Informativeness of Reward Functions in Reinforcement Learning

Rati Devidze
MPI-SWS
Saarbrücken, Germany
rdevidze@mpi-sws.org

Parameswaran Kamalaruban
The Alan Turing Institute
London, United Kingdom
pkamalaruban@gmail.com

Adish Singla
MPI-SWS
Saarbrücken, Germany
adishs@mpi-sws.org

ABSTRACT

Reward functions are central in specifying the task we want a reinforcement learning agent to perform. Given a task and desired optimal behavior, we study the problem of designing informative reward functions so that the designed rewards speed up the agent's convergence. In particular, we consider expert-driven reward design settings where an expert or teacher seeks to provide informative and interpretable rewards to a learning agent. Existing works have considered several different reward design formulations; however, the key challenge is formulating a reward informativeness criterion that adapts w.r.t. the agent's current policy and can be optimized under specified structural constraints to obtain interpretable rewards. In this paper, we propose a novel reward informativeness criterion, a quantitative measure that captures how the agent's current policy will improve if it receives rewards from a specific reward function. We theoretically showcase the utility of the proposed informativeness criterion for adaptively designing rewards for an agent. Experimental results on two navigation tasks demonstrate the effectiveness of our adaptive reward informativeness criterion.

KEYWORDS

Reinforcement Learning; Reward Design; Reward Informativeness

ACM Reference Format:
Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. 2024. Informativeness of Reward Functions in Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reward functions play a central role during the learning/training process of a reinforcement learning (RL) agent. Given a task the agent is expected to perform, many different reward functions exist under which an optimal policy has the same performance on the task. This freedom in choosing a reward function for the task, in turn, leads to the fundamental question of designing appropriate rewards for the RL agent that match certain desired criteria [28, 31, 37]. In this paper, we study the problem of designing *informative* reward functions so that the designed rewards speed up the agent's convergence [2, 8, 25, 28, 31, 37].

More concretely, we focus on expert-driven reward design settings where an expert or teacher seeks to provide informative rewards to a learning agent [13, 16, 21, 26, 28, 31, 35, 36, 45, 46]. In expert-driven reward design settings, the designed reward functions should also satisfy certain *structural constraints* apart from being informative, e.g., to ensure interpretability of reward signals or to match required reward specifications [5, 7, 11, 13, 18, 19, 21, 22]. For instance, informativeness and interpretability become crucial in settings where rewards are designed for human learners who are learning to perform sequential tasks in pedagogical applications such as educational games [33] and open-ended problem solving domains [27]. Analogously, informativeness and structural constraints become crucial in settings where rewards are designed for complex compositional tasks in the robotics domain that involve reward specifications in terms of automata or subgoals [19, 21]. To this end, an important research question is: *How to formulate reward informativeness criterion that can be optimized under specified structural constraints?*

Existing works have considered different reward design formulations; however, they have limitations in appropriately incorporating informativeness and structural properties. On the one hand, potential-based reward shaping (PBRS) is a well-studied family of reward design techniques [3, 11, 14, 16–18, 21, 31, 43]. While PBRS techniques enable designing informative rewards via utilizing informative potential functions (e.g., near-optimal value function for the task), the resulting reward functions do not adhere to specific structural constraints. On the other hand, optimization-based reward design techniques is another popular family of techniques [13, 26, 35, 36, 45, 46]. While optimization-based techniques enable enforcing specific structural constraints, there is a lack of suitable reward informativeness criterion that is amenable to optimization as part of these techniques. In this family of techniques, a recent work [13] introduced a reward informativeness criterion suitable for optimization under sparseness structure; however, their informativeness criterion doesn't account for the agent's current policy, making the reward design process agnostic to the agent's learning progress.

In this paper, we present a general framework, ExpAdaRD, for *expert-driven explicable and adaptive reward design.* ExpAdaRD utilizes a novel reward informativeness criterion, a quantitative measure that captures how the agent's current policy will improve if it receives rewards from a specific reward function. Crucially, the informativeness criterion adapts w.r.t. the agent's current policy and can be optimized under specified structural constraints to obtain interpretable rewards. Our main results and contributions are:

I. We introduce a reward informativeness criterion formulated within bi-level optimization. By analyzing it for a specific learning algorithm, we derive a novel informativeness criterion that is amenable to the reward optimization process (Sections 4.1 and 4.2).
II. We theoretically showcase the utility of our informativeness criterion in adaptively designing rewards by analyzing the convergence speed up of an agent in a simplified setting (Section 4.3).

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

III. We empirically demonstrate the effectiveness of our reward informativeness criterion for designing explicable and adaptive reward functions in two navigation environments. (Section 5).[1]

1.1 Related Work

Expert-driven reward design. As previously discussed, well-studied families of expert-driven reward design techniques include potential-based reward shaping (PBRS) [3, 11, 14, 16–18, 21, 31, 43], optimization-based techniques [13, 26, 35, 36, 45, 46], and reward shaping with expert demonstrations or feedback [6, 9, 10, 44]. Our reward design framework, ExpAdaRD, also uses an optimization-based design process. The key issue with existing optimization-based techniques is a lack of suitable reward informativeness criterion. A recent work [13] introduced an expert-driven explicable reward design framework (ExpRD) that optimizes an informativeness criterion under sparseness structure. However, their informativeness criterion doesn't account for the agent's current policy, making the reward design process agnostic to the agent's learning progress. In contrast, we propose an adaptive informativeness criterion enabling it to provide more informative reward signals. Technically, our proposed reward informativeness criterion is quite different from that proposed in [13] and is derived based on analyzing meta-gradients within bi-level optimization formulation.

Learner-driven reward design. Learner-driven reward design techniques involve an agent designing its own rewards throughout the training process to accelerate convergence [2, 4, 12, 15, 24, 30, 40, 42, 48]. These learner-driven techniques employ various strategies, including designing intrinsic rewards based on exploration bonuses [4, 24, 47], crafting rewards using domain-specific knowledge [42], using credit assignment to create intermediate rewards [2, 15], and designing parametric reward functions by iteratively updating reward parameters and optimizing the agent's policy based on learned rewards [12, 30, 40, 48]. While these learner-driven techniques are typically designing adaptive and online reward functions, these techniques do not emphasize the formulation of an informativeness criterion explicitly. In our work, we draw on insights from meta-gradient derivations presented in [12, 30, 40, 48] to develop an adaptive informativeness criterion tailored for the expert-driven reward design settings.

