\hat{R}_{\text{PBRS}}$ and $\hat{R}_{\text{EXPDRD}}$ using an abstraction. For a given hyperparameter $\alpha \in (0, 1)$, the set of possible locations $X$ are obtained by $\alpha$-level discretization of the line segment from $0.0$ to $1.0$, leading to a $1/\alpha$ set of locations. For the abstraction $\phi$ associated with this discretization [43], the abstract MDP $\bar{M}_{\phi}$ (see Section 3.5) has $|X_{\phi}| = 2^\alpha$ and $|A| = 3$. We use $\alpha = 0.05$. We compute the optimal state value function in the abstract MDP $\bar{M}_{\phi}$, lift it to the original state space via $\phi$, and use the lifted value function as the potential to design $\hat{R}_{\text{PBRS}}$ [35]. We follow the pipeline in Section 3.5 to design $\hat{R}_{\text{EXPDRD}}$ – in the subroutine, we run EXPDRD on $\bar{M}_{\phi}$ for a budget $B = 5$ and a full budget $B = |X_{\phi}|$; we set $\lambda = 0$. For other parameters $(\mathcal{H}, \ell, \Pi)$, we use the same choices as in Section 4.1.  

Results. The agent uses Q-learning method in the original MDP $\bar{M}$ 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 40 runs and convergence plots show mean with standard error bars. Figure 5a demonstrates that all three designed reward functions—$\hat{R}_{\text{PBRS}}, \hat{R}_{\text{EXPDRD}(B=5, \lambda=0)}$, $\hat{R}_{\text{EXPDRD}(B=|X_{\phi}|, \lambda=0)}$—substantially improves the convergence, whereas the agent is not able to learn under $\hat{R}_{\text{Orig}}$. Based on the visualizations in Figures 5b, 5c, and 5d, $\hat{R}_{\text{EXPDRD}(B=5, \lambda=0)}$ provides a good balance between convergence and sparseness. Interestingly, $\hat{R}_{\text{EXPDRD}(B=5, \lambda=0)}$ assigned a high positive reward for the “pick” action when the agent is in the locations with key (see "p" bar in Figure 5d).  

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, 14, 37, 38, 40, 44, 45, 46–48]). As we discussed in Section 2, the shaped reward function $\hat{R}_{\text{PBRS}}$ is obtained by modifying $\hat{R}$ using a state-dependent potential function. The technique preserves a strong invariance property: a
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. Importantly, ExpRD allows one to go beyond using a potential function for principled reward design, and provides a general recipe for developing an optimization-based reward design framework with different structural constraints. 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 as well as other structured rewards, (b) extending our framework for agent-driven reward design, (c) applying our framework in a transfer setting using techniques from [42, 57], and (d) investigating the usage of our informativeness criterion for discovering subgoals.

Optimization-based techniques for reward design. Beyond potential-based shaping, we can formulate reward design as an optimization problem [15–19]. 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 [17–19]. Our ExpRD framework builds on the optimization framework of [17–19]. 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, 49–53]. These agent-driven techniques use a wide-variety of ideas such as designing intrinsic rewards based on exploration bonus [49, 50, 54], designing rewards using some additional domain knowledge [51], and using credit assignment to create intermediate rewards [6, 52].

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, 14–19, 48]. 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 [13, 14, 37, 40, 55, 56]. In recent works [13, 14, 55, 56], 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 [21–24] – 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. Importantly, ExpRD allows one to go beyond using a potential function for principled reward design, and provides a general recipe for developing an optimization-based reward design framework with different structural constraints. 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 as well as other structured rewards, (b) extending our framework for agent-driven reward design, (c) applying our framework in a transfer setting using techniques from [42, 57], and (d) investigating the usage of our informativeness criterion for discovering subgoals.
References


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 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 Assumption 2 (see Appendix) 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 as a URL.
   (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 3a and 5a.
   (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]
A List of Appendices

In this section, we give a brief description of the content provided in the appendices of the paper.

- Appendix B provides proofs for Propositions 1 and 2. (Sections 3.2 and 3.3)
- Appendix C provides additional details and proofs for the theoretical analysis. (Section 3.4)
- Appendix D provides additional details and proofs for using state abstractions. (Section 3.5)
- Appendix E provides additional results for ROOMSNAVENV. (Section 4.1)
- Appendix F provides additional results for LINEKEYNAVENV. (Section 4.2)

B Proofs for Propositions 1 and 2 (Sections 3.2 and 3.3)

B.1 Proof of Proposition 1

**Proposition 1.** The goal-based reward function $R$, and the potential-based shaped reward function $R_{PBRS}$ given in (1) satisfy the following: $\delta_h^\pi(s, a) = \delta_\infty^\pi(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}, \pi \in \Pi^*, h \in \mathcal{H}.$

**Proof.** Consider any optimal policy $\pi \in \Pi^*, s \in \mathcal{S}, a \in \mathcal{A}$, and $h \in \mathcal{H}$. The $\infty$-step optimality gap induced by $R$ is $\delta_\infty^\pi(s, a) = V_\infty^*(s) - Q_\infty^*(s, a)$, and the $h$-step optimality gap induced by $R_{PBRS}$ is $\delta_h^\pi(s, a) = \tilde{Q}_h^\pi(s, \pi(s)) - \tilde{Q}_h^\pi(s, a)$. In the following, we express the two terms of $\delta_h^\pi$ in terms of $V_\infty^*$ and $\tilde{Q}_\infty^*$.

**The term $\tilde{Q}_h^\pi(s, \pi(s))$ for any $\pi \in \Pi^*$.** We show that $\tilde{Q}_h^\pi(s, \pi(s)) = 0$ for any non-negative integer $h$ by using mathematical induction. First ($h = 0$ case), we consider the 0-step optimal action value function:

$$\tilde{Q}_0^\pi(s, \pi(s)) = R_{PBRS}(s, \pi(s)) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' | s, \pi(s)) \tilde{V}_h^\pi(s') - \tilde{V}_h^\pi(s) = 0.$$  

Now assume that $\tilde{Q}_{h-1}^\pi(s, \pi(s)) = 0$. Then, consider the $h$-step optimal action value function:

$$\tilde{Q}_h^\pi(s, \pi(s)) = R_{PBRS}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' | s, \pi(s)) \tilde{Q}_{h-1}^\pi(s', \pi(s)) = 0.$$  

Thus, by mathematical induction, we have that $\tilde{Q}_h^\pi(s, \pi(s)) = 0$ for any non-negative integer $h$.

**The term $\tilde{Q}_h^\pi(s, a)$ for any $a \in \mathcal{A}$.** Consider the $h$-step optimal action value function:

$$\tilde{Q}_h^\pi(s, a) = R_{PBRS}(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s' | s, a) \tilde{Q}_{h-1}^\pi(s', \pi(s')) = 0.$$  

Finally, by combining these two terms, we get:

$$\delta_h^\pi(s, a) = \tilde{Q}_h^\pi(s, \pi(s)) - \tilde{Q}_h^\pi(s, a) = V_\infty^*(s) - \tilde{Q}_\infty^*(s, a) = \delta_\infty^*(s, a).$$  

$\square$
B.2 Proof of Proposition 2

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

**Proof.** We write the problem (P1) explicitly as follows:

\[
\max_{R} \sum_{\pi \in \Pi^{\dagger}} \sum_{h \in \mathcal{H}} \sum_{s \in S} \sum_{a \in A \setminus \Pi^{\dagger}_{s}} \ell(h^{\dagger}(s, a))
\]

subject to \( R(s, a) = 0, \forall s \in S \setminus \{Z \cup G\}, a \in A \)

\[
Q_{\infty}^{\dagger}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a)Q_{\infty}^{\dagger}(s', \pi^\dagger(s')) \forall s \in S, a \in A, \pi^\dagger \in \Pi^{\dagger}
\]

\[
Q_{\infty}^{\dagger}(s, a) \geq Q_{\infty}^{\dagger}(s, a) + \delta_{\infty}(s), \forall s \in S, a \in A \setminus \Pi^{\dagger}_{s}, \pi^\dagger \in \Pi^{\dagger}
\]

\[
Q_{0}^{\dagger}(s, a) = R(s, a), \forall s \in S, a \in A, \pi^\dagger \in \Pi^{\dagger}
\]

\[
Q_{h}^{\dagger}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a)Q_{h-1}^{\dagger}(s', \pi^\dagger(s')) \forall s \in S, a \in A, h \in \mathcal{H}, \pi^\dagger \in \Pi^{\dagger}
\]

\[
\delta_{h}^{\dagger}(s, a) = Q_{h}^{\dagger}(s, \pi^\dagger(s)) - Q_{h}^{\dagger}(s, a), \forall s \in S, a \in A, h \in \mathcal{H}, \pi^\dagger \in \Pi^{\dagger}
\]

\[
|R(s, a)| \leq R_{\max}, \forall s \in S, a \in A
\]

In the following, we show that the above problem is a concave optimization problem (the objective is concave and the constraints are linear) by writing it in the matrix form as follows:

\[
\max_{R \in [0, \infty)^{|S| \times |A|}} \sum_{\pi \in \Pi^{\dagger}} \sum_{h \in \mathcal{H}} \sum_{s \in S} \sum_{a \in A \setminus \Pi^{\dagger}_{s}} \ell(\langle w_{h}(s, a), R \rangle)
\]

subject to \( A \cdot R \leq b \),

for some vectors \( w_{h}(s, a), b \in \mathbb{R}^{S \times A} \), and some matrix \( A \in \mathbb{R}^{S \times A} \).

