Reconciling Intrinsic Rewards via Constrained Policy Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

State-of-the-art reinforcement learning (RL) algorithms typically use random sampling (e.g., ϵ-greedy) for exploration, but this method fails in hard exploration tasks like Montezuma's Revenge. To address the challenge of exploration, prior works incentivize the agent to visit novel states using an exploration bonus (also called intrinsic rewards), which led to excellent results on some hard exploration tasks. However, recent studies show that on many other tasks intrinsic rewards can bias policy optimization leading to poor performance compared to optimizing only the environment reward. The low performance results from the agent seeking intrinsic rewards and performing unnecessary exploration even when sufficient environment reward is provided. This inconsistency in performance across tasks prevents widespread use of intrinsic rewards with RL algorithms. We propose a principled constrained policy optimization procedure to eliminate the detrimental effects of intrinsic rewards while preserving their merits when applicable. Our method automatically tunes the importance of intrinsic reward: it suppresses intrinsic rewards when they are not needed and increases them when exploration is required. The end result is a superior exploration algorithm that does not require manual tuning to balance intrinsic rewards against environment rewards. Experimental results across 61 Atari games validate our claim.

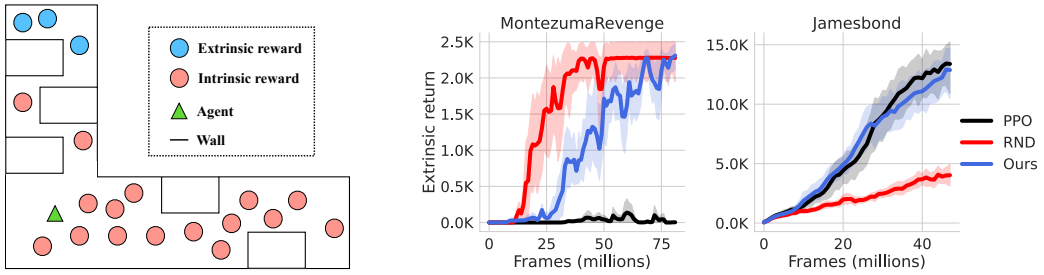
1 Introduction

Exploration-exploitation is at the core of reinforcement learning (RL) [1]. An agent must balance the time spent discovering better policies (exploration) and the time spent taking greedy actions to maximize rewards (exploitation). In the absence of methods that work well in both theory and practice, state-of-the-art methods rely on heuristic exploration strategies such as adding noise to actions or randomly sampling a suboptimal action with a small probability (e.g., ϵ-greedy). However, such strategies fail on notoriously hard exploration tasks such as the Atari game Montezuma's Revenge where the rewards are sparse [2]. In sparse reward scenarios, non-zero rewards are infrequent which hinders policy improvement.

A popular approach for solving hard exploration tasks is to supplement the environment (or the extrinsic) reward (r_E) with an exploration bonus (or intrinsic reward r_I) generated by the agent itself [3–5, 2, 6]. Intrinsic rewards encourage the agent to visit novel states and thus increase the chance of visiting states that yield extrinsic rewards. Many prior works [6, 2, 7] show that jointly optimizing intrinsic and extrinsic rewards (i.e., $r_E + \lambda r_I$, where $\lambda \geq 0$ is a hyperparameter) instead of only optimizing the extrinsic reward (i.e. $\lambda = 0$) improves performance on hard exploration tasks.

Despite success on hard exploration tasks such as *Montezuma's Revenge*, a recent study found that using intrinsic rewards does not consistently outperform simple exploration strategies such as ϵ-greedy across Atari games [8]. Whitney et al. [9] suggest that intrinsic rewards bias the policy optimization

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



(a) (b)

Figure 1: **(a)** Though the intrinsic rewards can guide the agent upwards in the correct direction, the majority of them will lead the agent astray along the bottom corridor. **(b)** RND does not consistently improve PPO in general Atari games, whereas our method matches the best-performing algorithm among RND and PPO consistently.

objective and can therefore degrade performance. In summary, jointly maximizing intrinsic and extrinsic rewards does not necessarily yield the optimal policy with regard to extrinsic rewards alone.

Fig. 1a illustrates this intuition using a toy example. Here the green triangle is the agent, and the blue/pink circles denote the location of extrinsic and intrinsic rewards, respectively. The pink intrinsic rewards are the easiest to find, and thus the agent tends to collect these initially. Thus, maximizing the sum of intrinsic and extrinsic rewards can lead the learned policy to move right along the bottom corridor and therefore be suboptimal with respect to the extrinsic rewards which are located at the top. In such a scenario, better performance will be obtained if the agent only optimizes using the extrinsic reward ($\lambda = 0$). The trouble, however, is that in most environments one cannot know a priori how to optimally trade off intrinsic and extrinsic rewards (i.e., choose λ).

A common practice is to find the best λ using an extensive hyperparameter search. The problem is that different values of λ are best for different tasks (see Fig. 4). Furthermore, as the agent progresses on a task, the best exploration-exploitation trade off can vary and a constant λ may not be optimal throughout training. In the initial absence of extrinsic rewards, exploration is preferred. Once enough extrinsic rewards are obtained, the agent might prefer exploitation of these rewards instead of exploration. The exact dynamics of the exploration-exploitation trade off are task-dependent, and per-task tuning is tedious, undesirable and often computationally infeasible.