2 PRELIMINARIES

Environment. An environment is defined as a Markov Decision Process (MDP) denoted by $M := (\mathcal{S}, \mathcal{A}, T, P_0, \gamma, R)$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces respectively. The state transition dynamics are captured by $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $T(s' \mid s, a)$ denotes the probability of transitioning to state s' by taking action a from state s. The discounting factor is denoted by γ, and P_0 represents the initial state distribution. The reward function is given by $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Policy and performance. We denote a stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ as a mapping from a state to a probability distribution over actions, and a deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ as a mapping from a state to an action. For any trajectory $\xi = \{(s_t, a_t)\}_{t=0,1,\ldots,H}$, we define its cumulative return with respect to reward function R as $J(\xi, R) := \sum_{t=0}^{H} \gamma^t \cdot R(s_t, a_t)$. The expected cumulative return (value) of a policy π with respect to R is then defined as $J(\pi, R) := \mathbb{E}\left[J(\xi, R) | P_0, T, \pi\right]$, where $s_0 \sim P_0(\cdot)$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim T(\cdot|s_t, a_t)$. A learning agent (learner) in our setting seeks to find a policy that has maximum value with respect to R, i.e., $\max_\pi J(\pi, R)$. We denote the state occupancy measure of a policy π by d^π. Furthermore, we define the state value function V^π_R and the action value function Q^π_R of a policy π with respect to R as follows, respectively: $V^\pi_R(s) = \mathbb{E}[J(\xi, R)|s_0 = s, T, \pi]$ and $Q^\pi_R(s, a) = \mathbb{E}[J(\xi, R)|s_0 = s, a_0 = a, T, \pi]$. The optimal value functions are given by $V^*_R(s) = \sup_\pi V^\pi_R(s)$ and $Q^*_R(s, a) = \sup_\pi Q^\pi_R(s, a)$.

Algorithm 1 A General Framework for Expert-driven Explicable and Adaptive Reward Design (ExpAdaRD)

```
Input: MDP $M := (\mathcal{S}, \mathcal{A}, T, P_0, \gamma, \overline{R})$, target policy $\pi^T$, learning algorithm $L$, informativeness criterion $I_L$, reward constraint set $\mathcal{R}$
Initialize: learner's initial policy $\pi^L_0$
for $k = 1, 2, \ldots, K$ do
    Expert/teacher updates the reward function: $R_k \leftarrow \arg\max_{R \in \mathcal{R}} I_L(R \mid \overline{R}, \pi^T, \pi^L_{k-1})$
    Learner updates the policy: $\pi^L_k \leftarrow L(\pi^L_{k-1}, R_k)$
Output: learner's policy $\pi^L_K$
```

3 EXPERT-DRIVEN EXPLICABLE AND ADAPTIVE REWARD DESIGN

In this section, we present a general framework for expert-driven reward design, ExpAdaRD, as outlined in Algorithm 1. In our framework, an expert or teacher seeks to provide informative and interpretable rewards to a learning agent. In each round k, we address a reward design problem involving the following key elements: an underlying reward function $\overline{R}$, a target policy π^T (e.g., a near-optimal policy w.r.t. $\overline{R}$), a learner's policy π^L_{k-1}, and a learning algorithm L. The main objective of this reward design problem is to craft a new reward function R_k under constraints $\mathcal{R}$ such that R_k provides informative learning signals when employed to update the policy π^L_{k-1} using the algorithm L. To quantify this objective, it is essential to define a reward informativeness criterion, $I_L(R \mid \overline{R}, \pi^T, \pi^L_{k-1})$, that adapts w.r.t. the agent's current policy and can be optimized under specified structural constraints to obtain interpretable rewards. Given this informativeness criterion I_L (to be developed in Section 4), the reward design problem can be formulated as follows:

$$\max_{R \in \mathcal{R}} I_L(R \mid \overline{R}, \pi^T, \pi^L_{k-1}). \tag{1}$$

Here, the set $\mathcal{R}$ encompasses additional constraints tailored to the application-specific requirements, including (i) policy invariance constraints $\mathcal{R}_{\text{inv}}$ to guarantee that the designed reward function induces the desired target policy and (ii) structural constraints $\mathcal{R}_{\text{str}}$ to obtain interpretable rewards, as further discussed below.

Invariance constraints. Let $\overline{\Pi}^* := \{\pi : \mathcal{S} \to \mathcal{A} \text{ s.t. } V^\pi_{\overline{R}}(s) = V^*_{\overline{R}}(s), \forall s \in \mathcal{S}\}$ denote the set of all deterministic optimal policies

[1] Github: https://github.com/machine-teaching-group/aamas2024-informativeness-of-reward-functions.

under $\overline{R}$. Next, we define $\mathcal{R}_{\text{inv}}$ as a set of invariant reward functions, where each $R \in \mathcal{R}_{\text{inv}}$ satisfies the following conditions [13, 31]:

$$Q^{\pi^T}_{R}(s,a) - V^{\pi^T}_{R}(s) \;\leq Q^{\pi^T}_{\overline{R}}(s,a) - V^{\pi^T}_{\overline{R}}(s), \quad \forall a \in \mathcal{A}, s \in \mathcal{S}.$$

When π^T is an optimal policy under $\overline{R}$ (i.e., $\pi^T \in \overline{\Pi}^*$), these conditions guarantee the following: (i) π^T is an optimal policy under R; (ii) any optimal policy induced by R is also an optimal policy under $\overline{R}$; (iii) reward function $\overline{R} \in \mathcal{R}_{\text{inv}}$, i.e., $\mathcal{R}_{\text{inv}}$ is non-empty.[2]

Structural constraints. We consider structural constraints as a way to obtain interpretable rewards (e.g., sparsity or tree-structured rewards) and satisfy application-specific requirements (e.g., bounded rewards). We denote the set of reward functions conforming to specified structural constraints as $\mathcal{R}_{\text{str}}$ [5, 7, 11, 13, 18, 19, 21, 22]. We implement these constraints via a set of parameterized reward functions, denoted as $\mathcal{R}_{\text{str}} = \{R_\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \text{ where } \phi \in \mathbb{R}^d\}$. For example, given a feature representation $f : \mathcal{S} \times \mathcal{A} \rightarrow \{0,1\}^d$, we employ $R_\phi(s,a) = \langle \phi, f(s,a) \rangle$ in our experimental evaluation (Section 5). In particular, we will use different feature representations to specify constraints induced by coarse-grained state abstraction [23] and tree structure [5]. Furthermore, it is possible to impose additional constraints on ϕ, such as bounding its ℓ_∞ norm by $R_{\max}$ or requiring that its support $\text{supp}(\phi)$, defined as $\{i : i \in [d], \phi_i \neq 0\}$, matches a predefined set $\mathcal{Z} \subseteq [d]$ [13].

4 INFORMATIVENESS CRITERION FOR REWARD DESIGN

In this section, we focus on developing a reward informativeness criterion that can be optimized for the reward design formulation in Eq. (1). We first introduce an informativeness criterion formulated within a bi-level optimization framework and then propose an intuitive informativeness criterion that can be generally applied to various learning algorithms.

Notation. In the subscript of the expectations $\mathbb{E}$, let $\pi(a|s)$ mean $a \sim \pi(\cdot|s)$, $\mu^\pi(s,a)$ mean $s \sim d^\pi, a \sim \pi(\cdot|s)$, and $\mu^\pi(s)$ mean $s \sim d^\pi$. Further, we use shorthand notation $\mu^\pi_{s,a}$ and μ^π_s to refer $\mu^\pi(s,a)$ and $\mu^\pi(s)$, respectively.