**Notation.** We mainly follow the notation from [58]. Given a deterministic policy \( \pi : S \to A \), we define the transition matrix \( T_{\pi} \in \mathbb{R}^{|S| \times |A|} \) induced by \( \pi \) as follows:

\[
[T_{\pi}]_{(s,a),(s',a')} := \begin{cases} T(s'|s, a), & \text{if } a' = \pi(s') \\ 0, & \text{otherwise.} \end{cases}
\]

Also, for any \( s \in S \), we define \( \text{Id}_{\pi}(s) \in \mathbb{R}^{|A| \times |A|} \) as follows:

\[
[\text{Id}_{\pi}(s)]_{a,a} := \begin{cases} 1, & \text{if } a = \pi(s) \\ 0, & \text{otherwise.} \end{cases}
\]

Then, we define \( \text{Id}_{\pi} \in \mathbb{R}^{S \times A \times |S| \times |A|} \) as a block diagonal matrix with block size of \( |A| \times |A| \), and \( \text{Id}_{\pi}(s) \) as the \( s \)th diagonal block, \( \forall s \in S \). We define the diagonal matrix \( L_{\Pi^{\dagger}} \in \mathbb{R}^{S \times A \times |S| \times |A|} \), whose \( (s, a) \)th diagonal entry is given by:

\[
[L_{\Pi^{\dagger}}]_{(s,a),(s,a)} := \begin{cases} 0, & \text{if } a \in \Pi^{\dagger}_{s} \\ 1, & \text{otherwise.} \end{cases}
\]

We define the diagonal matrix \( L_{Z} \in \mathbb{R}^{S \times A \times |S| \times |A|} \), whose \( (s, a) \)th diagonal entry is given by:

\[
[L_{Z}]_{(s,a),(s,a)} := \begin{cases} 0, & \text{if } s \in Z \\ 1, & \text{otherwise.} \end{cases}
\]

Let \( e_{i} \in \mathbb{R}^{S \times |A|} \) be a vector having 1 only in the \( i \)th entry, and 0 elsewhere. Let \( \delta^{*} \in \mathbb{R}^{|S| \times |A|} \) be a vector such that its \( (s, a) \)th entry is given by \( \delta^{*}_{(s,a)} = \delta_{\infty}(s), \forall a \in A \). Let 1 be in \( \mathbb{R}^{S \times |A|} \) be a vector of all ones. Let \( \text{Id} \in \mathbb{R}^{S \times A \times |S| \times |A|} \) be the identity matrix.
**Bound constraint.** The bound constraint in Eq. (10) can be written as follows:

\[ R_{\text{max}} \cdot 1 \succeq R \succeq - R_{\text{max}} \cdot 1. \]

The above is linear inequality in \( R \).

**Sparsity constraint.** The sparsity constraint in Eq. (4) can be written as follows:

\[ L_{\mathcal{Z}} R = 0. \]

The above is linear equality in \( R \).

**Global optimality constraints.** The recursive form of the action value function \( Q_{\pi}^{\infty}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a)Q_{\pi}^{\infty}(s', \pi(s')) \) can be written in the matrix form as follows:

\[ Q_{\pi}^{\infty} = R + \gamma T_{\pi}Q_{\pi}^{\infty} \implies Q_{\pi}^{\infty} = (\text{Id} - \gamma T_{\pi})^{-1} R. \]

Then, the global optimality constraints in Eq. (6) can be written as follows, for all \( \pi \in \Pi^{1} \):

\[ (\text{Id}_{\pi} - \text{Id}) Q_{\pi}^{\infty} \succeq L_{\Pi} \delta_{\pi} \implies (\text{Id}_{\pi} - \text{Id}) (\text{Id} - \gamma T_{\pi})^{-1} R \succeq L_{\Pi} \delta_{\pi}. \]

The above is linear inequality in \( R \).

**Information \( I_{\ell}(R) \) is concave in \( R \).** For \( h = 0 \), \( Q_{0}^{\pi}(s, a) = R(s, a) \) can be written as follows:

\[ Q_{0}^{\pi} = R. \]

For \( h = 1 \), \( Q_{1}^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a)Q_{0}^{\pi}(s', \pi(s')) \) can be written as follows:

\[ Q_{1}^{\pi} = R + \gamma T_{\pi}Q_{0}^{\pi} = (\text{Id} + \gamma T_{\pi}) R. \]

For \( h = 2 \), \( Q_{2}^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a)Q_{1}^{\pi}(s', \pi(s')) \) can be written as follows:

\[ Q_{2}^{\pi} = R + \gamma T_{\pi}Q_{1}^{\pi} = (\text{Id} + \gamma T_{\pi} + \gamma^2 T_{\pi}^2) R. \]

For any \( h, Q_{h}^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a)Q_{h-1}^{\pi}(s', \pi(s')) \) can be written as follows:

\[ Q_{h}^{\pi} = (\text{Id} + \gamma T_{\pi} + \gamma^2 T_{\pi}^2 + \cdots + \gamma^h T_{\pi}^h) R, \]

where \( T_{\pi}^{(h)} = T_{\pi}T_{\pi} \cdots T_{\pi} \). Then, we can write \( \delta_{h}^{\pi}(s, a) = Q_{h}^{\pi}(s, \pi(s)) - Q_{h}^{\pi}(s, a) \) as follows:

\[ \delta_{h}^{\pi}(s, a) = \left( (\text{Id}_{\pi} - \text{Id}) \left( \text{Id} + \gamma T_{\pi} + \gamma^2 T_{\pi}^2 + \cdots + \gamma^h T_{\pi}^h \right) R, e_{(s,a)} \right), \]

i.e., \( \delta_{h}^{\pi}(s, a) \) is linear in \( R \) for every \( s \in S \), and \( a \in A \). From the above equation, one can easily show that \( \delta_{h}^{\pi}(s, a) = \langle w_{h;\pi}^{\pi}(s,a), \hat{R} \rangle \), where \( w_{h;\pi}^{\pi}(s,a) := \rho_{h;\pi}(s,a) - \rho_{h;\pi}^{\pi}(s,a) \).

Since \( \ell : \mathbb{R} \to \mathbb{R} \) is monotonically non-decreasing concave function, we have that \( \ell \circ \delta_{h}^{\pi}(s, a) \) is concave [59]. From the fact that the sum of concave functions is concave, \( I_{\ell}(R) \) is concave in \( R \).

In summary, for the problem \((P1)\), the objective is concave and the constraints are of linear form \((A \cdot R \succeq b)\). Thus, \((P1)\) is a concave optimization problem.

**Feasibility.** One can easily verify that the original reward function \( \hat{R} \) satisfies all the constraints in (4)-(10) of the sparse reward shaping formulation for any \( \mathcal{Z} \), i.e., \( \hat{R} \) is a feasible solution. Furthermore, when \( \mathcal{Z} = S \setminus G \), the potential-based shaped reward function \( \hat{R}_{PBRS} \) given in (1) satisfies all the constraints in (4)-(10) of the sparse reward shaping formulation.

\[ \square \]
C Additional Details and Proofs for Theoretical Analysis (Section 3.4)

First, we define the submodularity and weak submodularity notions of a normalized set function, which are used in the proof of Theorem 1.

Definition 2 [Submodularity [60]]. Let \( g : 2^V \to \mathbb{R} \) be a normalized set function \( (g(\emptyset) = 0) \). \( g \) is submodular if for all \( W \subseteq V \) and \( j, k \in V \setminus W \):
\[
g(W \cup \{k\}) - g(W) \geq g(W \cup \{j, k\}) - g(W \cup \{j\}).
\]

Definition 3 [Weak Submodularity [30]]. Let \( \mathcal{Y}, \mathcal{X} \subseteq \mathcal{V} \) be two disjoint sets, and \( g : 2^V \to \mathbb{R} \) be a normalized set function. The submodularity ratio of \( \mathcal{X} \) with respect to \( \mathcal{Y} \) is given by
\[
\gamma_{\mathcal{X}, \mathcal{Y}} := \frac{\sum_{j \in \mathcal{Y}} (g(\mathcal{X} \cup \{j\}) - g(\mathcal{X}))}{g(\mathcal{X} \cup \mathcal{Y}) - g(\mathcal{X})}.
\]

The submodularity ratio of a set \( W \) with respect to an integer \( k \) is given by
\[
\gamma_{W,k} := \min_{\mathcal{Y},\mathcal{X} : \mathcal{Y} \cup \mathcal{X} \subseteq \mathcal{W}, |\mathcal{X}| \leq k} \gamma_{\mathcal{X}, \mathcal{Y}}.
\]

Let \( \gamma > 0 \). We call a function \( \gamma \)-weakly submodular at a set \( W \) and an integer \( k \) if \( \gamma_{W,k} \geq \gamma \).