We present an exploration strategy that alleviates the need to manually tune the relative importance of extrinsic and intrinsic rewards. Our method leverages the bias of intrinsic rewards when it is useful for exploration and mitigates this bias when it does not result in higher extrinsic rewards. This is achieved using an *extrinsic optimality constraint* that forces the extrinsic rewards earned by the joint policy to be equal to those earned by the policy that is optimal with respect to extrinsic rewards alone. Directly optimizing with this constraint in general settings is intractable because the optimal extrinsic reward is unknown. We devise a practical algorithm which we call **Extrinsic-Intrinsic Policy Optimization (EIPO)** that uses an approximation to solve this constrained optimization problem (Section 3).

While in principle we can apply EIPO to any intrinsic reward method, we evaluate performance using the state-of-the-art, *random network distillation (RND)* [7]. Fig. 1b presents *teaser* results on two Atari games: (i) *Montezuma's revenge* on which joint optimization with RND (red) substantially outperforms a PPO policy [10] optimized with only extrinsic rewards (black); (ii) *Jamesbond* where PPO substantially outperforms RND. These results demonstrate that in some games intrinsic reward bias helps, but in others it can hurt. In contrast, our algorithm EIPO (blue) matches the best algorithm in both games, showing that it can leverage intrinsic rewards as necessary. A large-scale study on all 61 Atari games reinforces this finding. Additionally, in some games EIPO outperforms multiple strong baselines with and without intrinsic rewards, indicating that our method both mitigates performance degradation resulting from intrinsic reward bias and improves performance beyond the current state-of-the-art.

2 Preliminaries

We consider a discrete-time Markov Decision Process (MDP) consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, and an extrinsic reward function $\mathcal{R}_E : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. We distinguish the extrinsic and

intrinsic reward components by E and I, respectively. The extrinsic reward function $\mathcal{R}_E$ refers to the actual task objective (e.g., game score). The agent starts from an initial state s_0 sampled from the initial state distribution $\rho_0 : \mathcal{S} \mapsto \mathbb{R}$. At each timestep t, the agent perceives a state s_t of the environment, takes action a_t sampled from the policy π, receives extrinsic reward $r_t^E = \mathcal{R}_E(s_t, a_t)$, and moves to the next state s_{t+1} according to the transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$. The agent's goal is to use interactions with the environment to find the optimal policy π such that the extrinsic objective value $J_E(\pi)$ is maximized:

$$\max_{\pi} J_E(\pi), \text{ where } J_E(\pi) = \mathbb{E}_{s_0,a_0,\cdots\sim\pi}\Big[\sum_{t=0}^{\infty}\gamma^t r_t^E\Big] \text{ (Extrinsic objective)}, \tag{1}$$
$$s_0 \sim \rho_0, a_t \sim \pi(a|s_t), s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)\ \forall t > 0$$

where γ denotes a discount factor. For brevity, we abbreviate $\mathbb{E}_{s_0,a_0,\cdots\sim\pi}\Big[.\Big]$ as $\mathbb{E}_{\pi}\Big[.\Big]$ unless specified.

Curiosity-driven exploration methods [6, 7, 2] attempt to ease exploration by providing "intrinsic rewards" (or "exploration bonus") to the agent to encourage it to visit unseen states. With the intrinsic reward function $\mathcal{R}_I : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, the optimization objective is:

$$\max_{\pi\in\Pi} J_{E+I}(\pi), \text{ where } J_{E+I}(\pi) = \mathbb{E}_{\pi}\big[\sum_{t=0}^{\infty}\gamma^t(r_t^E + \lambda r_t^I)\big] \text{ (Mixed objective)}, \tag{2}$$

where λ denotes the intrinsic reward scaling coefficient. We abbreviate intrinsic reward at timestep t as $r_t^I = \mathcal{R}_I(s_t, a_t)$. State-of-the-art curiosity-exploration strategies [7, 11] often optimize the objective in Eq. 2 using Proximal Policy Optimization (PPO) [10].

3 Mitigating the Bias of Intrinsic Rewards

Simply maximizing the sum of intrinsic and extrinsic rewards does not guarantee a policy that also maximizes the extrinsic reward. Formally, $\arg\max_{\pi_{E+I}\in\Pi} J_{E+I}(\pi_{E+I}) \neq \arg\max_{\pi_E\in\Pi} J_E(\pi_E)$. At convergence the optimal policy $\pi^*_{E+I} = \arg\max_{\pi_{E+I}} J_{E+I}(\pi_{E+I})$ could be suboptimal w.r.t. J_E, which measures the agent's task performance. Because the agent's performance is measured only based on extrinsic reward, we propose to enforce an *extrinsic optimality constraint* to ensure that the optimal "mixed" policy $\pi^*_{E+I} = \arg\max_{\pi_{E+I}} J_{E+I}(\pi_{E+I})$ leads to as much extrinsic reward as the optimal "extrinsic" policy $\pi^*_E = \arg\max_{\pi_E\in\Pi} J_E(\pi_E)$. The resulting optimization objective is the following:

$$\max_{\pi_{E+I}\in\Pi} J_{E+I}(\pi_{E+I}) \tag{3}$$
$$\text{subject to } J_E(\pi_{E+I}) - \max_{\pi_E} J_E(\pi_E) = 0 \qquad \text{(Extrinsic optimality constraint)}.$$

Solving this optimization problem can be viewed as proposing a policy π_{E+I} that maximizes J_{E+I}, and checking if the proposed π_{E+I} is feasible given the extrinsic optimality constraint.