4.1 Bi-Level Formulation for Reward Informativeness $I_L(R)$

We consider parametric reward functions of the form $R_\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, where $\phi \in \mathbb{R}^d$, and parametric policies of the form $\pi_\theta : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where $\theta \in \mathbb{R}^n$. Let $\overline{R}$ be the underlying reward function, and let π^T be a target policy (e.g., a near-optimal policy w.r.t. $\overline{R}$). We measure the performance of any policy π_θ w.r.t. $\overline{R}$ and π^T using the following performance metric: $J(\pi_\theta; \overline{R}, \pi^T) = \mathbb{E}_{\mu^{\pi^T}_s}\left[\mathbb{E}_{\pi_\theta(a|s)}\left[A^{\pi^T}_{\overline{R}}(s,a)\right]\right]$, where $A^{\pi^T}_{\overline{R}}(s,a) = Q^{\pi^T}_{\overline{R}}(s,a) - V^{\pi^T}_{\overline{R}}(s)$ is the advantage function of policy π^T w.r.t. $\overline{R}$. Given a current policy π_θ and a reward function R, the learner updates the policy parameter using a learning algorithm L as follows: $\theta_{\text{new}} \leftarrow L(\theta, R)$.

To evaluate the informativeness of a reward function R_ϕ in guiding the convergence of the learner's policy $\pi^L := \pi_{\theta^L}$ towards the target policy π^T, we define the following informativeness criterion:

$$\begin{aligned} I_L(R_\phi \mid \overline{R}, \pi^T, \pi^L) \; &:= J(\pi_{\theta^L_{\text{new}}(\phi)}; \overline{R}, \pi^T) \\ &\qquad \text{where} \quad \theta^L_{\text{new}}(\phi) \leftarrow L(\theta^L, R_\phi). \end{aligned} \tag{2}$$

The above criterion measures the performance of the resulting policy after the learner updates π^L using the reward function R_ϕ. However, this criterion relies on having access to the learning algorithm L and evaluating this criterion requires potentially expensive policy updates using L. In the subsequent analysis, we further examine this criterion to develop an intuitive alternative that is independent of any specific learning algorithm and does not require any policy updates for its evaluation.

Analysis for a specific learning algorithm L. Here, we present an analysis of the informativeness criterion defined above, considering a simple learning algorithm L. Specifically, we consider an algorithm L that utilizes parametric policies $\{\pi_\theta : \theta \in \mathbb{R}^n\}$ and performs single-step vanilla policy gradient updates using Q-values computed using h-depth planning [12, 40, 48]. We update the policy parameter θ by employing a reward function R in the following manner:

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

where $Q^{\pi_\theta}_{R,h}(s,a) = \mathbb{E}\left[\sum_{t=0}^{h} \gamma^t R(s_t,a_t) \middle| s_0 = s, a_0 = a, T, \pi_\theta\right]$ is the h-depth Q-value with respect to R, and α is the learning rate. Furthermore, we assume that L uses a tabular representation, where $\theta \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$, and a softmax policy parameterization given by $\pi_\theta(a|s) := \frac{\exp(\theta(s,a))}{\sum_b \exp(\theta(s,b))}, \forall s \in \mathcal{S}, a \in \mathcal{A}$. For this L, the following proposition provides an intuitive form of the gradient of I_L in Eq. (2).

Proposition 1. *The gradient of the informativeness criterion in Eq.* (2) *for the simplified learning algorithm L with h-depth planning described above takes the following form:*

$$\nabla_\phi I_L(R_\phi \mid \overline{R}, \pi^T, \pi^L) \;\approx$$

$$\alpha \cdot \nabla_\phi \mathbb{E}_{\mu^{\pi^L}_{s,a}}\left[\mu^{\pi^T}_s \cdot \pi^L(a|s) \cdot \left(A^{\pi^T}_{\overline{R}}(s,a) - A^{\pi^T}_{\overline{R}}(s,\pi^L(s))\right) \cdot A^{\pi^L}_{R_\phi,h}(s,a)\right],$$

where $A^{\pi^T}_{\overline{R}}(s,\pi^L(s)) = \mathbb{E}_{\pi^L(a'|s)}\left[A^{\pi^T}_{\overline{R}}(s,a')\right]$, *and* $A^{\pi^L}_{R_\phi,h}(s,a) = Q^{\pi^L}_{R_\phi,h}(s,a) - V^{\pi^L}_{R_\phi,h}(s)$.

Proof. We discuss key proof steps here and provide a more detailed proof in the longer version of the paper. For the simple learning algorithm L described above, we can write the derivative of the informativeness criterion in Eq. (2) as follows:

$$\begin{aligned} &\left[\nabla_\phi I_L(R_\phi \mid \overline{R}, \pi^T, \pi^L)\right]_\phi \\ &\qquad \overset{(a)}{=} \; \left[\nabla_\phi \theta^L_{\text{new}}(\phi) \cdot \nabla_{\theta^L_{\text{new}}(\phi)} J(\pi_{\theta^L_{\text{new}}(\phi)}; \overline{R}, \pi^T)\right]_\phi \\ &\qquad \overset{(b)}{\approx} \; \left[\nabla_\phi \theta^L_{\text{new}}(\phi)\right]_\phi \cdot \left[\nabla_\theta J(\pi_\theta; \overline{R}, \pi^T)\right]_{\theta^L}, \end{aligned}$$

where the equality in (a) is due to chain rule, and the approximation in (b) assumes a smoothness condition of $\Big\| \left[\nabla_\theta J(\pi_\theta; \overline{R}, \pi^T)\right]_{\theta^L_{\text{new}}(\phi)} -$

[2] We can guarantee point (i) of π^T being an optimal policy under R by replacing the right-hand side with $-\epsilon$ for $\epsilon > 0$; however, this would not guarantee (ii) and (iii).

$\left[\nabla_\theta J(\pi_\theta;\overline{R},\pi^T)\right]_{\theta^L}\Big\|_2 \leq c \cdot \left\|\theta^L_{\text{new}}(\phi) - \theta^L\right\|_2$ for some $c > 0$. For the L described above, we can obtain intuitive forms of the terms $\left[\nabla_\phi \theta^L_{\text{new}}(\phi)\right]_\phi$ and $\left[\nabla_\theta J(\pi_\theta;\overline{R},\pi^T)\right]_{\theta^L}$. For any $s \in \mathcal{S}, a \in \mathcal{A}$, let $\mathbf{1}_{s,a} \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ denote a vector with 1 in the (s,a)-th entry and 0 elsewhere. By using the meta-gradient derivations presented in [1, 32, 38], we simplify the first term as follows:

$$\left[\nabla_\phi \theta^L_{\text{new}}(\phi)\right]_\phi = \alpha \cdot \mathbb{E}_{\mu_s^{\pi^L}}\left[\sum_a \pi^L(a|s) \cdot \left[\nabla_\phi A^{\pi^L}_{R_\phi,h}(s,a)\right]_\phi \cdot \mathbf{1}^\top_{s,a}\right].$$