A set function \( g : 2^V \to \mathbb{R} \) is called monotone if and only if \( g(\mathcal{X}) \leq g(\mathcal{Y}) \) for all \( \mathcal{X} \subseteq \mathcal{Y} \).

For any \( x \in \mathbb{R}^{|\mathcal{S}|-|\mathcal{A}|} \) and \( U \subseteq \mathcal{S} \), \( x_U \) is defined as \( x_{U} (j, a) = x(j, a), \forall a \in \mathcal{A} \) when \( j \in U \), and \( x_{U} (j, a) = 0, \forall a \in \mathcal{A} \) otherwise. For any \( j \in \mathcal{S} \), \( e_j \in \mathbb{R}^{|\mathcal{S}|-|\mathcal{A}|} \) is defined as \( e_j(j', a) = 1, \forall a \in \mathcal{A} \) when \( j' = j \), and \( e_j(j', a) = 0, \forall a \in \mathcal{A} \) otherwise. The following assumption captures the additional requirements on the regularized informativeness criterion \( I^\text{reg}_f \):

Assumption 2. Let \( \mathcal{Z} \) be any set such that \( \mathcal{Z} \subseteq \mathcal{S}\setminus\mathcal{G} \). The regularized informativeness criterion \( I^\text{reg}_f \) satisfies the following:

- \( \left\| \nabla I^\text{reg}_f(R(\mathcal{Z})) \right\|_2 \leq d^\text{opt}_{\max} \),
- \( \left\| \nabla I^\text{reg}_f(R(\mathcal{Z})) \right\|_2 \geq d^\text{non}_{\min}, \forall j \in \mathcal{S}\setminus(\mathcal{Z} \cup \mathcal{G}) \),
- \( \left\| \nabla I^\text{reg}_f(R(\mathcal{Z})) \right\|_\infty \leq d^\text{non}_{\max}, \forall j \in \mathcal{S}\setminus(\mathcal{Z} \cup \mathcal{G}) \), and
- \( \exists \kappa \leq 1 \text{ such that } \forall j \in \mathcal{S}\setminus(\mathcal{Z} \cup \mathcal{G}) : R(\mathcal{Z}) \pm \kappa \cdot \frac{d^\text{non}_{\max}}{M_{|\mathcal{Z}|+|\mathcal{G}|+1}} \cdot e_j \in \mathcal{R} \).

C.1 Proof of Theorem 1

Let \( \mathcal{Z} \subseteq \mathcal{S}\setminus\mathcal{G} \). Consider the set function \( f : 2^\mathcal{S} \to \mathbb{R}_+ \) defined in (2):
\[
f(\mathcal{Z}) = \max_{R : \text{supp}(R) \subseteq \mathcal{Z}\setminus\mathcal{G}, R \in \mathcal{R}} (I^\text{reg}_f(R) - I^\text{reg}_f(R^{(\emptyset)})) + \lambda \cdot (D(\mathcal{Z} \cup \mathcal{G}) - D(\mathcal{G})),
\]
where \( R^{(\emptyset)} = \arg \max_{R : \text{supp}(R) \subseteq \mathcal{G}, R \in \mathcal{R}} I^\text{reg}_f(R) \). Note that \( f \) is a normalized, monotone set function.

For a given sparsity budget \( B \), let \( \mathcal{Z}^\text{Greedy}_B \) be the set selected by our Algorithm 1, and \( \mathcal{Z}^\text{OPT}_B \) be the optimal set that maximizes the regularized variant of problem (P3). The corresponding \( f \) values of these sets are denoted by \( f^\text{Greedy}_B \) and \( f^\text{OPT}_B \) respectively.

Theorem 1. Let \( I^\text{reg}_f \) satisfies Assumption 1 and Assumption 2 requirements. Then, we have \( f^\text{Greedy}_B \geq (1 - e^{-\gamma}) f^\text{OPT}_B \), where \( \gamma = \frac{\kappa - m_{2B+|\mathcal{G}]}{M_{2B+|\mathcal{G}|}} \cdot \frac{d^\text{non}_{\max}}{(d^\text{non}_{\max})^2 + (d^\text{non}_{\min})^2} \).

Proof. If \( f \) is \( \gamma \)-weakly submodular at the set \( \mathcal{Z}_B \) and the integer \( B \) (i.e., \( \gamma_{Z_B,B} \geq \gamma \)), then, using Theorem 3 from [29] (which holds for any normalized, monotone, \( \gamma \)-weakly submodular function), we can complete the proof of Theorem 1:
\[
f^\text{Greedy} \geq (1 - e^{-\gamma}) f^\text{OPT} \geq (1 - e^{-\gamma}) f^\text{OPT}.
\]
Thus, it remains to prove the weak submodularity of \( f \). Let \( f_0 \) denote \( f \) with \( \lambda = 0 \), and define \( D(Z) := D(Z \cup G) - D(G) \). Note that \( D \) is a normalized, monotone, submodular function. Then, the submodularity ratio of \( f \) with general \( \lambda \) is bounded as follows:

\[
\gamma_{X,Y} = \frac{\sum_{j \in Y} (f_0(X \cup \{j\}) - f_0(X)) + \lambda \sum_{j \in Y} (D(X \cup Y) - D(X))}{f_0(X \cup Y) - f_0(X) + \lambda (D(X \cup Y) - D(X))} \\
\geq \min \left( \frac{\sum_{j \in Y} (f_0(X \cup \{j\}) - f_0(X))}{f_0(X \cup Y) - f_0(X)}, 1 \right),
\]

where the inequality is due to the fact that the submodularity ratio of \( D \) is \( \geq 1 \) [29]. If the submodularity ratio of \( f_0 \) is \( \geq 1 \), then \( \gamma_{X,Y} \geq 1 \). This would lead to the following bound:

\[
f_{\text{Greedy}} \geq \left( 1 - \frac{1}{\epsilon} \right) f_{\text{OPT}}.
\]

If the submodularity ratio of \( f_0 \) is \( \leq 1 \) (this would be the case in general; thus, we consider this case in the theorem), then the submodularity ratio \( \gamma_{X,Y} \) of \( f \) with general \( \lambda \) is lower bounded by the submodularity ratio of \( f_0 \). By applying Lemma 1 with \( \left( Z_{\text{Greedy}}^B, B \right) \), we have that (since \( |Z_{\text{Greedy}}^B| = B \)):

\[
\gamma_{Z_{\text{Greedy}}^B, B} \geq \frac{\kappa \cdot m_{2B+\|G\|}}{M_{2B+\|G\|}} \cdot \left( \frac{(d_{\text{min}}^\text{non})^2}{(d_{\text{max}}^\text{opt})^2 + (d_{\text{min}}^\text{opt})^2} \right) =: \gamma.
\]

This completes the proof.

\[\square\]

The following lemma provides a lower bound on the submodularity ratio \( \gamma_{Z,k} \) of \( f_0 \) (for any \( Z \) s.t. \( |Z| \leq B \) and \( k \leq B \)).

**Lemma 1.** Let the regularized informativeness criterion \( I^\text{reg}_\ell \) satisfies the Assumption 1 and 2. Then, for any set \( Z \) s.t. \( Z \subseteq S \setminus G, |Z| \leq B \), and \( k \leq B \), the submodularity ratio \( \gamma_{Z,k} \) of \( f_0 \) is lower bounded by

\[
\gamma_{Z,k} \geq \frac{\kappa \cdot m_{|Z|+\|G\|+k}}{M_{|Z|+\|G\|+k}} \cdot \left( \frac{(d_{\text{min}}^\text{non})^2}{(d_{\text{max}}^\text{opt})^2 + (d_{\text{min}}^\text{opt})^2} \right).
\]

**Proof.** Since \( I^\text{reg}_\ell \) is \( m_{2B+\|G\|}\)-restricted strongly concave and \( M_{2B+\|G\|} \)-restricted smooth on \( \Omega_{2B+\|G\|} \), we have that \( I^\text{reg}_\ell \) is \( m_{|Z|+\|G\|+k} \)-restricted strongly concave and \( M_{|Z|+\|G\|+k} \)-restricted smooth on \( \Omega_{|Z|+\|G\|+k} \) for any \( Z \) s.t. \( |Z| \leq B \) and \( k \leq B \). In addition, \( I^\text{reg}_\ell \) is \( M_{|Z|+\|G\|+1} \)-restricted smooth on \( \Omega_{|Z|+\|G\|+1} \) since \( \Omega_{|Z|+\|G\|+k} \supseteq \Omega_{|Z|+\|G\|+1} \supseteq \Omega_{|Z|+\|G\|+1} \) and \( M_{|Z|+\|G\|+k} \geq M_{|Z|+\|G\|+1} \).

Consider the two sets \( X, Y \) such that \( (X \cup G) \cap Y = \emptyset, X \subseteq Z, \) and \( |Y| \leq k \). We proceed by upper bounding the denominator and lower bounding the numerator of Eq. (11). Let \( \bar{K} = |X| + |G| + k \).