The constrained objective is difficult to optimize because evaluating the extrinsic optimality constraint requires $J_E(\pi^*_E)$, which is unknown. We show that this optimization problem can be solved by transforming it into an *unconstrained min-max* optimization problem using Lagrangian duality (Section 3.1). Next, we present an iterative algorithm for min-max optimization by alternating between maximization and minimization in Section 3.2 and an implementation in Section 3.3.

3.1 The Dual Objective: Unconstrained Min-Max Optimization Problem

The Lagrangian dual problem for the primal constrained optimization problem in Eq. 3 is:

$$\min_{\alpha\in\mathbb{R}}\Big[\max_{\pi_{E+I}\in\Pi} J_{E+I}(\pi_{E+I}) + \alpha(J_E(\pi_{E+I}) - \max_{\pi_E\in\Pi} J_E(\pi_E))\Big], \tag{4}$$

where $\alpha \in \mathbb{R}$ is the Lagrangian multiplier. We simplify Eq. 4 by merging $J_{E+I}(\pi)$ and $J_E(\pi)$:

$$J^{\alpha}_{E+I}(\pi_{E+I}) = J_{E+I}(\pi_{E+I}) + \alpha J_E(\pi_{E+I}) = \mathbb{E}_{\pi_{E+I}}\Big[\sum_{t=0}^{\infty}\gamma^t((1+\alpha)r^E(s_t, a_t) + r^I(s_t, a_t))\Big].$$

Substituting $J^{\alpha}_{E+I}(\pi_{E+I})$ into Eq. 4 and rearranging terms yields the following min-max problem:

$$\min_{\alpha \in \mathbb{R}^+} \Big[\max_{\pi_{E+I} \in \Pi} \min_{\pi_E \in \Pi} J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E) \Big]. \tag{5}$$

3.2 Algorithm for Optimizing the Dual Objective

We now describe an algorithm for solving π_{E+I}, π_E, and α in each of the subproblems in Eq. 5.

Extrinsic policy π_E (min-stage). π_E is optimized via the minimization sub-problem, which can be re-written as a maximization problem:

$$\min_{\pi_E \in \Pi} J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E) \quad \rightarrow \quad \max_{\pi_E \in \Pi} \alpha J_E(\pi_E) - J^{\alpha}_{E+I}(\pi_{E+I}) \tag{6}$$

The central challenge in this optimization is that evaluating objectives $J^{\alpha}_{E+I}(\pi_{E+I})$ and $J_E(\pi_E)$ requires sampling trajectories from both policies π_{E+I} and π_E which is data inefficient. If the two policies are similar, we can perform the optimization using data collected only from π_{E+I} using the approximation proposed in [12]. To make this approximation we first use an identity from [13] to re-write the objective $\alpha J_E(\pi_E) - J^{\alpha}_{E+I}(\pi_{E+I})$ as (see Appendix A.1 for detailed derivation):

$$\alpha J_E(\pi_E) - J^{\alpha}_{E+I}(\pi_{E+I}) = \mathbb{E}_{\pi_E}\Big[\sum_{t=0}^{\infty} \gamma^t U^{\pi_{E+I}}_{\min}(s_t, a_t)\Big], \tag{7}$$

$$\text{where } U^{\pi_{E+I}}_{\min}(s_t, a_t) := \alpha r^E_t + \gamma V^{\pi_{E+I}}_{E+I}(s_{t+1}) - V^{\pi_{E+I}}_{E+I}(s_t),$$

$$V^{\pi_{E+I}}_{E+I}(s_t) := \mathbb{E}_{\pi_{E+I}}\Big[\sum_{t=0}^{\infty} \gamma^t (r^E_t + r^I_t) | s_0 = s_t\Big].$$

Now we can use the approximation from [12] to formulate Eq. 7 as:

$$\mathbb{E}_{\pi_E}\Big[\sum_{t=0}^{\infty} \gamma^t U^{\pi_{E+I}}_{\min}(s_t, a_t)\Big] \approx \mathbb{E}_{\pi_{E+I}}\Big[\sum_{t=0}^{\infty} \gamma^t \frac{\pi_E(a_t|s_t)}{\pi_{E+I}(a_t|s_t)} U^{\pi_{E+I}}_{\min}(s_t, a_t)\Big] \tag{8}$$

$$\text{subject to } \mathbb{E}_{s \sim \pi_{E+I}}\Big[\mathrm{D_{KL}}(\pi_{E+I}(.|s) || \pi_E(.|s))\Big] \leq \delta,$$

where $\mathrm{D_{KL}}$ denotes the Kullback–Leibler divergence and δ is a small constant threshold that limits the divergence between two policies. Note that the expectation is computed with respect to data collected only from π_{E+I}. We solve for the new π_E by optimizing a clipped version of the objective in Eq. 8, similar to PPO. More details are provided in Appendix A.2.