Then, we simplify the second term as follows:

$$\begin{aligned}&\left[\nabla_\theta J(\pi_\theta;\overline{R},\pi^T)\right]_{\theta^L}\\ &\quad= \mathbb{E}_{\mu_s^{\pi^T}}\left[\sum_a \pi^L(a|s) \cdot \left(A^{\pi^T}_{\overline{R}}(s,a) - A^{\pi^T}_{\overline{R}}(s,\pi^L(s))\right) \cdot \mathbf{1}_{s,a}\right].\end{aligned}$$

Taking the matrix product of two terms completes the proof. □

4.2 Intuitive Formulation for Reward Informativeness $I_h(R)$

Based on Proposition 1, for the simple learning algorithm L discussed in Section 4.1, the informativeness criterion in Eq. (2) can be written as follows:

$$\begin{aligned}I_L(R_\phi \mid \overline{R},\pi^T,\pi^L) \;\approx\; &\alpha \cdot \mathbb{E}_{\mu_{s,a}^{\pi^L}}\Big[\mu_s^{\pi^T} \cdot \pi^L(a|s)\\ &\cdot \big(A^{\pi^T}_{\overline{R}}(s,a) - A^{\pi^T}_{\overline{R}}(s,\pi^L(s))\big) \cdot A^{\pi^L}_{R_\phi,h}(s,a)\Big] + \kappa,\end{aligned}$$

for some $\kappa \in \mathbb{R}$. By dropping the constant terms α and κ, we define the following intuitive informativeness criterion:

$$\begin{aligned}&I_h(R_\phi \mid \overline{R},\pi^T,\pi^L) \;:=\\ &\mathbb{E}_{\mu_{s,a}^{\pi^L}}\Big[\mu_s^{\pi^T} \cdot \pi^L(a|s) \cdot \big(A^{\pi^T}_{\overline{R}}(s,a) - A^{\pi^T}_{\overline{R}}(s,\pi^L(s))\big) \cdot A^{\pi^L}_{R_\phi,h}(s,a)\Big].\end{aligned} \tag{3}$$

The above criterion doesn't require the knowledge of the learning algorithm L and only relies on π^L, $\overline{R}$, and π^T. Therefore, it serves as a generic informativeness measure that can be used to evaluate the usefulness of reward functions for a range of limited-capacity learners, specifically those with different h-horizon planning budgets. In practice, we use the criterion I_h with $h = 1$. In this case, the criterion simplifies to the following form:

$$\begin{aligned}I_{h=1}(R_\phi \mid \overline{R},\pi^T,\pi^L) \;:=\; &\mathbb{E}_{\mu_{s,a}^{\pi^L}}\Big[\mu_s^{\pi^T} \cdot \pi^L(a|s)\\ &\cdot \big(A^{\pi^T}_{\overline{R}}(s,a) - A^{\pi^T}_{\overline{R}}(s,\pi^L(s))\big) \cdot \big(R_\phi(s,a) - R_\phi(s,\pi^L(s))\big)\Big],\end{aligned}$$

where $R_\phi(s,\pi^L(s)) = \mathbb{E}_{\pi^L(b|s)}\left[R_\phi(s,b)\right]$. Intuitively, this criterion measures the alignment of a reward function R_ϕ with better actions according to policy π^T, and how well it boosts the reward values for these actions in each state.

4.3 Using $I_h(R)$ in ExpAdaRD Framework

Next, we will use the informativeness criterion I_h for designing reward functions to accelerate the training process of a learning agent within the ExpAdaRD framework. Specifically, we use I_h in place of I_L to address the reward design problem formulated in Eq. (1):

$$\max_{R_\phi \in \mathcal{R}} I_h(R_\phi \mid \overline{R},\pi^T,\pi^L_{k-1}), \tag{4}$$

where the set $\mathcal{R}$ captures the additional constraints discussed in Section 3 (e.g., $\mathcal{R} = \mathcal{R}_{\text{inv}} \cap \mathcal{R}_{\text{str}}$). In Section 5, we will implement ExpAdaRD framework with two types of structural constraints and design adaptive reward functions for different learners; below, we theoretically showcase the utility of using I_h by analyzing the improvement in the convergence in a simplified setting.

More concretely, we present a theoretical analysis of the reward design problem formulated in Eq. (4) without structural constraints and in a simplified setting to illustrate how this informativeness criterion for adaptive reward shaping can substantially improve the agent's convergence speed toward the target policy. For our theoretical analysis, we consider a finite MDP M, with the target policy π^T being an optimal policy for this MDP. We use a tabular representation for the reward, i.e., $\phi \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$. We consider a constraint set $\mathcal{R} = \{R : |R(s,a)| \leq R_{\max}, \forall s \in \mathcal{S}, a \in \mathcal{A}\}$. Additionally, we use the informativeness criterion in Eq. (3) with $h = 1$, i.e., $I_{h=1}(R_\phi \mid \overline{R},\pi^T,\pi^L)$. For the policy, we also use a tabular representation, i.e., $\theta \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$. We use a greedy (policy iteration style) learning algorithm L that first learns the h-step action-value function $Q^{\pi^L_{k-1}}_{R_k,h}$ w.r.t. current reward R_k and updates the policy by selecting actions greedily based on the value function, i.e., $\pi^L_k(s) \leftarrow \arg\max_a Q^{\pi^L_{k-1}}_{R_k,h}(s,a)$ with random tie-breaking. In particular, we consider a learner with $h = 1$, i.e., we have $\pi^L_k(s) \leftarrow \arg\max_a R_k(s,a)$. For the above setting, the following theorem provides a convergence guarantee for Algorithm 1.

Theorem 1. *Consider Algorithm 1 with inputs π^T, L, I_h, and $\mathcal{R}$ as described above. We define a policy $\pi^{T,\text{Adv}}$ induced by the advantage function of the target policy π^T (w.r.t. $\overline{R}$) as follows: $\pi^{T,\text{Adv}}(s) \leftarrow \arg\max_a A^{\pi^T}_{\overline{R}}(s,a)$ with random tie-breaking. Then, the learner's policy π^L_k will converge to the policy $\pi^{T,\text{Adv}}$ in $O(|\mathcal{A}|)$ iterations.*

Proof and additional details are provided in the longer version of the paper. We note that the target policy π^T does not need to be optimal for better convergence, and the results also hold with a sufficiently good (weak) target policy $\pi^{\widetilde{T}}$ s.t. $\pi^{\widetilde{T},\text{Adv}}$ is near-optimal.

5 EXPERIMENTAL EVALUATION

In this section, we evaluate our expert-driven explicable and adaptive reward design framework, ExpAdaRD, on two environments: Room (Section 5.1) and LineK (Section 5.2). Room corresponds to a navigation task in a grid-world where the agent has to learn a policy to quickly reach the goal location in one of four rooms, starting from an initial location. Even though this environment has small state and action spaces, it provides a rich problem setting to validate different reward design techniques. In fact, variants of Room have been used in the literature [3, 11–13, 18, 20, 21, 29, 39]. LineK corresponds to a navigation task in a one-dimensional space where the agent has to first pick the key and then reach the goal. The agent's location is represented as a node in a long chain. This environment is inspired by variants of navigation tasks in the literature where an agent needs to perform subtasks [12, 13, 31, 34]. Both the Room and LineK environments have sparse and delayed rewards, which pose a challenge for learning optimal behavior.