First, we apply Definition 1 with \( x = R(X) \) and \( y = R(X \cup Y) \) (note that \( (x, y) \in \Omega_{\bar{K}} \)):

\[
\frac{m_{\bar{K}}}{2} \left\| R(X \cup Y) - R(X) \right\|^2 \leq I^\text{reg}_\ell \left( R(X \cup Y) - R(X) \right) + \left( \nabla I^\text{reg}_\ell (R(X)), R(X \cup Y) - R(X) \right).
\]

Rearranging and noting that \( I^\text{reg}_\ell \) is monotone for increasing supports:

\[
0 \leq I^\text{reg}_\ell \left( R(X \cup Y) \right) - I^\text{reg}_\ell \left( R(X) \right) \leq \left( \nabla I^\text{reg}_\ell (R(X)), R(X \cup Y) - R(X) \right) - \frac{m_{\bar{K}}}{2} \left\| R(X \cup Y) - R(X) \right\|^2 \\
\leq \max_{v \in (X \cup Y \cup G)} \left( \nabla I^\text{reg}_\ell (R(X)), v - R(X) \right) - \frac{m_{\bar{K}}}{2} \left\| v - R(X) \right\|^2.
\]

Setting \( v = R(X) + \frac{1}{m_{\bar{K}}} \nabla I^\text{reg}_\ell (R(X)) \chi_{X \cup Y \cup G} \) that achieves the maximum above, we have

\[
0 \leq I^\text{reg}_\ell \left( R(X \cup Y) \right) - I^\text{reg}_\ell \left( R(X) \right) \leq \frac{1}{2m_{\bar{K}}} \left\| \nabla I^\text{reg}_\ell (R(X)) \chi_{X \cup Y \cup G} \right\|^2.
\]
Then, we have:

\[
\frac{1}{2m_{\pi}} \left( \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_{X \cup G} \right\|_2^2 + \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_{Y} \right\|_2^2 \right),
\]

where the last equality is due to \((\mathcal{X} \cup G) \cap \mathcal{Y} = \emptyset\).

Next, consider a single state \(j \in \mathcal{Y}\). The function \(I_{\ell}^{\text{reg}}\) at \(R(\mathcal{X} \cup \{j\})\) is larger than the function at any other \(R\) on the same support. In particular, \(I_{\ell}^{\text{reg}}(R(\mathcal{X} \cup \{j\})) \geq I_{\ell}^{\text{reg}}(y_j)\), where \(y_j := R(\mathcal{X}) + \frac{\kappa}{M_{|\mathcal{X}|+|G|+1}} \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_j\). Noting that \((x = R(\mathcal{X}), y = y_j) \in \bar{\Omega}_{|\mathcal{X}|+|G|+1}\) and applying Definition 1:

\[
I_{\ell}^{\text{reg}}(R(\mathcal{X} \cup \{j\})) - I_{\ell}^{\text{reg}}(R(\mathcal{X}))
\]

\[
\geq I_{\ell}^{\text{reg}} \left( R(\mathcal{X}) + \frac{\kappa}{M_{|\mathcal{X}|+|G|+1}} \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_j \right) - I_{\ell}^{\text{reg}}(R(\mathcal{X}))
\]

\[
\geq \left\langle \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X})), \frac{\kappa}{M_{|\mathcal{X}|+|G|+1}} \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_j \right\rangle - \frac{\kappa}{2M_{|\mathcal{X}|+|G|+1}} \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_j \right\|_2^2
\]

\[
= \frac{\kappa}{2M_{|\mathcal{X}|+|G|+1}} \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_j \right\|_2^2.
\]

Summing over all \(j \in \mathcal{Y}\):

\[
\sum_{j \in \mathcal{Y}} \left[ I_{\ell}^{\text{reg}}(R(\mathcal{X} \cup \{j\})) - I_{\ell}^{\text{reg}}(R(\mathcal{X})) \right] \geq \frac{\kappa}{2M_{|\mathcal{X}|+|G|+1}} \sum_{j \in \mathcal{Y}} \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_j \right\|_2^2
\]

\[
= \frac{\kappa}{2M_{|\mathcal{X}|+|G|+1}} \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_{\mathcal{Y}} \right\|_2^2.
\]

Then, we have:

\[
\gamma_{\mathcal{X}, \mathcal{Y}} \geq \frac{\kappa \cdot m_{|\mathcal{X}|+|G|+k}}{M_{|\mathcal{X}|+|G|+k}} \cdot \frac{1}{\left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_{X \cup G} \right\|_2^2 / 2 + \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_Y \right\|_2^2}
\]

\[
\geq \frac{\kappa \cdot m_{|\mathcal{X}|+|G|+k}}{M_{|\mathcal{X}|+|G|+k}} \cdot \frac{1}{\left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_{X \cup G} \right\|_2^2 / 2 + \left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_Y \right\|_2^2 + 1}
\]

\[
\geq \frac{\kappa \cdot m_{|\mathcal{X}|+|G|+k}}{M_{|\mathcal{X}|+|G|+k}} \cdot \frac{1}{\left( d_{\text{max}}^{\text{opt}} \right)^2 + 1} \cdot \frac{1}{\left( d_{\text{min}}^{\text{non}} \right)^2 + 1},
\]

where (i) is due to \(\left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_{X \cup G} \right\|_2^2 \leq (d_{\text{max}}^{\text{opt}})^2 \) and \(\left\| \nabla I_{\ell}^{\text{reg}}(R(\mathcal{X}))_Y \right\|_2^2 \geq |\mathcal{Y}| (d_{\text{min}}^{\text{non}})^2 \geq (d_{\text{mon}}^{\text{non}})^2 \) (see Assumption 2); and (ii) is due to \(m_{|\mathcal{X}|+|G|+k} \geq m_{|\mathcal{Z}|+|G|+k} \) and \(M_{|\mathcal{X}|+|G|+k} \geq M_{|\mathcal{Z}|+|G|+k} \geq M_{|\mathcal{X}|+|G|+1} = M_{|\mathcal{Z}|+|G|+1} \) (note that \(1 \leq |\mathcal{Y}| \leq k \) and \(1 \leq |\mathcal{X}| \leq |\mathcal{Z}|\).
D Additional Details and Proofs for using State Abstractions (Section 3.5)

We present an extension of our EXP-RD framework that is scalable to large state spaces by leveraging the techniques from state abstraction literature [32–34]. 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 (called EXP-RD-Abs):

1. By using the original MDP \( \mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, R) \) and the abstraction \( \phi \), we construct an abstract MDP \( \mathcal{M}_\phi = (\mathcal{X}_\phi, \mathcal{A}, T_\phi, \gamma, P_0, R_\phi) \) 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 \( R_\phi(x, a) = \frac{1}{|\phi^{-1}(x)|} \sum_{s \in \phi^{-1}(x)} R(s, a) \). We compute the set of optimal policies \( \hat{\Pi}_\phi \) for the MDP \( \mathcal{M}_\phi \).

2. We run our EXP-RD framework on \( \mathcal{M}_\phi \) with \( \Pi^* = \hat{\Pi}_\phi \), and the resulting reward is denoted \( \hat{R}_\phi \).

3. We define the reward function \( \hat{R} \) on the state space \( \mathcal{S} \) as follows: \( \hat{R}(s, a) = \hat{R}_\phi(\phi(s), a) \). The corresponding MDP is denoted by \( \hat{M} = (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, \hat{R}) \).

In summary, the EXP-RD-Abs pipeline is given by: \( \mathcal{M} \to \mathcal{M}_\phi \to \hat{M}_\phi \to \hat{M} \).

Define \( \epsilon_\phi := \min_{x \in \mathcal{X}_\phi} \min_{a \in \mathcal{A}} |\Pi_{\phi,x} - \hat{\Pi}_{\phi,x}|, \) where \( \delta_{\phi,\infty} \) is the \( \infty \)-step optimality gap in the abstract MDP \( \mathcal{M}_\phi = (\mathcal{X}_\phi, \mathcal{A}, T_\phi, \gamma, P_0, R_\phi) \). For our analysis, we require the abstraction \( \phi : \mathcal{S} \to \mathcal{X}_\phi \) to satisfy the conditions discussed above. The original reward function \( R \) and the reward function \( \hat{R} \) output by the EXP-RD-Abs pipeline satisfy the following:

- \( \phi \) is \((\epsilon_\Pi, \epsilon_T)\)-approximate model irrelevant abstraction [34] for the MDP \( \mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma, P_0, R) \), i.e., \( \forall s_1, s_2 \in \mathcal{S} \) where \( \phi(s_1) = \phi(s_2) \), we have, \( \forall a \in \mathcal{A} : |R(s_1, a) - R(s_2, a)| \leq \epsilon_\Pi \) and \( \sum_{x' \in \mathcal{X}_\phi} \sum_{s' \in \phi^{-1}(x')} (T(s'|s, a) - T(s'|s', a)) \leq \epsilon_T \).