Mixed policy π_{E+I} (max-stage). This subproblem of solving π_{E+I} is posed as

$$\max_{\pi_{E+I} \in \Pi} J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E). \tag{9}$$

Similarly to solving π_E we use the approximation used in Eq. 8 to rewrite the objective $J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E)$ as follows (see Appendix A.1 for details):

$$J_{\alpha E+I}(\pi_{E+I}) - \alpha J_E(\pi_E) \approx \mathbb{E}_{\pi_E}\Big[\sum_{t=0}^{\infty} \gamma^t \frac{\pi_{E+I}(a_t|s_t)}{\pi_E(a_t|s_t)} U^{\pi_E}_{\max}(s_t, a_t)\Big] \tag{10}$$

$$\text{subject to } \mathbb{E}_{s \sim \pi_E}\Big[\mathrm{D_{KL}}(\pi_E(.|s) || \pi_{E+I}(.|s))\Big] \leq \delta,$$

$$\text{where } U^{\pi_E}_{\max}(s_t, a_t) := (1+\alpha) r^E_t + r^I_t + \gamma \alpha V^{\pi_E}_E(s_{t+1}) - \alpha V^{\pi_E}_E(s_t)$$

$$V^{\pi_E}_E(s_t) := \mathbb{E}_{\pi_E}\Big[\sum_{t=0}^{\infty} \gamma^t r^E_t | s_0 = s_t\Big].$$

Lagrangian multiplier α. Let $g(\alpha) := \max_{\pi_{E+I} \in \Pi} \min_{\pi_E \in \Pi} J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E)$. We solve $\min_{\alpha} g(\alpha)$ using gradient descent with step size β as shown below (see Appendix A.1 for details):

$$\alpha \leftarrow \alpha - \beta \nabla_{\alpha} g(\alpha) \approx \alpha - \beta \mathbb{E}_{\pi_E}\Big[\sum_{t=0}^{\infty} \gamma^t \frac{\pi_{E+I}(a_t|s_t)}{\pi_E(a_t|s_t)} r^E_t + \gamma V^{\pi_E}_E(s_{t+1}) - V^{\pi_E}_E(s_t)\Big], \tag{11}$$

$$\text{subject to } \mathbb{E}_{s \sim \pi_E}\Big[\mathrm{D_{KL}}(\pi_E(.|s) || \pi_{E+I}(.|s))\Big] \leq \delta.$$

3.3 Implementation

Min-max alternation schedule We use an iterative optimization scheme that alternates between solving for π_E by minimizing the objective in Eq. 6, and solving for π_{E+I} by maximizing the objective in Eq. 9. Typically, min-max optimization switches between minimization and maximization stages at every iteration. In our implementation, at each iteration trajectories are collected using one of the the two policies, and both policies are trained. We find that switching the sampling policy in each iteration hinders the performance of our method. We hypothesize that frequently switching leads to insufficient training for each policy, and thus we use the objective value between two subsequent iterations to determine when to switch. Let $J[i]$ be the objective value $J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E)$ at iteration i. We switch the stage when the objective value does not improve anymore. The switching rule can be summarized as:

$$\begin{cases} J[i] - J[i-1] \leq 0 \implies \texttt{max_stage} \leftarrow \texttt{False}, & \text{if } \texttt{max_stage = True} \\ J[i] - J[i-1] \geq 0 \implies \texttt{max_stage} \leftarrow \texttt{True}, & \text{if } \texttt{max_stage = False}, \end{cases}$$

where `max_stage` is a binary variable indicating whether the current stage is max-stage. When max-stage is done, α is updated using Eq. 11.

Parameter sharing In an image state space (e.g., Atari) it is preferable to share the convolutional neural network (CNN) backbone between π_E and π_{E+I} for the sake of data efficiency. However, if the backbone is shared, maximizing the objective (Eq. 9) over π_{E+I} could decrease $\alpha J_E(\pi_E)$ and impede the performance of π_E due to gradient interference in the shared backbone. Similarly, minimizing the objective (Eq. 6) over π_E could decrease J^{α}_{E+I} and degrade the performance of π_{E+I} as well. To prevent such interference, we jointly optimize both π_{E+I} and π_E with additional auxiliary objectives $\max_{\pi_{E+I}} J_{E+I}(\pi_{E+I})$ and $\max_{\pi_E} J_E(\pi_E)$ during the min-stage and max-stage, respectively. Intuitively, the auxiliary objective compensates against gradient interference by introducing a counteracting force that attempts to fix the objective not being directly optimized during the current stage. For example, when maximizing $J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E)$ over π_{E+I}, the auxiliary objective $\max_{\pi_E} J_E(\pi_E)$ can prevent any updates from decreasing $J(\pi_E)$.

In summary, we outline the practical policy optimization algorithm, *Extrinsic-Intrinsic Policy Optimization (EIPO)*, for solving Eq. 5 in Algorithm 1. The detailed version is presented in Algorithm 2 in the Appendix. Note that we solve the constrained problem (Eq. 10 and Eq. 8) using a PPO-based approach. The full implementation details, including hyperparameters can be found in Appendix A.2.