5.1 Evaluation on Room

Room (Figure 1a). This environment is based on the work of [13] that also serves as a baseline technique. The environment is represented as an MDP with $\mathcal{S}$ states corresponding to cells in a gridworld with the "blue-circle" indicating the agent's initial location. The goal ("green-star") is located at the top-right corner cell. Agent can take four actions given by $\mathcal{A} := \{\text{"up"}, \text{"left"}, \text{"down"}, \text{"right"}\}$. An action takes the agent to the neighbouring cell represented by the direction of the action; however, if there is a wall ("brown-segment"), the agent stays at the current location. There are also a few terminal walls ("thick-red-segment") that terminate the episode, located at the bottom-left corner cell, where "left" and "down" actions terminate the episode; at the top-right corner cell, "right" action terminates. The agent gets a reward of $R_{\max}$ after it has navigated to the goal and then takes a "right" action (i.e., only one state-action pair has a reward); note that this action also terminates the episode. The reward is 0 for all other state-action pairs. Furthermore, when an agent takes an action $a \in \mathcal{A}$, there is $p_{\text{rand}} = 0.05$ probability that an action $a' \in \mathcal{A} \setminus \{a\}$ will be executed. The environment-specific parameters are as follows: $R_{\max} = 10$, $\gamma = 0.95$, and the environment resets after a horizon of $H = 30$ steps.

Reward structure. In this environment, we consider a configuration of nine 3×3 grids along with a single 1×1 grid representing the goal state, as visually depicted in Figure 1b. To effectively represent the state space, we employ a state abstraction function denoted as $\psi : \mathcal{S} \rightarrow \{0, 1\}^{10}$. For each state $s \in \mathcal{S}$, the i-th entry of $\psi(s)$ is set to 1 if s resides in the i-th grid, and 0 otherwise. Building upon this state abstraction, we introduce a feature representation function, $f : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}^{10 \cdot |\mathcal{A}|}$, defined as follows: $f(s, a)_{(\cdot, a)} = \psi(s)$, and $f(s, a)_{(\cdot, a')} = \mathbf{0}, \forall a' \neq a$. Here, for any vector $v \in \{0, 1\}^{10 \cdot |\mathcal{A}|}$, we use the notation $v_{(i,a)}$ to refer to the (i, a)-th entry of the vector. Finally, we establish the set $\mathcal{R}_{\text{str}} = \{R_\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \text{ where } \phi \in \mathbb{R}^d\}$, where $R_\phi(s, a) = \langle \phi, f(s, a) \rangle$. Further, we define $\mathcal{R} := \mathcal{R}_{\text{inv}} \cap \mathcal{R}_{\text{str}}$ as discussed in Section 3. We note that $\overline{R} \in \mathcal{R}$.

Evaluation setup. We conduct our experiments with a tabular REINFORCE agent [41], and employ an optimal policy under the underlying reward function $\overline{R}$ as the target policy π^T. Algorithm 1 provides a sketch of the overall training process and shows how the agent's training interleaves with the expert-driven reward design process. Specifically, during training, the agent receives rewards based on the designed reward function R; the performance is always evaluated w.r.t. $\overline{R}$ (also reported in the plots). In our experiments, we considered two settings to systematically evaluate the utility of adaptive reward design: (i) a single learner with a uniformly random initial policy (where each action is taken with a probability of 0.25) and (ii) a diverse group of learners, each with distinct initial policies. To generate a collection of distinctive initial policies, we introduced modifications to a uniformly random policy. These modifications were designed to incorporate a 0.5 probability of the agent selecting suboptimal actions when encountering various "gate-states" (i.e., states with openings for navigation to other rooms). In our evaluation, we included five such unique initial policies.

Techniques evaluated. We evaluate the effectiveness of the following reward design techniques:

(i) $R^{\text{Orig}} := \overline{R}$ is a default baseline without any reward design.

(ii) R^{Invar} is obtained via solving the optimization problem in Eq. (4) with the substitution of I_h by a constant. This technique does not involve explicitly maximizing any reward informativeness during the optimization process.

(iii) R^{ExpRD} is obtained via solving the optimization problem proposed in [13]. This optimization problem is equivalent to Eq. (4), with the substitution of I_h by a non-adaptive informativeness criterion. We have employed the hyperparameters consistent with those provided in their work.

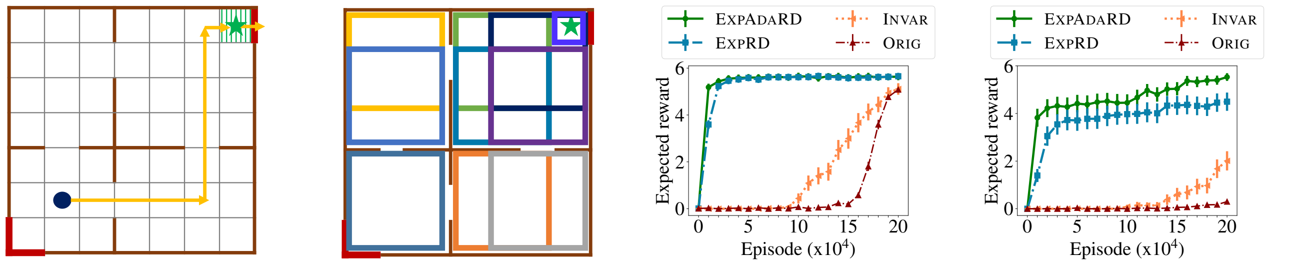
(iv) R_k^{ExpAdaRD} is based on our framework ExpAdaRD and obtained via solving the optimization problem in Eq. (4). For stability of the learning process, we update the policy more frequently than the reward as typically considered in the literature [12, 30, 48] – we provide additional details in the longer version of the paper.

Results. Figure 1 presents the results for both settings (i.e., a single learner and a diverse group of learners). The reported results are averaged over 40 runs (where each run corresponds to designing rewards for a specific learner), and convergence plots show the mean performance with standard error bars.[3] As evident from the results in Figures 1c and 1d, the rewards designed by ExpAdaRD significantly speed up the learner's convergence to optimal behavior when compared to the rewards designed by baseline techniques. Notably, the effectiveness of ExpAdaRD becomes more pronounced in scenarios featuring a diverse group of learners with distinct initial policies, where adaptive reward design plays a crucial role. Figure 3 presents a visualization of the designed reward functions generated by different techniques at various episodes. Notably, the rewards R^{Orig}, R^{Invar}, and R^{ExpRD} are agnostic to the learner's policy and remain constant throughout the training process. In Figures 3d, 3e, and 3f, we illustrate the R_k^{ExpAdaRD} rewards designed by our technique for three learners each with its distinct initial policy at $k = 1000, 2000, 3000, 100000$, and 200000 episodes. As observed in these plots, ExpAdaRD rapidly assigns high-magnitude numerical values to the designed rewards and adapts these rewards w.r.t. the learner's current policy. Initially (see $k = 1000$ episode plots), the rewards designed by ExpAdaRD encourage the agent to quickly reach the goal state ("green-star") by providing positive reward signals for optimal actions ("up", "right") followed by modifying reward signals in each episode to align with the learner's current policy.