- The change in the transition dynamics \( T \) during the compression-decompression process using the abstraction \( \phi \) is very small, i.e., \( \max_{s, a} \sum_{s'} (T(s'|s, a) - T_\phi(s'|s, a)) \leq \frac{\epsilon_\Pi}{2(1-\gamma)} \).

The following theorem shows that any optimal policy induced by the reward \( \hat{R} \) resulting from the EXP-RD-Abs pipeline acts nearly optimal w.r.t. \( \hat{R} \):

**Theorem 2.** Let \( \phi : \mathcal{S} \to \mathcal{X}_\phi \) satisfy the conditions discussed above. The original reward function \( R \), and the reward function \( \hat{R} \) output by the EXP-RD-Abs pipeline satisfy the following: \( \max_{s} |\nabla^*_s (s) - \nabla^*_\phi (s)| \leq \frac{2\epsilon_\Pi}{1-\gamma} + \frac{2\epsilon_T \gamma R_{\max}}{2(1-\gamma)^2}, \forall \Pi \in \hat{\Pi}^*, i.e., any optimal policy induced by \( \hat{R} \) acts nearly optimal w.r.t. \( \hat{R} \).

**Proof.** Given an abstract policy \( \pi : \mathcal{X}_\phi \to \mathcal{A} \) acting on \( \mathcal{X}_\phi \), we define the lifted policy \( \pi_{\hat{M}} : \mathcal{S} \to \mathcal{A} \) as \( \pi_{\hat{M}}(s) := \pi(\phi(s)), \forall s \in \mathcal{S} \). Similarly, given a set of policies \( \Pi = \{ \pi : \mathcal{X}_\phi \to \mathcal{A} \} \), we define \( \Pi_{\hat{M}} := \{ \pi_{\hat{M}} : \pi \in \Pi \} \). We define an auxiliary MDP \( \hat{M} = (\mathcal{S}, \mathcal{A}, \hat{T}, \gamma, P_0, \hat{R}) \), where \( \hat{R}(s, a) = \hat{R}_\phi(\phi(s), a) \), and \( \hat{T}(s'|s, a) = \frac{T_\phi(\phi(s'), \phi(s), a)}{|\phi^{-1}(\phi(s'))|} \).

**Step \( \mathcal{M} \to \mathcal{M}_\phi \).** Since \( \phi \) is \((\epsilon_\Pi, \epsilon_T)\)-approximate model irrelevant abstraction, we have the following (see [34]):

\[
|\nabla^*_\phi (s, a) - \nabla^*_\phi (\phi(s), a)| \leq \frac{\epsilon_\Pi}{1-\gamma} + \frac{\gamma \cdot \epsilon_T \cdot R_{\max}}{2(1-\gamma)^2}, \forall s \in \mathcal{S}, a \in \mathcal{A},
\]

where \( \nabla^*_\phi \) is the optimal action value function of the MDP \( \mathcal{M}_\phi \). Then, for any \( \pi \in \hat{\Pi}_{\phi,\hat{M}} \), we have the following (see [61]):

\[
\max_{s} |\nabla^*_\phi (s) - \nabla^*_\phi (\phi(s))| \leq \frac{2}{1-\gamma} \max_{s, a} |\nabla^*_\phi (s, a) - \nabla^*_\phi (\phi(s), a)| \leq \frac{2\epsilon_\Pi}{(1-\gamma)^2} + \frac{\gamma \cdot \epsilon_T \cdot R_{\max}}{2(1-\gamma)^3},
\]

i.e., any optimal policy of \( \mathcal{M}_\phi \), when lifted to \( \mathcal{S} \), acts as a near-optimal policy in \( \hat{M} \).
Step $\tilde{M}_\phi \rightarrow \tilde{M}_\phi$. In the step 2 of our EXPRD-Abs pipeline, we set $\Pi^\dagger = \tilde{\Pi}^\phi_*$. Our EXPRD framework ensures that any optimal policy for $\tilde{M}_\phi$ is also optimal in $\tilde{M}_\phi$, i.e., $\tilde{\Pi}^\phi_* \subseteq \Pi^\dagger$. In addition, since $\Pi^\dagger = \Pi^\phi$ and $\Pi^\dagger \subseteq \tilde{\Pi}^\phi_*$, we have that $\tilde{\Pi}^\phi_\phi = \Pi^\phi$.

Step $\tilde{M}_\phi \rightarrow \tilde{M}$. By the definition of $\tilde{M}$, $\phi$ is a model irrelevant abstraction for $\tilde{M}$. Thus, we have the following (see [34]):

$$\tilde{Q}^\phi_\phi(s, a) = \tilde{Q}^\phi_\phi(\phi(s), a), \quad \forall s \in S, a \in A. \tag{12}$$

From the above equation, note that $\tilde{\Pi}^\phi = \tilde{\Pi}^\phi_* |_{\tilde{M}}$. Finally, we have that, for any $\pi \in \tilde{\Pi}^\phi$:

$$\max_s |\tilde{V}^\phi_\phi(s) - V^\pi_\phi(s)| \leq \frac{2 \epsilon_R}{(1 - \gamma)^2} + \frac{\gamma \cdot \epsilon_T \cdot R_{\max}}{2(1 - \gamma)^3},$$

i.e., any optimal policy of $\tilde{M}$ acts as a near-optimal policy in the original MDP $\tilde{M}$.

**Optimality in $\tilde{M}$**. Our EXPRD framework guarantees the following:

$$\tilde{\tilde{Q}}^\phi_\phi(x, \pi^\dagger(x)) \geq \tilde{\tilde{Q}}^\phi_\phi(x, a) + \epsilon_\phi, \quad \forall x \in X_\phi, a \in A \setminus \tilde{\Pi}^\phi_\phi, x, \pi^\dagger \in \Pi^\dagger,$$

which can be rewritten as follows:

$$\tilde{\tilde{Q}}^\phi_\phi(\phi(s), \pi^\dagger(\phi(s))) \geq \tilde{\tilde{Q}}^\phi_\phi(\phi(s), a) + \epsilon_\phi, \quad \forall s \in S, a \in A \setminus \tilde{\Pi}^\phi_\phi, \pi^\dagger \in \Pi^\dagger.$$

From the above inequality and using (12), we have the following:

$$\tilde{Q}^\phi_\phi(s, [\pi^\dagger]_{\tilde{M}}(s)) \geq \tilde{Q}^\phi_\phi(s, a) + \epsilon_\phi, \quad \forall s \in S, a \in A \setminus \Pi^\phi_\phi, [\pi^\dagger]_{\tilde{M}}, \pi^\dagger \in [\Pi^\dagger]_{\tilde{M}},$$

which can be rewritten as follows:

$$\tilde{Q}^\phi_\phi(s, \pi^\dagger(s)) \geq \tilde{Q}^\phi_\phi(s, a) + \epsilon_\phi, \quad \forall s \in S, a \in A \setminus \Pi^\phi_\phi, \pi^\dagger \in \tilde{\Pi}^\phi_\phi.$$

From the above inequality, for any deterministic policy $\pi \notin \tilde{\Pi}^\phi_\phi$, we have (at least on one state $s \in S$):

$$V^\phi_\phi(s) = \tilde{Q}^\phi_\phi(s, \pi^\dagger(s)) \geq \tilde{Q}^\phi_\phi(s, \pi(s)) + \epsilon_\phi \geq \tilde{Q}^\phi_\phi(s, \pi(s)) + \epsilon_\phi = \tilde{V}^\phi_\phi(s) + \epsilon_\phi,$$

i.e., $\max_s |\tilde{V}^\phi_\phi(s) - V^\pi_\phi(s)| \geq \epsilon_\phi$.

**Comparison $\tilde{M}$ vs. $\tilde{M}$**. Now, we show that any deterministic optimal policy in $\tilde{M}$ is also optimal in $\tilde{M}$, i.e., $\tilde{\Pi}^\phi_\phi \subseteq \tilde{\Pi}^\phi_*$. Let $\max_s, a \|T(\cdot | s, a) - \tilde{T}(\cdot | s, a)\|_1 = \beta_T$. Then, for any $\tilde{\pi} \in \tilde{\Pi}^\phi_\phi$ and $s \in S$, we have:

$$|\tilde{V}^\phi_\phi(s) - \tilde{V}^\pi_\phi(s)| \leq |\tilde{V}^\phi_\phi(s) - \tilde{V}^\tilde{\pi}_\phi(s)| + |\tilde{V}^\tilde{\pi}_\phi(s) - \tilde{V}^\phi_\phi(s)| \leq \frac{2 \gamma \beta_T R_{\max}}{(1 - \gamma)^2} < \epsilon_\phi,$$

where the second last inequality is due to Lemma 3 and Lemma 4 from [36]. Then, from the optimality in $\tilde{M}$, it must me the case that $\tilde{\pi} \in \tilde{\Pi}^\phi_\phi$.