Algorithm 1 Extrinsic-Intrinsic Policy Optimization (EIPO)

Initialize policies π_{E+I} and π_E, `max_stage[0]` $\leftarrow$ `False`, and $J[0] \leftarrow 0$
for $i = 1 \cdots$ **do** ▷ i denotes iteration index
  if `max_stage[i - 1]` **then** ▷ Max-stage: rollout by π_E and update π_{E+I}
    Collect trajectories τ_E using π_E and compute $U^{\pi_E}_{\max}(s_t, a_t)\ \forall (s_t, a_t) \in \tau_E$
    Update π_{E+I} by Eq. 10 and π_E by auxiliary objective (Section 3.3)
    $J[i] \leftarrow J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E)$
    `max_stage[i]` $\leftarrow J[i] - J[i-1] \leq 0$
  else ▷ Min-stage: rollout by π_{E+I} and update π_E
    Collect trajectories τ_{E+I} using π_{E+I} and compute $U^{\pi_{E+I}}_{\min}(s_t, a_t)\ \forall (s_t, a_t) \in \tau_{E+I}$
    Update π_E by Eq. 8 and π_{E+I} by auxiliary objective (Section 3.3)
    $J[i] \leftarrow J^{\alpha}_{E+I}(\pi_{E+I}) - \alpha J_E(\pi_E)$
    `max_stage[i]` $\leftarrow J[i] - J[i-1] \geq 0$
  end if
  if `max_stage[i - 1] = True` and `max_stage[i] = False` **then**
    Update α (Eq. 32) ▷ Update when the max-stage is done
  end if
end for

4 Experiments

We evaluate our method in 61 Atari games from the Arcade Learning Environment (ALE) [14] benchmark, as well as Pycolab environments [15] for illustrative examples. We compare against the following baselines, where the text inside the parentheses denotes the acronym.

- **PPO [10]**: The policy is trained using PPO with extrinsic rewards only: $\pi^* = \arg\max_{\pi \in \Pi} J_E(\pi)$.
- **RND [7]**: The policy is trained using PPO with the sum of extrinsic rewards r_t^E and intrinsic rewards λr_t^I generated by RND: $\pi^* = \arg\max_{\pi \in \Pi} J_{E+I}(\pi)$.
- ***Ext-norm-RND* (*EN*)**: We propose a variant of RND where the extrinsic rewards are normalized using running mean and standard deviation (Appendix A.2). We include this as a baseline since our empirical results show that extrinsic reward normalization improves over RND in games where vanilla RND performs much worse than PPO.
- ***Decay-RND* (*DY*)**: We propose a variant of *Ext-norm-RND* where intrinsic rewards are progressively scaled down to eliminate exploration bias over time. Intrinsic rewards r_t^I are scaled by $\lambda(i)$, where i denotes the iteration number. The objective function turns into $J_{E+I} = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t (r_t^E + \lambda(i) r_t^I)\Big]$, where $\lambda(i)$ is defined as $\lambda(i) = \text{clip}(\frac{i}{I}(\lambda_{\text{max}} - \lambda_{\text{min}}), \lambda_{\text{min}}, \lambda_{\text{max}})$, where λ_{max} and λ_{min} denote the predefined maximum and minimum $\lambda(i)$, and I is the iteration after which decay is fixed. We follow the linear decay schedule used to decay ϵ-greedy exploration in DQN [16].
- ***Decoupled-RND* (*DC*) [17]**: We adapt the method proposed in [17] to *Ext-norm-RND*

$$\pi^*_{E+I} = \arg\max_{\pi_{E+I}} \mathbb{E}_{\pi_{E+I}}\Big[\sum_{t=0}^{\infty} \gamma^t (r_t^E + r_t^I) - \mathrm{D}_{\mathrm{KL}}(\pi_E || \pi_{E+I})\Big], \quad \pi^*_E = \arg\max_{\pi_E} \mathbb{E}_{\pi_I}\Big[\sum_{t=0}^{\infty} \gamma^t r_t^E\Big].$$

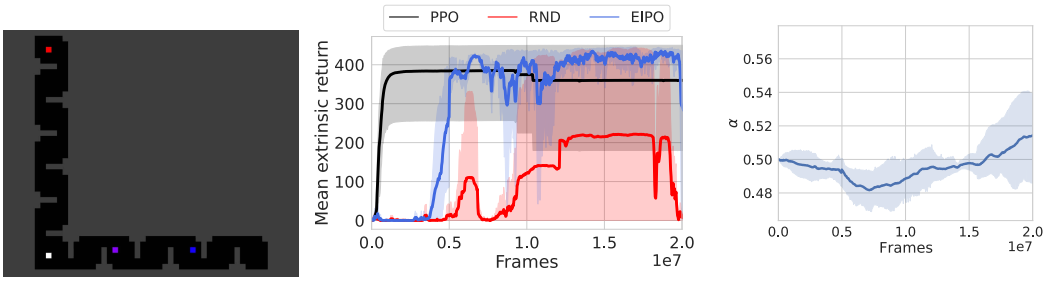
  The exploration policy π_{E+I} collects trajectories for training π_{E+I} and π_E. π_{E+I} and π_E maximize mixed and extrinsic objectives, respectively. The $\mathrm{D}_{\mathrm{KL}}(\pi_E || \pi_{E+I})$ term in the objective incentivizes π_I to perform differently from π_E.

Our method EIPO is implemented using *Ext-norm-RND*. The policies π_{E+I} and π_E are parametrized by two separate multi-layer perceptrons (MLP) with a shared CNN backbone. The value functions $V_{E+I}^{\pi_{E+I}}$ and $V_E^{\pi_E}$ are represented in the same fashion, and share the same CNN backbone with both policy networks. The hyperparameters and implementation details are described in the Appendix A.2. Note that while we chose to use RND due to it's state-of-the-art performance in hard exploration Atari games, our method is applicable to any bonus-based exploration method.