5.2 Evaluation on LineK

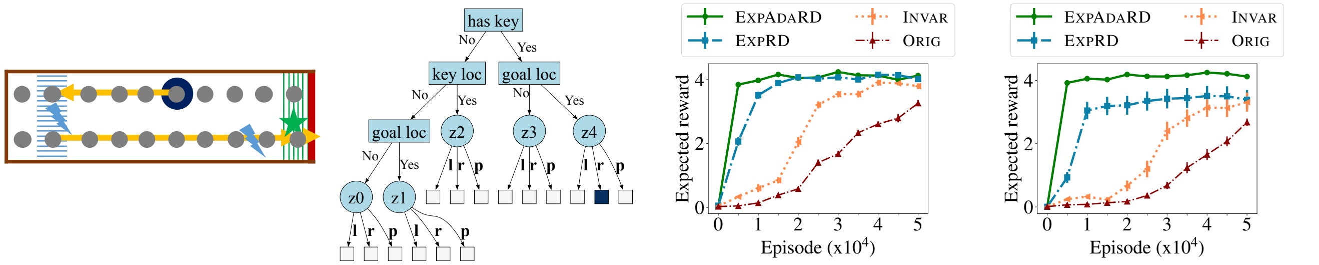
LineK (Figure 2a). This environment corresponds to a navigation task in a one-dimensional space where the agent has to first pick the key and then reach the goal. The environment used in our experiments is based on the work of [13] that also serves as a baseline technique. We represent the environment as an MDP with $\mathcal{S}$ states corresponding to nodes in a chain with the "gray circle" indicating the agent's initial location. Goal ("green-star") is available in the rightmost state, and the key is available at the

[3] We conducted the experiments on a cluster consisting of machines equipped with a 3.30 GHz Intel Xeon CPU E5-2667 v2 processor and 256 GB of RAM.



(a) Room: Environment (b) Room: Abstracted features (c) Room: Single learner setting (d) Room: Diverse learners setting

Figure 1: Results for Room. (a) shows the environment. (b) shows the abstracted feature space used for the representation of designed reward functions as a structural constraint. (c) shows results for the setting with a single learner. (d) shows results for the setting with a diverse group of learners with different initial policies. ExpAdaRD designs adaptive reward functions w.r.t. the learner's current policies, whereas other techniques are agnostic to the learner's policy. See Section 5.1 for details.



(a) LineK: Environment (b) LineK: Tree-based features (c) LineK: Single learner setting (d) LineK: Diverse learners setting

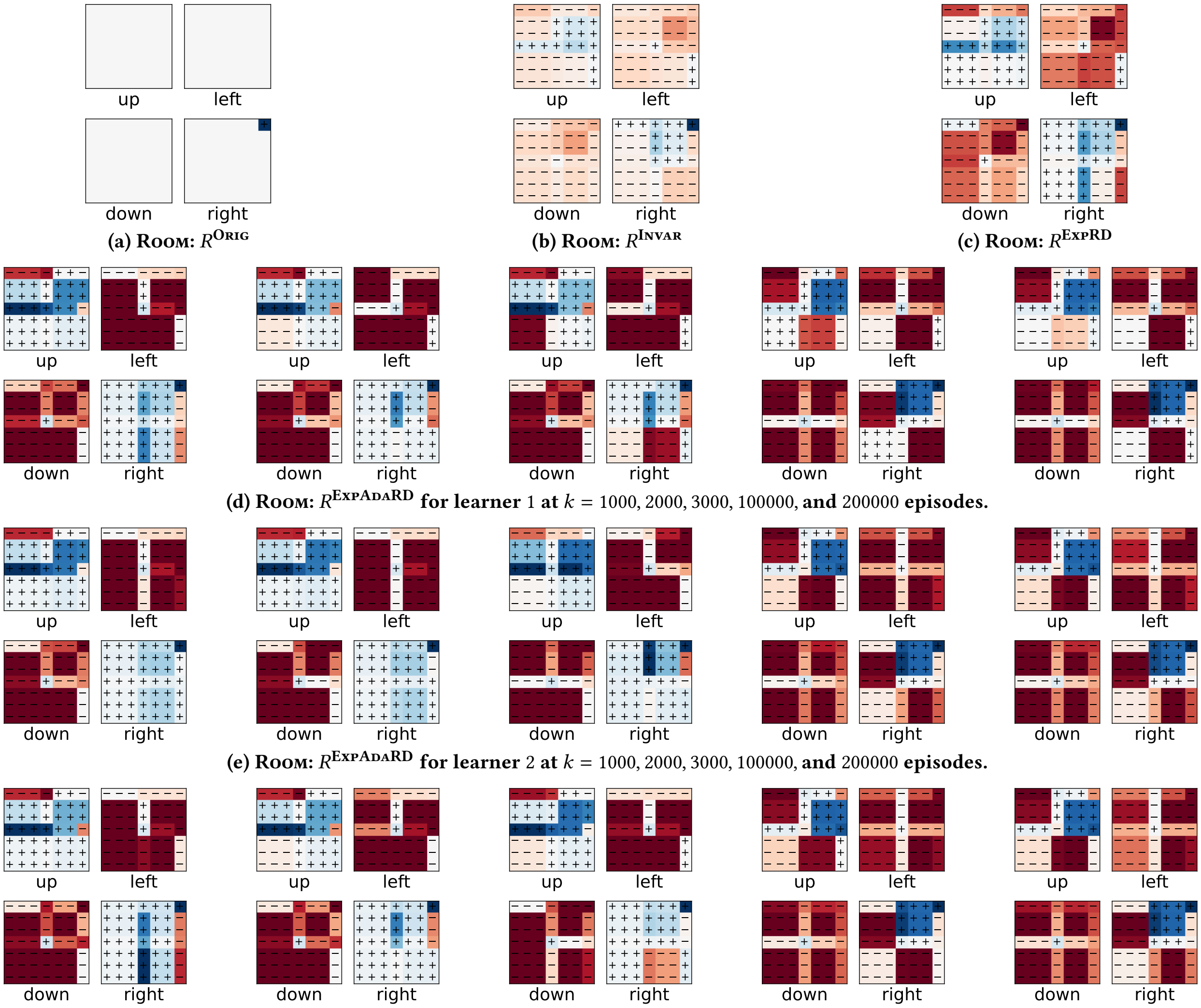
Figure 2: Results for LineK. (a) shows the environment. (b) shows the tree-based feature space used for the representation of designed reward functions as a structural constraint. (c) shows results for the setting with a single learner. (d) shows results for the setting with a diverse group of learners with different initial policies. ExpAdaRD designs adaptive reward functions w.r.t. the learner's current policies, whereas other techniques are agnostic to the learner's policy. See Section 5.2 for details.

state shown as "cyan-bolt". The agent can take three actions given by $\mathcal{A} := \{\text{"left"}, \text{"right"}, \text{"pick"}\}$. "pick" action does not change the agent's location, however, when executed in locations with the availability of the key, the agent acquires the key; if the 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 to the neighboring node according to the direction of the action. Similar to Room, the agent's move action is not applied if the new location crosses the wall, and there is p_{rand} probability of a random action. The agent gets a reward of $R_{\max}$ after it has navigated to the goal locations after acquiring the key and then takes a "right" action; note that this action also terminates the episode. The reward is 0 elsewhere and there is a discount factor γ. We set $p_{\text{rand}} = 0.1$, $R_{\max} = 10$, $\gamma = 0.95$, and the environment resets after a horizon of $H = 30$ steps.