Finally, for any $\pi \in \tilde{\Pi}^\phi_\phi$, we have:

$$\max_s |\tilde{V}^\phi_\phi(s) - V^\pi_\phi(s)| \leq \frac{2 \epsilon_\pi}{(1 - \gamma)^2} + \frac{\gamma \cdot \epsilon_T \cdot R_{\max}}{2(1 - \gamma)^3},$$

i.e., any optimal policy of $\tilde{M}$ acts as a near-optimal policy in the original MDP $\tilde{M}$. □
E Additional Details and Results for ROOMSNAVENV (Section 4.1)

In this appendix, we expand on Section 4.1 and provide a more detailed description of the setup as well as additional results. Full implementation of our techniques is available in a Github repo as mentioned in Footnote 1.

Recall that the MDP for ROOMSNAVENV has $|S| = 49$ states corresponding to cells in the grid-world and four actions given by $A := \{\text{“up”}, \text{“left”}, \text{“down”}, \text{“right”}\}$. To refer to a specific state, we will use an enumeration scheme where the bottom-left cell is $s = 0$; the cell numbers increase going from left to right and bottom to top. With this convention, the top-right cell with the goal is $s = 48$, and four “gates” (cells that need to be crossed to go across rooms when navigating to the goal) correspond to states $\{9, 15, 19, 37\}$. In this MDP, we have one goal state $s = 48$, i.e., the set $G$ in the problem (P3) is $\{48\}$. Furthermore, the original reward function has $R(48, \text{“right”}) = R_{\text{max}}$ and is 0 elsewhere.

Additional details for the techniques evaluated. Below, we describe different reward design techniques along with hyperparameters that are evaluated in this section. More concretely, we have:

(i) $\hat{R}_{\text{ORIG}}$ simply represents the default reward function $\bar{R}$.
(ii) $\hat{R}_{\text{PBRS}}$ is obtained via the PBRS technique based on Eq. 1, see Section 2.
(iii) $\hat{R}_{\text{RAFT}(B)}$ is designed manually based on the ideas discussed in Section 2. For selecting the states that we will assign non-zero rewards, we first develop a set function $D$ as described below after this list. Then, for a fixed budget $B$, we pick a set of top $B + |G|$ states that maximize the value of the set function $D$. Then, we assign rewards to these picked states as follows: (a) for the $B$ states excluding $|G|$ goal states, we assign a reward of $+1$ for one of the optimal action and $-1$ for others; (b) for $|G|$ goal states, we assign the same rewards as $R$. For the evaluation, we use $B = 5$ and denote the function as $\hat{R}_{\text{RAFT}(B=5)}$.
(iv) $\hat{R}_{\text{PBRS-RAFT}(B=5)}$ is obtained via the reward shaping technique from [42]. First, we compute the optimal state value function $\hat{V}^*_\infty$ w.r.t. $\hat{R}_{\text{RAFT}(B=5)}$ designed above, i.e., we need to solve the task with the reward function $\hat{R}_{\text{RAFT}(B=5)}$. Then, we obtain the reward function $\hat{R}_{\text{PBRS-RAFT}(B=5)}$ using the PBRS technique based on Eq. 1 with the value function $\hat{V}^*_\infty$ instead of the optimal value function $\hat{V}^*_\infty$ w.r.t. $\bar{R}$.
(v) $\hat{R}_{\text{EXPRD}(B, \lambda \to \infty)}$ is the reward function designed by our EXPRD framework for a budget $B$ and an extreme setting of $\lambda \to \infty$. For this setting, 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 $\hat{R}_{\text{RAFT}}$ above. For the evaluation, we use $B = 5$ and denote the designed reward function as $\hat{R}_{\text{EXPRD}(B=5, \lambda \to \infty)}$. As discussed in Section 3, the budget $B$ here refers to the additional number of states that are allowed to be in $\text{supp}(\bar{R})$ along with the goal states $G$ (see (P3)). Apart from hyperparameters $B$ and $\lambda$, EXPRD requires a choice of $\Pi^\dagger$, $H$, and $I(R)$—we discuss that below after this list.
(vi) $\hat{R}_{\text{EXPRD}(B, \lambda=0)}$ is the reward function designed by our EXPRD framework for a budget $B$ and an important setting of $\lambda = 0$ where the problem (P3) reduces to (P2) corresponding to fully automated reward design without using any prior knowledge about the importance of states. For budget $B$, we consider values from $\{3, 5, |S|\}$ and denote the designed reward functions as $\hat{R}_{\text{EXPRD}(B=3, \lambda=0)}$, $\hat{R}_{\text{EXPRD}(B=5, \lambda=0)}$, and $\hat{R}_{\text{EXPRD}(B=|S|, \lambda=0)}$. As stated above, the budget $B$ here refers to the additional number of states that are allowed to be in $\text{supp}(\bar{R})$ along with the goal states $G$; the choice of $\Pi^\dagger$, $H$, and $I(R)$ is discussed below.

Here we describe the set function $D$ used for computing $\hat{R}_{\text{RAFT}(B=5)}$ and $\hat{R}_{\text{EXPRD}(B=5, \lambda \to \infty)}$. For the set function $D$, we used a simple modular function given by $D(S) := \sum_{s \in S} w_s$ where $w_s$ is a weight/score assigned to a state $s$ capturing its importance in terms of reward design. We used the following weights: $w_s = 2$ for $s = 48$ (the goal state); $w_s = 1$ for $s = 0$, $s = 15$, $s = 19$, and $s = 37$ (the four “gates”); $w_s = 0.5$ for $s = 8$, $s = 11$, $s = 29$, and $s = 32$ (centers of the four rooms); and $w_s = 0.1$ otherwise. Even though this function is simple, it captures the prior knowledge one expects to intuitively apply in practice. In general, one could learn such $D$ automatically using the techniques from [21–24].
In Figure 6, we compare the designed reward functions w.r.t. these different criteria. In the “Sparseness” column, the quantity $|\text{supp}(\tilde{R})|$ is given by Eq. 15, and the set $\Pi^\perp$ contains only one policy from $\tilde{\Pi}$. Later in this section, we also consider variations of $\mathcal{H}$ and $I(\tilde{R})$, and report additional results in Figures 9 and 10.

Results w.r.t. different criteria. Next, we evaluate the above-mentioned designed reward functions w.r.t. criteria of sparseness, invariance, informativeness, and convergence. Sparseness is measured by $|\text{supp}(\tilde{R})|$, and informativeness is measured by $I(\tilde{R})$ that is used in the optimization problem (P3). Convergence is measured w.r.t. the number of episodes needed to get a specific % of the total expected reward, and is based on the convergence results in Figure 3a by taking various horizontal slices of the convergence plot. To measure the invariance property, we consider two different notions stated below:

$$\min_{\pi^* \in \Pi^*} \min_{s \in S} \left( Q^\pi_\infty(s, \pi^*(s)) - Q^\pi_\infty(s, \pi^*(s)) \right) \text{ for any } \pi^* \in \Pi^*$$  \hspace{1cm} (13)

$$\min_{\pi \in \Pi^*} \min_{s \in S} \min_{a \in A \setminus \Pi^*} \left( Q^\pi_\infty(s, \pi(s)) - Q^\pi_\infty(s, a) \right)$$ \hspace{1cm} (14)

The notion in Eq. (13) looks at one of the optimal policy $\pi^*$ w.r.t. $\tilde{R}$, and compares the gap in Q action values w.r.t. $\tilde{R}$ – this quantity should be zero to ensure that none of the optimal policies w.r.t. $\tilde{R}$ is suboptimal w.r.t. $\tilde{R}$. The notion in (14) is closer to the invariance constraint that we incorporate in the optimization problem of EXPRD – this quantity should be non-negative to ensure that none of the optimal policies w.r.t. $\tilde{R}$ is suboptimal w.r.t. $\tilde{R}$.

In Figure 6, we compare the designed reward functions w.r.t. these different criteria. In the “Sparseness” column, the quantity $|\text{supp}(\tilde{R})|$ is $B + 1$ for $\tilde{R}_{\text{RAFT}(B=5)}$, $\tilde{R}_{\text{EXPDRD}(B=5, \lambda \to \infty)}$, $\tilde{R}_{\text{EXPDRD}(B=3, \lambda=0)}$, and $\tilde{R}_{\text{EXPDRD}(B=5, \lambda=0)}$ as the goal states $\mathcal{G}$ are included in the design. In the “Invariance property” columns, we see that $\tilde{R}_{\text{RAFT}(B=5)}$ fails to satisfy the invariance property highlighting the well-known “reward bugs” that can arise in this approach and mislead the agent into learning suboptimal policies (see Section 2 and [2, 3]); this issue is further emphasized in the “Convergence” columns for $\tilde{R}_{\text{RAFT}(B=5)}$, highlighting that the agent is stuck with a suboptimal policy.