4.1 Illustrative example

We exemplify intrinsic reward bias using a simple navigation environment shown in Fig. 2a which illustrates a task with conflicting intrinsic and extrinsic rewards. The white sprite denotes the agent, which observes a 5×5 window view of it's surroundings. If it reaches the (red) location at the top of the map, an extrinsic reward is provided. The (blue and magenta) blocks are randomly placed in the bottom corridor at the start of each episode. Because these blocks randomly change location, they induce RND intrinsic rewards throughout the course of training. Because intrinsic rewards along the bottom corridor are easier to obtain than the extrinsic reward, optimizing the mixed objective J_{E+I} can yield a policy that results in the agent exploring the bottom corridor (i.e., exploiting the intrinsic reward) without ever discovering the extrinsic reward at the top. Fig. 2b plots the evolution of the average extrinsic returns for PPO, RND, and EIPO across 5 random seeds during training. We find that EIPO outperforms both RND and PPO. RND gets distracted by colored blocks and underperforms compared to vanilla PPO. Notably, PPO performs slightly worse than EIPO, possibly because the extrinsic reward at the top cannot be trivially found without the guidance of intrinsic rewards.

We hypothesize that EIPO aggressively explores early on during training, and then focuses on exploiting extrinsic rewards after discovering the red location. To validate this hypothesis, we plot the average α parameter, which controls the importance of extrinsic optimality of EIPO at each training iteration in Fig. 2c. It can be seen that α decreases in the beginning, before increasing after the extrinsic return starts rising in Fig. 2b. This observation corroborates our hypothesis that EIPO is able



(a) (b) (c)

Figure 2: **(a)** The white sprite is the agent at the starting location. The red location is the extrinsic reward. The magenta and the blue blocks denote intrinsic attractors. **(b)** RND fails to match PPO while our method EIPO outperforms both RND and PPO, demonstrating that EIPO mitigates the performance degradation resulting from intrinsic rewards. **(c)** α controls the importance of extrinsic optimality constraint. It decreases in the beginning before it increases when extrinsic return rises between 0.5 and 1.0 millions frames. This shows that our method emphasizes intrinsic rewards at the start of training, and capitalizes on extrinsic rewards once found.

to dynamically leverage intrinsic rewards for exploration while ultimately achieving high extrinsic returns.

4.2 Does EIPO generally improve PPO performance?

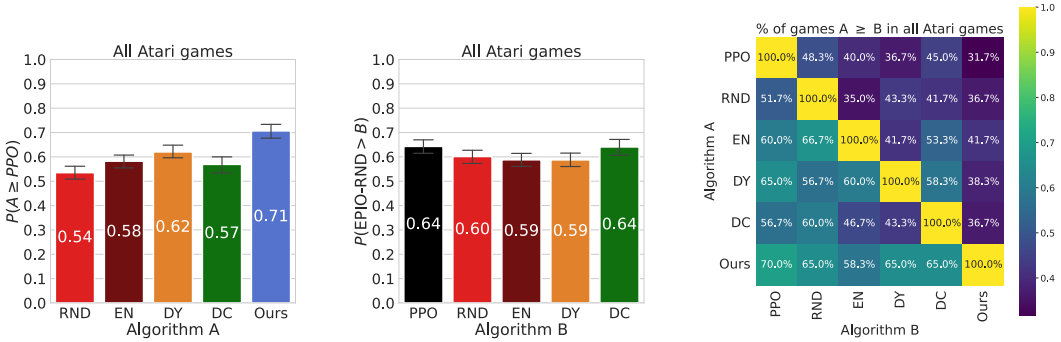
We validate whether EIPO prevents the possible performance degradation introduced by intrinsic rewards, and consistently either improves or matches the performance of PPO in 61 Atari games. As our goal is to investigate if an algorithm generally performs better than PPO, rather than measuring raw performance we evaluate each algorithm using the "probability of improvement" (Appendix A.4) metric suggested in [18]. We ran at least 5 random seeds for each method in each environment, collecting the median extrinsic returns within the last 100 episodes and calculating the probability of improvement $P(A \geq \text{PPO})$ with 95%-confidence interval against PPO for each algorithm A. The confidence interval is estimated using the bootstrapping method. Fig. 3a shows the probability of improvement for each method against PPO: $P(A > \text{PPO})\ \forall A \neq \text{PPO}$. It can be seen that RND is the least likely to improve PPO, suggesting that RND does not yield consistent performance improvements across all 61 Atari games. *Ext-norm-RND* achieves a higher probability of improvement than RND, which suggests that normalizing extrinsic rewards improves the performance degradation of RND. *Decay-RND* has higher probability of improvement than *Ext-norm-RND*. This indicates that decaying the scale of intrinsic rewards over time is also a viable method to alleviate the performance drop of RND in Atari tasks, with the notable drawback that the decay schedule has to be tuned beforehand. *Decoupled-RND* attains a probability of improvement that is higher than RND yet lower than both *Ext-norm-RND* and *Decay-RND*. EIPO exhibited the highest probability of improvement among all the baselines, which shows that EIPO is more likely to improve the performance of PPO across Atari games.