Reward structure. We adopt a tree structured representation of the state space, as visually depicted in Figure 2b. To formalize this representation, we employ a state abstraction function denoted as $\psi : \mathcal{S} \to \{0,1\}^5$. For each state $s \in \mathcal{S}$, the i-th entry of $\psi(s)$ is set to 1 if s maps to the i-th circled node of the tree (i.e., parent to leaf nodes), and 0 otherwise. Then, we define the set $\mathcal{R}_{\text{str}}$ in a manner similar to that outlined in Section 5.1. Further, we define $\mathcal{R} := \mathcal{R}_{\text{inv}} \cap \mathcal{R}_{\text{str}}$ as discussed in Section 3. We note that $\overline{R} \in \mathcal{R}$.

Evaluation setup and techniques evaluated. Our evaluation setup for LineK environment is exactly the same as that used for Room environment (described in Section 5.1). In particular, all the hyperparameters (related to the REINFORCE agent, reward design techniques, and training process) are the same as in Section 5.1. In this evaluation, we again have two settings to evaluate the utility of adaptive reward design: (i) a single learner with a uniformly random initial policy (where each action is taken with a probability of 0.33) and (ii) a diverse group of learners, each with distinct initial policies. To generate a collection of distinctive initial policies, we introduced modifications to a uniformly random policy. These modifications were designed to incorporate a 0.7 probability of the agent selecting suboptimal actions from various states. In our evaluation, we included five such unique initial policies.

Results. Figure 2 presents the results for both settings (i.e., a single learner and a diverse group of learners). The reported results are averaged over 30 runs, and convergence plots show the mean performance with standard error bars. These results further demonstrate the effectiveness and robustness of ExpAdaRD across different settings in comparison to baselines. Analogous to Figure 3 in Section 5.1, Figure 4 presents a visualization of the designed reward functions produced by different techniques at various training episodes. These results illustrate the utility of our proposed informativeness criterion for adaptive reward design, particularly when dealing with various structural constraints to obtain interpretable rewards, including tree-structured reward functions.



(a) Room: R^{Orig}

(b) Room: R^{Invar}

(c) Room: R^{ExpRD}

(d) Room: R^{ExpAdaRD} for learner 1 at k = 1000, 2000, 3000, 100000, and 200000 episodes.

(e) Room: R^{ExpAdaRD} for learner 2 at k = 1000, 2000, 3000, 100000, and 200000 episodes.

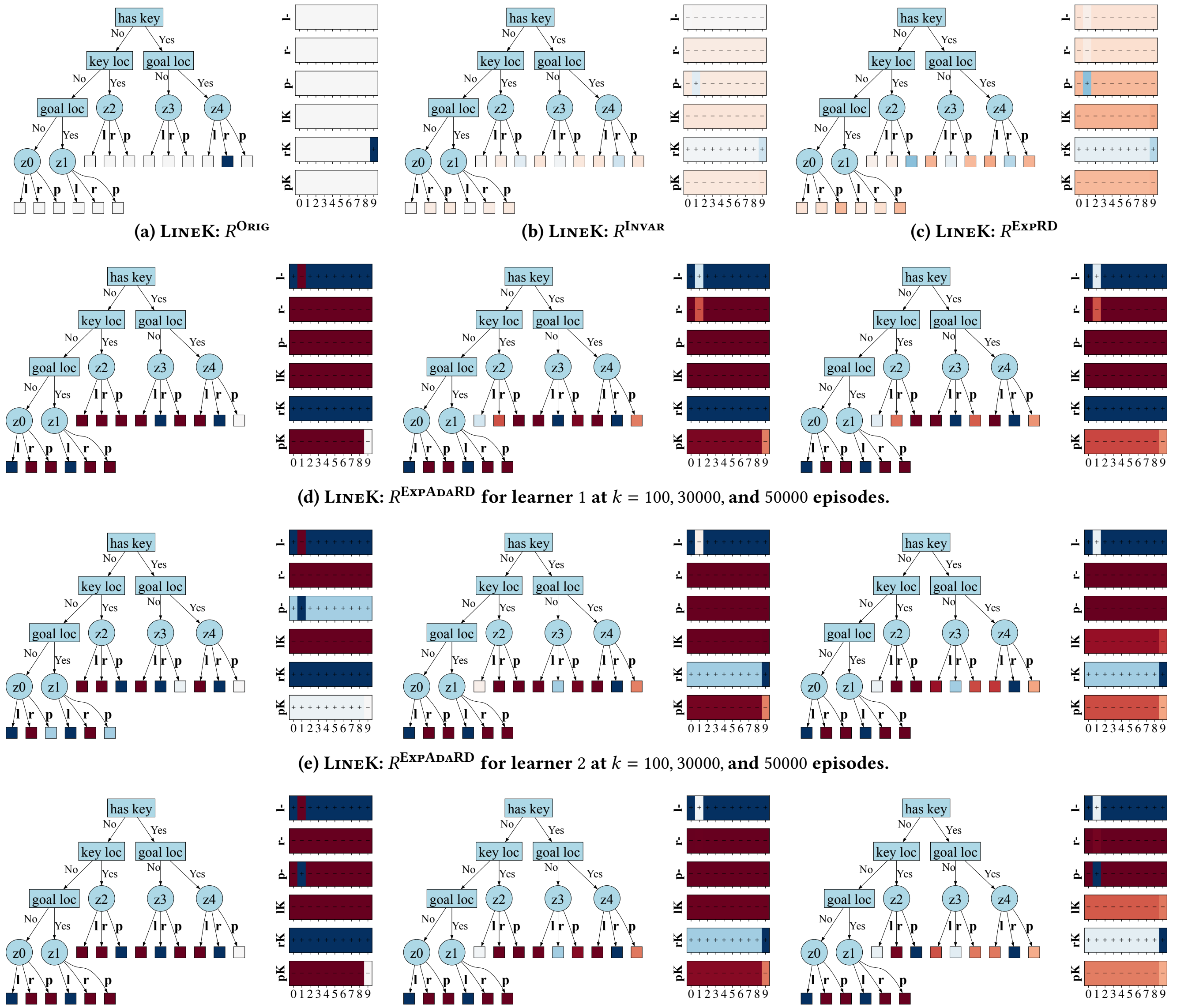
(f) Room: R^{ExpAdaRD} for learner 3 at k = 1000, 2000, 3000, 100000, and 200000 episodes.