The last three columns related to “Convergence” highlight that the informativeness criteria we use in the optimization problem is a useful indicator about the agent’s convergence when learning from designed reward functions. Furthermore, EXPRD can provide an effective trade-off in sparseness and informativeness while maintaining invariance property and speed up the agent’s convergence. Even for small budgets of $B = 3$ or $B = 5$, the reward functions $\tilde{R}_{\text{EXPDRD}(3, \lambda=0)}$ and $\tilde{R}_{\text{EXPDRD}(5, \lambda=0)}$ lead to substantial speedups in the agent’s convergence in contrast to the original reward function $\tilde{R}$. Figures 7f and 7g further highlights that the states picked by EXPRD are important – the Algorithm 1 automatically picked the “gates” in the design process.

| Reward $\tilde{R}$ | Sparseness $|\text{supp}(\tilde{R})|$ | Invariance property $Q^\pi_\infty(s, \pi^*(s)) - Q^\pi_\infty(s, \pi^*(s))$ | Convergence: #Episodes to % value $\%$ of the total expected reward | |
|-------------------|----------------|--------------------------------------------------|--------------------------------------------------| |
| $\tilde{R}_{\text{RAFT}(B=5)}$ | $B + 1$ | $-0.1557$ | $1.688$ | $6.752$ | $20.570$ | |
| $\tilde{R}_{\text{EXPDRD}(B=5, \lambda \to \infty)}$ | $B + 1$ | $-0.1122$ | $1010$ | $\infty$ | $\infty$ | |
| $\tilde{R}_{\text{EXPDRD}(B=3, \lambda=0)}$ | $B + 1$ | $-0.0797$ | $35$ | $79$ | $146$ | |
| $\tilde{R}_{\text{EXPDRD}(B=5, \lambda=0)}$ | $B + 1$ | $-0.0107$ | $49$ | $773$ | $14,252$ | |

Apart from $B$ and $\lambda$, EXPRD requires us to specify $\Pi^\perp$, $\mathcal{H}$, and $I(\tilde{R})$. For the results reported in Figures 3 and 6, we use the following parameter choices for EXPRD: $\mathcal{H} = \{1, 4, 8, 16, 32\}$, $I(\tilde{R})$ is given by Eq. 15, and the set $\Pi^\perp$ contains only one policy from $\tilde{\Pi}$. Later in this section, we also consider variations of $\mathcal{H}$ and $I(\tilde{R})$, and report additional results in Figures 9 and 10.

### Invariance property

The notion $\mathcal{H}$ captures the invariance property w.r.t. criteria of sparseness, invariance, informativeness, and convergence. Here, the invariance property is captured through two different notions stated in Eq. 13 and Eq. 14 (a negative value represents a violation in the invariance property). Convergence is measured w.r.t. the number of episodes needed to get a specific % of the total expected reward, and are based on the convergence results in Figure 3a.
Visualizations of the designed reward functions. Figure 7 below shows a visualization of the eight different designed reward functions – this visualization is a variant of the visualization shown in Figure 3, where only three reward functions were shown.

Figure 7: Results for ROOMSNAVEnv. These plots show visualization of different designed reward functions discussed in Figure 6 – this visualization is a variant of the visualization shown in Figure 3 where only three reward functions were shown. For each of the reward functions, the first plot titled $R(s, \cdot) \neq 0$ shows which states have a non-zero reward assigned to at least one action and are marked with Gray color. The next four plots titled $R(s, \text{"up"})$, $R(s, \text{"left"})$, $R(s, \text{"down"})$, $R(s, \text{"right"})$ show rewards assigned to each state/action: here, a negative reward is shown in Red color with sign “−”, a positive reward is shown in Blue color with sign “+” and zero reward is shown in white. The magnitude of the reward is indicated by Red or Blue color intensity (see color representation in Figure 3).
Results w.r.t. variations in \(I(R)\). For the results reported in Figures 3 and 9a, we fix \(\mathcal{H} = \{1, 4, 8, 16, 32\}\), the set \(\Pi^*\) contains only one policy from \(\Pi^*\), and we use the following functional form for \(I(R)\) corresponding to the negated hinge loss:

\[
I_1(R) := \frac{1}{|\Pi^*| \cdot |\mathcal{H}| \cdot |S|} \sum_{\pi^* \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s \in S} \max_{a \in A^{\pi^*}_s} \left(-\max(0, \delta^*_{\mathcal{H}}(s) - \delta^*_h(s, a))\right)
\]

(15)

Here, we perform additional experiments to understand the effect of variations in \(I(R)\) on the reward functions designed by EXPRD. In Figures 9b, 9c, and 9d, we consider the following different functional forms of \(I(R)\) corresponding to the negated hinge loss, respectively:

\[
I_2(R) := \frac{1}{|\Pi^*| \cdot |\mathcal{H}| \cdot |S|} \sum_{\pi^* \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s \in S} \max_{a \in A^{\pi^*}_s} \left(-\max(0, \delta^*_{\mathcal{H}}(s) - \delta^*_h(s, a))\right)
\]

(16)

\[
I_3(R) := \frac{1}{|\Pi^*| \cdot |\mathcal{H}| \cdot |S|} \sum_{\pi^* \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s \in S} \sum_{a \in A^{\pi^*}_s} \left(-\max(0, \delta^*_{\mathcal{H}}(s) - \delta^*_h(s, a))\right)
\]

(17)

\[
I_4(R) := \frac{1}{|\Pi^*| \cdot |\mathcal{H}| \cdot |S|} \sum_{\pi^* \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s \in S} \sum_{a \in A^{\pi^*}_s} \left(-\max(0, \delta^*_{\mathcal{H}}(s) - \delta^*_h(s, a))\right)
\]

(18)

Finally, in Figures 9e and 9f, we use the following different functional forms of \(I(R)\) corresponding to the linear and negated exponential functions (instead of negated hinge loss), respectively:

\[
I_5(R) := \frac{1}{|\Pi^*| \cdot |\mathcal{H}| \cdot |S|} \sum_{\pi^* \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s \in S} \sum_{a \in A^{\pi^*}_s} \left(-\delta^*_{\mathcal{H}}(s, a) - \delta^*_h(s, a)\right)
\]

(19)

\[
I_6(R) := \frac{1}{|\Pi^*| \cdot |\mathcal{H}| \cdot |S|} \sum_{\pi^* \in \Pi^*} \sum_{h \in \mathcal{H}} \sum_{s \in S} \sum_{a \in A^{\pi^*}_s} \left(-\exp(\delta^*_{\mathcal{H}}(s, a) - \delta^*_h(s, a))\right)
\]

(20)

Additionally, we report results by varying the choice of the set \(\mathcal{H}\). More concretely, in Figure 10, we fix the functional form of \(I(R)\) as given Eq. 15, the set \(\Pi^*\) is same as above, and we vary the set \(\mathcal{H}\) as follows: \(\{1, 4, 8, 16, 32\}\), \(\{1, 2, \ldots, 19, 20\}\), and \(\{10, 11, \ldots, 19, 20\}\). Note that the value 20 corresponds to \(\frac{1}{1-\gamma}\).

All the results in this section are reported as an average over 40 runs and convergence plots show mean with standard error bars. Overall, the convergence behavior in Figures 9 and 10 suggests that the reward functions designed by our EXP RD framework are effective under different functional forms of \(I(R)\) and different choices of the set \(\mathcal{H}\).

Run times for a varying number of states and actions. Here, we report the run times for solving an instance of the optimization problem (P1) when set \(Z\) is fixed. In order to easily vary the number of states \(|S|\) as well as the number of actions \(|A|\), we consider a simple chain navigation environment where an agent can take “left” or “right” actions to navigate across the states (think of this as a one-dimensional variant of ROOMSNAVENV). To increase \(|A|\) beyond size 2, we added dummy actions which keep the agent’s location unchanged. For reporting the run times, we consider \(|\Pi^*| = 1, \mathcal{H} = \{1, 4, 8, 16, 32\}\), and vary \(|S|\) as well as \(|A|\). These run times are reported when solving the formulation of the optimization problem in terms of matrices as shown in Section 2. Numbers are reported in seconds and are based on an average of 5 runs for each setting. These run times are obtained by running the computation on a laptop machine with 2.3 GHz Quad-Core Intel Core i5 processor and 16 GB RAM. Overall, these run times are of the same order as that of solving an optimization problem instance in environment poisoning attacks reported in the literature (see [19] and Section 5).

| \(|A|\) | \(|S|\) | 25    | 50    | 75    | 100   | 125   | 150   | 175   | 200   |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 2     |       | 0.42s | 0.91s | 1.63s | 2.35s | 3.22s | 4.34s | 6.42s | 7.62s |
| 5     |       | 1.11s | 3.04s | 6.73s | 13.48s| 26.89s| 51.52s| 102.22s| 335.38s|

Figure 8: Run times for solving an instance of the optimization problem (P1) as we vary \(|S|\) and \(|A|\).
Figure 9: Results for ROOMSNAVENV. The plots show convergence in performance of the agent w.r.t. training episodes. Here, performance is measured as the expected reward per episode computed using $R$; note that the x-axis is exponential in scale. As the parameter choices for ExPRD, we use $\mathcal{H} = \{1, 4, 8, 16, 32\}$ and the set $\Pi^\dagger$ contains only one policy from $\Pi^*$. Each plot is obtained for a different functional form of $I(R)$.