In addition, Fig. 3c shows the percentage of Atari games in which one method matches or exceeds the performance of another method with a pairwise comparison. Note that *"percentage of games $A \geq B$"* $R(A, B)$ is not equivalent to probability of improvements $P(A \geq B)$. Let μ_A be the mean score of algorithm A over 5 random seeds in an environment. $R(A, B)$ is defined as: $R(A, B) = \frac{\mathbb{1}\left[\mu_A \geq \mu_B\right]}{N}$ where N denotes the number of Atari games. The last row of Fig. 3c shows that EIPO performs better than all the baselines in the majority of games (i.e., $R(\text{EIPO}, B) > 0.5$).

4.3 Does EIPO perform better than the baselines?

As all our baselines improve upon vanilla RND, we compare EIPO with each baseline pairwise in terms of "strict probability of improvement". Strict probability of improvement $P(A > B)$ (Appendix A.4.1) measures the likelihood that an algorithm A is strictly better than an algorithm B. Fig. 3b shows $P(\textit{EIPO-RND} > B)\ \forall B$. It can be seen that EIPO outperforms all the baselines by

more than 50%. As suggested in [18], $P(\textit{EIPO-RND} > B) > 0.5$ with a confidence interval less than 0.5 indicates that a results has statistical significance. We can therefore conclude that EIPO will generally perform better than all other baselines across all 61 Atari games.



(a) (b) (c)

Figure 3: **(a)** *EIPO-RND* (ours) has a higher probability of improvement $P(\textit{EIPO-RND} \geq \text{PPO})$ over PPO than all other baselines. It suggests *EIPO-RND* is more likely to attain a higher score than PPO, compared to other methods. **(b)** Strict probability of improvement $P(\textit{EIPO-RND} > B) > 0.5\ \forall B$ indicates that *EIPO-RND* performs strictly better than the baselines in the majority of trials. **(c)** *EIPO-RND* performs better than PPO and all the other baselines in a majority of Atari games.

4.4 Does EIPO preserve the performance gain of RND?

We report the PPO-normalized score [18] (Appendix A.4) to validate whether EIPO preserves the performance gain from RND when applicable. Table 1 displays the environment-wise mean and 95%-confidence interval of PPO-normalized scores for each algorithm in the set of games where RND achieves a higher mean extrinsic return than PPO. Surprisingly, *EIPO-RND* yields a higher performance gain than RND does, which implies that EIPO not only mitigates the bias introduced by intrinsic rewards in Atari games, but can also improve performance.

Table 1: *EIPO-RND* even exhibited higher performance gain than RND in the games where RND is better than PPO.

Algorithm	PPO-normalized score Mean	(CI)
RND	384.57	(85.57, 756.69)
Ext-norm RND	427.08	(86.53, 851.52)
Decay-RND	383.83	(84.19, 753.17)
Decoupled-RND	1.54	(1.09, 2.12)
EIPO-RND	**435.56**	(109.45, 874.88)

4.5 Does EIPO outperform tuned RND?

We empirically find that the performance of RND is heavily dependent on the difference in magnitude between r^E and r^I. Because this difference can be controlled through the intrinsic scaling coefficient λ and the best scaling coefficient is not known a priori, we tune λ for RND in several games where RND substantially under-performs compared to PPO in order to see if EIPO can match or outperform the strongest tuned baseline. A variety of λ values are selected for each environment based on the relative difference in magnitudes between intrinsic rewards r^I and extrinsic rewards r^E in each environment (λ values are detailed in Appendix A.4.3). Fig. 4 shows the extrinsic return in 6 selected environments given each λ choice. Overall, no choice of λ consistently generates performance improvements over RND. For instance, $r^E \approx \lambda r^I$ substantially improves RND in *Jamesbond*, yet deteriorates RND in *YarsRevenge*. This shows that the range of acceptable constant λ values is highly environment-dependent, and could be narrow or potentially nonexistent. Moreover, the effect of λ is unpredictable. Intuitively, raising λ increases the weight of intrinsic rewards in the optimization objective and potentially introduces more bias, yet $r^E < \lambda r^I$ outperforms $r^E \gg \lambda r^I$ in an easy exploration game *Jamesbond* (based on taxonomy in [2]). In contrast, EIPO consistently performs better than PPO and RND using $\lambda = 1$ in all environments, demonstrating that EIPO is far less susceptible to an ill-tuned λ.

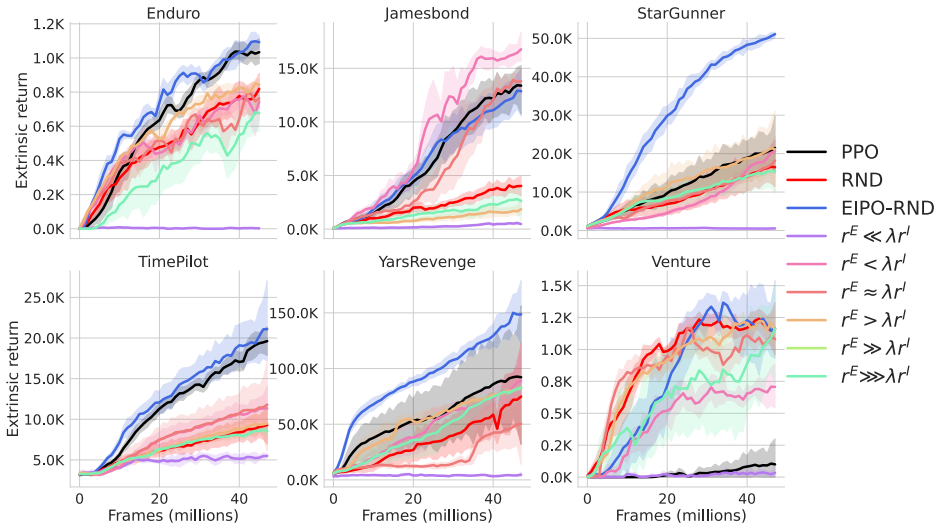


Figure 4: None of the intrinsic reward scaling coefficients λ consistently yield performance improvements over RND and PPO, whereas our method EIPO consistently outperforms RND and PPO without any tuning across environments. It indicates that our method is less susceptible to λ.