Figure 3: Visualization of reward functions designed by different techniques in the Room environment for all four actions {"up", "left", "down", "right"}. (a) shows original reward function R^{Orig}. (b) shows reward function R^{Invar}. (c) shows reward function R^{ExpRD} designed by expert-driven non-adaptive reward design technique [13]. (d, e, f) show reward functions R^{ExpAdaRD} designed by our framework ExpAdaRD for three learners, each with its distinct initial policy, at different training episodes k. A negative reward is shown in Red color with the sign "-", a positive reward is shown in Blue color with the sign "+", and a zero reward is shown in white. The color intensity indicates the magnitude of the reward.

6 CONCLUDING DISCUSSIONS

We studied the problem of expert-driven reward design, where an expert/teacher seeks to provide informative and interpretable rewards to a learning agent. We introduced a novel reward informativeness criterion that adapts w.r.t. the agent's current policy. Based on this informativeness criterion, we developed an expert-driven adaptive reward design framework, ExpAdaRD. We empirically demonstrated the utility of our framework on two navigation tasks.

Next, we discuss a few limitations of our work and outline a future plan to address them. First, we conducted experiments on simpler environments to systematically investigate the effectiveness of our informativeness criterion in terms of adaptivity and structure of designed reward functions. It would be interesting to extend the evaluation of the reward design framework in more complex environments (e.g., with continuous state/action spaces) by leveraging an abstraction-based pipeline considered in [13]. Second, we considered fixed structural properties to induce interpretable reward functions. It would also be interesting to investigate the usage of our informativeness criterion for automatically discovering or optimizing the structured properties (e.g., nodes in the tree structure). Third, we empirically showed the effectiveness of our adaptive rewards, but adaptive rewards could also lead to instability in the agent's learning process. It would be useful to analyze our adaptive reward design framework in terms of an agent's convergence speed and stability.



(a) LineK: $R^{\textsc{Orig}}$ **(b) LineK: $R^{\textsc{Invar}}$** **(c) LineK: $R^{\textsc{ExpRD}}$**

(d) LineK: $R^{\textsc{ExpAdaRD}}$ for learner 1 at $k = 100, 30000$, and 50000 episodes.

(e) LineK: $R^{\textsc{ExpAdaRD}}$ for learner 2 at $k = 100, 30000$, and 50000 episodes.

(f) LineK: $R^{\textsc{ExpAdaRD}}$ for learner 3 at $k = 100, 30000$, and 50000 episodes.

Figure 4: Visualization of reward functions designed by different techniques in the LineK environment for all three actions {"left", "right", "pick"}. (a) shows original reward function $R^{\textsc{Orig}}$. (b) shows reward function $R^{\textsc{Invar}}$. (c) shows reward function $R^{\textsc{ExpRD}}$ designed by expert-driven non-adaptive reward design technique [13]. (d, e, f) show reward functions $R^{\textsc{ExpAdaRD}}$ designed by our framework ExpAdaRD for three learners, each with its distinct initial policy, at different training episodes k. These plots illustrate reward values for all combinations of triplets: agent's location (indicated as "key loc", "goal loc" in tree plots), agent's status whether it has acquired the key or not (indicated as "has key" in tree plots and letter "K" in bar plots), and three actions (indicated as 'l' for "left", 'r' for "right", 'p' for "pick"). A negative reward is shown in Red color with the sign "-", a positive reward is shown in Blue color with the sign "+", and a zero reward is shown in white. The color intensity indicates the reward magnitude.

ACKNOWLEDGMENTS

Funded/Co-funded by the European Union (ERC, TOPS, 101039090). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

ETHICS STATEMENT

This work presents a reward informativeness criterion that can be utilized in designing adaptive, informative, and interpretable rewards for a learning agent. Given the algorithmic nature of our work applied to agents, we do not foresee direct negative societal impacts of our work in the present form.

REFERENCES

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to Learn by Gradient Descent by Gradient Descent. In *NeurIPS*. 3981–3989.

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

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

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

[5] Tom Bewley and Freddy Lécué. 2022. Interpretable Preference-based Reinforcement Learning with Tree-Structured Reward Functions. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 118–126.

[6] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. 2015. Reinforcement Learning from Demonstration through Shaping. In *IJCAI*. AAAI Press, 3352–3358.

[7] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. 2017. Decision-Making with Non-Markovian Rewards: From LTL to Automata-based Reward Shaping. In *RLDM*. 279–283.

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

[9] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. 2014. Active Reward Learning.. In *Robotics: Science and Systems*.

[10] Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, and Fabio Patrizi. 2020. Imitation Learning over Heterogeneous Agents with Restraining Bolts. In *ICAPS*, Vol. 30. AAAI Press, 517–521.

[11] Alper Demir, Erkin Çilden, and Faruk Polat. 2019. Landmark Based Reward Shaping in Reinforcement Learning with Hidden States. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 1922–1924.

[12] Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. 2022. Exploration-Guided Reward Shaping for Reinforcement Learning under Sparse Rewards. In *NeurIPS*. 5829–5842.

[13] Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. 2021. Explicable Reward Design for Reinforcement Learning Agents. In *NeurIPS*. 20118–20131.

[14] Sam Devlin and Daniel Kudenko. 2012. Dynamic Potential-based Reward Shaping. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 433–440.

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

[16] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. 2019. Using Natural Language for Reward Shaping in Reinforcement Learning. In *IJCAI*. ijcai.org, 2385–2391.

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

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

[19] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. 2022. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *Journal of Artificial Intelligence Research* 73 (2022), 173–208.

[20] Michael R. James and Satinder P. Singh. 2009. SarsaLandmark: An Algorithm for Learning in POMDPs with Landmarks. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 585–591.

[21] Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. 2021. Temporal-Logic-Based Reward Shaping for Continuing Reinforcement Learning Tasks. In *AAAI*. AAAI Press, 7995–8003.

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

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

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

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

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

[27] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. 2008. Programming by Choice: Urban Youth Learning Programming with Scratch. In *SIGCSE*. ACM, 367–371.

[28] Maja J. Mataric. 1994. Reward Functions for Accelerated Learning. In *ICML*. Morgan Kaufmann, 181–189.

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

[30] Farzan Memarian, Wonjoon Goo, Rudolf Lioutikov, Scott Niekum, and Ufuk Topcu. 2021. Self-Supervised Online Reward Shaping in Sparse-Reward Environments. In *IROS*. IEEE, 2369–2375.

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

[32] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On First-Order Meta-Learning Algorithms. *CoRR* abs/1803.02999 (2018).

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

[34] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. In *ICML*. PMLR, 4254–4263.

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

[36] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2021. Policy Teaching in Reinforcement Learning via Environment Poisoning Attacks. *Journal of Machine Learning Research* 22, 210 (2021), 1–45.

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

[38] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-Learning with Memory-Augmented Neural Networks. In *ICML*. PMLR, 1842–1850.

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

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

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

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

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

[44] Baicen Xiao, Qifan Lu, Bhaskar Ramasubramanian, Andrew Clark, Linda Bushnell, and Radha Poovendran. 2020. FRESH: Interactive Reward Shaping in High-Dimensional State Spaces using Human Feedback. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 1512–1520.

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

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

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

[48] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. 2018. On Learning Intrinsic Rewards for Policy Gradient Methods. In *NeurIPS*. 4649–4659.