Figure 10: Results for ROOMSNAVENV. The plots show convergence in performance of the agent w.r.t. training episodes. Here, performance is measured as the expected reward per episode computed using $R$; note that the x-axis is exponential in scale. As the parameter choices for ExPRD, we use $I(R)$ from Eq. 15 and the set $\Pi^\dagger$ contains only one policy from $\Pi^*$. Each plot is obtained for a different choice of $\mathcal{H}$. Note that Figure 10a is same as Figure 9a.
In this appendix, we expand on Section 4.2 and provide a more detailed description of the setup as well as additional results. Full implementation of our techniques is available in a Github repo as mentioned in Footnote 1.

### Additional details for the techniques evaluated
Below, we describe different reward design techniques along with hyperparameters that are evaluated in this section. More concretely, we have:

1. $\hat{R}_{\text{ORIG}}$ simply represents the default reward function $\overline{R}$.
2. $\hat{R}_{\text{PBRS}}$ is obtained via the PBRS technique based on Eq. 1 and using an abstraction (see Section 3.5, [35]). We first define an abstraction $\phi : S \rightarrow \mathcal{X}_\phi$ as described below after this list. Based on this abstraction $\phi$, we construct an abstract MDP $\overline{M}_\phi$ using the original MDP $M$, and compute the optimal state value function $V_{\phi,\infty}$ in the abstract MDP $\overline{M}_\phi$. Finally, we lift $V_{\phi,\infty}$ to the original state space $S$ (see Appendix D), and use the lifted value function as the potential function for the PBRS.
3. $\hat{R}_{\text{PBRS-ABS}}$ is a variant of $\hat{R}_{\text{PBRS}}$. Similar to $\hat{R}_{\text{PBRS}}$, we compute the optimal state value function $V_{\phi,\infty}$ in the abstract MDP $\overline{M}_\phi$. We use this value function as the potential function for the PBRS to design $\hat{R}_{\text{PBRS,}\phi}$ in the MDP $M_\phi$. Finally, we lift $\hat{R}_{\text{PBRS,}\phi}$ to the original state space $S$ (see Appendix D). Note that $\hat{R}_{\text{PBRS-ABS}}$ is not guaranteed to satisfy the invariance property of $\hat{R}_{\text{PBRS}}$.
4. $\hat{R}_{\text{EXPDRD}(B,\lambda=0)}$ is the reward function designed by our pipeline in Section 3.5 that relies on our EXPDRD framework and an abstraction. We use the same abstraction $\phi : S \rightarrow \mathcal{X}_\phi$ for all the techniques and is described below after this list. In the subroutine, we run EXPDRD on $\overline{M}_\phi$ for a budget $B = 5$ and a full budget $B = |\mathcal{X}_\phi|$; we set $\lambda = 0$. We denote the designed reward functions as $\hat{R}_{\text{EXPDRD}(B=5,\lambda=0)}$ and $\hat{R}_{\text{EXPDRD}(B=|\mathcal{X}_\phi|,\lambda=0)}$. Similar to Figure 9a, we fix $\mathcal{H} = \{1, 4, 8, 16, 32\}$, and we use the functional form given in Eq. 15 for $I(R)$. Here, we describe the abstraction $\phi$ used for computing $\hat{R}_{\text{PBRS}}, \hat{R}_{\text{PBRS-ABS}}$, and $\hat{R}_{\text{EXPDRD}(B,\lambda=0)}$.

Recall the description of the original MDP $M$ from Section 4.2 – the state corresponds to the agent’s status comprising of the current location (a point x in $[0, 1]$) and a binary flag whether the agent has acquired a key. For a given hyperparameter $\alpha \in (0, 1)$, we obtain a finite set of locations $X$ by $\alpha$-level discretization of the line segment $[0, 1]$, leading to a $1/\alpha$ number of locations. For the abstraction $\phi$ associated with this discretization, the abstract MDP $M_\phi$ has $|\mathcal{X}_\phi| = 2/\alpha$ corresponding to $1/\alpha$ locations and a binary flag for the key. We use $\alpha = 0.05$ in the experiments.

#### Results for Q-learning agent with 0.01-level location discretization.
For the results reported in the main paper (Figure 5a) and in Figure 11a, the agent uses Q-learning method in a discretized version of the original MDP $\overline{M}$ with a 0.01-level discretization of the location (i.e., the number of states in the agent’s discretized MDP is 200). The rest of the method’s parameters are same as in Section 4.1, i.e., 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 $\overline{R}$, 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. For this agent, the convergence results are reported in Figure 11a as an average over 40 runs. These results demonstrate that all four designed reward functions—$\hat{R}_{\text{PBRS}}, \hat{R}_{\text{PBRS-ABS}}, \hat{R}_{\text{EXPDRD}(B=5,\lambda=0)}, \hat{R}_{\text{EXPDRD}(B=|\mathcal{X}_\phi|,\lambda=0)}$—substantially improves the convergence, whereas the agent is not able to learn under $\overline{R}_{\text{ORIG}}$.

#### Results for Q-learning agent with 0.005-level location discretization.
Here, we demonstrate that our abstraction based pipeline in Section 3.5 is robust to the state representation used by the agent. In particular, for the results reported in Figure 11b, the agent uses a discretized version of the original MDP $\overline{M}$ with a 0.005-level discretization of the location. As in the setting above, the agent uses Q-learning method in this discretized version of the original MDP $\overline{M}$. Similar to Figure 11a, Figure 11b demonstrates that the performance associated with all four designed reward functions—$\hat{R}_{\text{PBRS}}, \hat{R}_{\text{PBRS-ABS}}, \hat{R}_{\text{EXPDRD}(B=5,\lambda=0)}, \hat{R}_{\text{EXPDRD}(B=|\mathcal{X}_\phi|,\lambda=0)}$—substantially improves the convergence in contrast to $\overline{R}_{\text{ORIG}}$. 

---

28
Results for REINFORCE agent with continuous location representation. For the results reported in Figure 11c, the agent uses the REINFORCE policy gradient method (see \cite{7, 62}) in the original MDP $M$ with continuous representation of the location. We use a neural network to learn the policy, which takes a continuous value in $[0, 1]$ (the location) and a binary flag (whether the agent has acquired a key) as the input representing a state $s$. The neural network has a hidden layer with 256 nodes. Given a state $s$ (the input to the network), the policy network outputs three scores for three different actions. Then, applying softmax operation over these three scores gives the policy’s action distribution. We use the REINFORCE method with a learning rate 0.0005. The gradient update happens at the end of each episode. In contrast to the maximum episode length of 50 used by Q-learning agents, we set this to 150 for the REINFORCE agent.

Figure 11c shows convergence results for this agent as an average over 20 runs; for each individual run, we additionally applied a moving-window average over a window size of 100 episodes. With neural representation for states, the policy invariance might not hold anymore. However, Figure 11c demonstrates that all four designed reward functions—$\tilde{R}_{\text{PBRS}}$, $\tilde{R}_{\text{PBRS-Abs}}$, $\tilde{R}_{\text{EXPRD}(B=5, \lambda=0)}$, $\tilde{R}_{\text{EXP}(B=|X_\phi|, \lambda=0)}$—substantially improves the convergence (slightly weaker compared to Figures 11a and 11b), whereas the agent is not able to learn under $\tilde{R}_{\text{ORIG}}$. This observation highlights our pipeline in Section 3.5 as a promising approach for reward design in high-dimensional settings. As future work, we plan to (both theoretically and empirically) investigate the effectiveness of the reward functions designed by EXPRD or its adaptions in accelerating the learning process in high-dimensional settings for policy gradient methods.

Visualizations of the designed reward functions. Figure 12 shows visualization of the five different designed reward functions discussed above – this visualization is a variant of the visualization shown in Figure 5 where only three reward functions were shown. This visualization provides important insights into the reward functions designed by EXPRD. Interestingly, $\tilde{R}_{\text{EXP}(B=5, \lambda=0)}$ assigned a high positive reward for the “pick” action when the agent is in the locations with key (see $R((x, -), \ (“pick”) \) bar in Figure 12d).
Figure 12: Results for LINKEYNAVENV. These plots show visualization of the five different designed reward functions discussed above – this visualization is a variant of the visualization shown in Figure 5 where only three reward functions were shown. For each of the reward functions, we show a total of 8 horizontal bars. Denoting a state as tuple (x, −) (i.e., location x when the key has not been picked) or (x, key) (i.e., location x when the key has been picked), these 8 horizontal bars have the following interpretation. The two bars, titled $R((x, -), \cdot) \neq 0$ and $R((x, \text{key}), \cdot) \neq 0$, indicate states in Gray color for which a non-zero reward is assigned to at least one action; in these two bars, we have further highlighted the segment $[0.9, 1]$ with the goal, and the segment $[0.1, 0.2]$ with the key. The remaining six bars, titled $R((x, -), \text{left})$, $R((x, -), \text{right})$, $R((x, -), \text{pick})$, $R((x, \text{key}), \text{left})$, $R((x, \text{key}), \text{right})$, and $R((x, \text{key}), \text{pick})$, show rewards assigned to each state/action: here, a negative reward is shown in Red color, a positive reward is shown in Blue color, and zero reward is shown in white. The magnitude of the reward is indicated by Red or Blue color intensity and we use the same color representation as in Figure 5.