5 Discussion

Comparison with *Decay-RND* *Decay-RND* requires hand-tuning the decay schedule of λ to control the weighting between extrinsic r^E and intrinsic rewards r^I (i.e., $r^E + \lambda r^I$). By contrast, the Lagrangian multiplier α in EIPO adjusts the relative importance between both reward functions automatically. Fig. 5 shows that the EIPO-optimized trajectory of α during training is quite different across environments, indicating that the optimal decay schedule is highly environment dependent. As such, it is unlikely that there exists a single decay schedule that works well across all environments. Additionally, Fig. 5 shows that α decreases over time and yet still outperforms *Decay-RND* in a handful of environments (see the learning curve on *Carnival* in Appendix A.7). This implies that in certain domains it may be better to amplify the intrinsic reward rather than decay it.

Comparison with *Decoupled-RND* Though *Decoupled-RND* does not require tuning the decay schedule of λ, the experimental results in Section 4 show that *Decoupled-RND* does not improve upon *Ext-norm-RND*, and can even deteriorate *Ext-norm-RND* on *Montezuma's Revenge*. We hypothesize that this is because *Decoupled-RND* cannot adaptively tune the importance of intrinsic and extrinsic rewards during optimization. *Decoupled-RND* enforces the constraint that the exploration policy π_{E+I} perform similarly to the exploitation policy π_E, but if this constraint excessively penalizes π_{E+I} in the early stages of training, π_{E+I} will tend to degenerate to π_E and lose the exploration incentive generated by r^I. EIPO can adaptively adjust the trade off between both policies despite enforcing the KL-divergence constraint. Consequently, EIPO loosens the extrinsic optimality constraint at the start of training before tightening it later on. The curve of α w.r.t. time shown in Appendix A.7 validates our hypothesis, showing that α decreases first before increasing again in the middle of training.

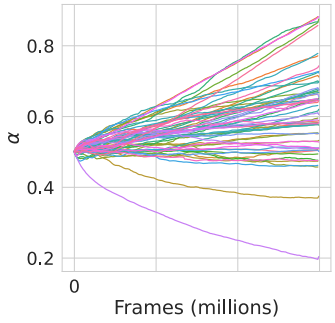


Figure 5: Each colored curve denotes the evolution of α in an environment when using EIPO. The α trajectories vary drastically across environments.

To conclude, we propose a theoretically motivated framework to mitigate the bias of intrinsic rewards. Our resulting algorithm, EIPO, exhibits consistent performance improvements over PPO while preserving the performance gains of curiosity-driven exploration on hard exploration tasks. Moreover, EIPO is able to adapt the importance of intrinsic rewards during optimization, reducing the need to hand-tune hyperparameters such as exploration schedules (ϵ-greedy), or λ decay (*Decay-RND*). While EIPO reduces the need for hand-tuning, a practitioner must still choose a good intrinsic reward formulation to start with. This limitation highlights an interesting avenue for future work, which could be to incorporate intrinsic reward function optimization into our framework, hopefully increasing the flexiblity of our method on a range of tasks.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

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

[3] Jurgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*, 1991.

[4] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

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

[6] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2778–2787, 2017.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

[8] Adrien Ali Taïga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

[9] William F Whitney, Michael Bloesch, Jost Tobias Springenberg, Abbas Abdolmaleki, Kyunghyun Cho, and Martin Riedmiller. Decoupled exploration and exploitation policies for sample-efficient reinforcement learning. *arXiv preprint arXiv:2101.09458*, 2021.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[12] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[13] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

[14] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[15] Thomas Stepleton. The pycolab game engine, 2017. *URL https://github. com/deepmind/pycolab*, 2017.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[17] Lukas Schäfer, Filippos Christianos, Josiah Hanna, and Stefano V Albrecht. Decoupling exploration and exploitation in reinforcement learning. *arXiv preprint arXiv:2107.08966*, 2021.

[18] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[19] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *International Conference of Representation Learning*, 2015.

[20] Cetin Meriçli, Tekin Meriçli, and H Levent Akin. A reward function generation method using genetic algorithms: a robot soccer case study. In *AAMAS*, pages 1513–1514, 2010.

[21] Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23, 2010.

[22] Xiaoxiao Guo, Satinder Singh, Richard Lewis, and Honglak Lee. Deep learning for reward design to improve monte carlo tree search in atari games. *arXiv preprint arXiv:1604.07095*, 2016.

[23] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

[24] Zeyu Zheng, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado Van Hasselt, David Silver, and Satinder Singh. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, pages 11436–11446. PMLR, 2020.

[25] Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

[26] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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] See the supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. A.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. A.3
   (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] See Sec. A.4
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]