SOLVING SPARSE-REWARD ENVIRONMENTS USING META-GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration is essential in reinforcement learning since agents rely on trial and error to learn an optimal policy. However, when rewards are sparse, naive exploration strategies, like noise injection, are often insufficient. Intrinsic rewards can provide principled guidance for exploration by combining them with extrinsic rewards to optimize a policy or using them to train subpolicies for hierarchical learning. However, the former approach induces suboptimality as it changes the performance objective, while the latter is sample inefficient and also suboptimal. To address these challenges, we propose an algorithm, intrinsic reward policy optimization (IRPO), that uses intrinsic rewards within a meta-gradient framework to drive exploration without changing the performance objective. We achieve this by updating a shared meta-policy such that one of the subpolicies (derived from the meta-policy using intrinsic rewards) optimizes extrinsic rewards. We demonstrate that IRPO outperforms relevant baselines in sparse-reward environments with similar wall-clock time to standard policy gradient methods, and justify IRPO design choices through ablation studies. We also prove that IRPO converges to an ϵ-approximate first-order stationary point.

1 INTRODUCTION

Reinforcement learning (RL) aims to learn an optimal policy through trial and error, making exploration a key challenge, particularly in sparse-reward environments. Many exploration strategies have been proposed, such as noise injection in the action space Sutton et al. (1999a); Schulman et al. (2017); Haarnoja et al. (2018) or in the policy parameter space Fortunato et al. (2017); Plappert et al. (2018) (see Figure 1b). However, these methods do not induce sufficient diversity in collected experiences in widely used complex sparse-reward environments.

Intrinsic rewards offer an alternative that provides principled guidance for exploration. One approach incentivizes agents to visit states with high uncertainty by maximizing the sum of extrinsic rewards and uncertainty-based intrinsic rewards (see Figure 1c). Common intrinsic rewards include those characterized by state visitation Bellemare et al. (2016); Tang et al. (2017); Machado et al. (2020), policy entropy Schulman et al. (2017); Haarnoja et al. (2018), or model uncertainty Burda et al. (2018); Yang et al. (2024). These methods enhance exploration, but are suboptimal (they change the performance objective Chen et al. (2022); Ladosz et al. (2022)) and can struggle to scale to continuous and high-dimensional state spaces (e.g., when using count-based state visitation Ladosz et al. (2022)). Alternatively, hierarchical approaches use pretrained subpolicies—typically based on diffusion-maximizing intrinsic rewards, like those derived by decomposing the graph Laplacian Machado et al. (2017a); Wu et al. (2018); Gomez et al. (2023)—as temporally extended actions Sutton et al. (1999b); Machado et al. (2023). Hierarchical RL does not change the performance objective (see Figure 1d), but using subpolicies as temporally extended actions induces suboptimality Sutton et al. (1999b) and sample inefficiency (due to subpolicy pretraining).

To address these challenges, we propose an algorithm, intrinsic reward policy optimization (IRPO), that uses intrinsic rewards within a meta-gradient framework to drive exploration without changing the performance objective. As shown in Figure 1a, at each iteration, IRPO derives multiple subpolicies from a shared meta-policy, where each subpolicy optimizes its own intrinsic reward. We then update the meta-policy such that one of the final subpolicies optimizes the extrinsic reward. We achieve this by using meta-gradients computed by backpropagating gradients of the final sub-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

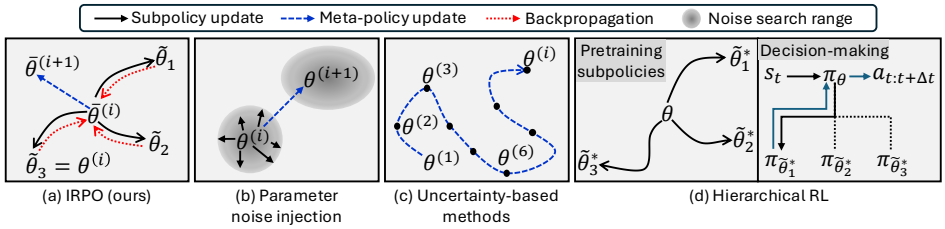

Figure 1: (a) IRPO optimizes multiple subpolicies (from the current meta-policy $\pi_{\tilde{\theta}^{(i)}}$) using intrinsic rewards and uses their gradients to update the meta-policy via backpropagation. Then, IRPO selects the best-performing subpolicy as an evaluation policy, i.e., $\theta^{(i)} = \tilde{\theta}_3$. This approach does not change the original performance objective while driving exploration through intrinsic rewards. (b) Parameter noise injection generates nearby policies by adding random noise and uses their gradients to update the evaluation policy. (c) Uncertainty-based methods augment extrinsic rewards with intrinsic rewards based on various forms of uncertainty. (d) Hierarchical RL pretrains subpolicies (based on intrinsic rewards) and uses them as temporally extended actions.

policies, evaluated with respect to extrinsic rewards. Thus, we can drive exploration using intrinsic rewards without inheriting the suboptimality of prior methods. Since we do not pretrain subpolicies, IRPO also avoids the sample inefficiency of hierarchical RL.

2 Background

We consider a Markov decision process (MDP) Puterman (2014) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to [0, R_{\max}]$ is the reward function bounded by $R_{\max} \in \mathbb{R}_+$, and $\gamma \in [0, 1)$ is the discount factor. At time step t, the agent executes an action $a_t \in \mathcal{A}$ given the current state $s_t \in \mathcal{S}$, after which the system transitions to state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and the agent receives reward $r_t = R(s_t, a_t)$. The agent acts following a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Let $g_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ be the return, $V_R^\pi(s) = \mathbb{E}_\pi [g_t \mid s_t = s]$ be the state-value function, and $Q_R^\pi(s, a) = \mathbb{E}_\pi [g_t \mid s_t = s, a_t = a]$ be the action-value function. Policy gradient methods Sutton et al. (1999a) optimize a policy π_θ parameterized by $\theta \in \mathbb{R}^m$ by performing gradient ascent on the performance objective $J_R(\theta) = \mathbb{E}_{d_0} [V_R^{\pi_\theta}(s_0)]$, where d_0 is the initial state distribution. Specifically, they perform the following update:

$$\theta^{(i+1)} = \theta^{(i)} + \eta \nabla_{\theta^{(i)}} J_R(\theta^{(i)}), \tag{1}$$

where i is the current iteration, η is a learning rate, $\nabla_\theta J_R(\theta) = \mathbb{E}_{d^{\pi_\theta}} [Q_R^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a \mid s)]$ is the policy gradient, and d^{π_θ} is the discounted state occupancy induced by π_θ.

One can express the policy gradient in a variance-reduced form as follows Schulman et al. (2018):

$$\nabla_\theta J_R(\theta) = \mathbb{E}_{d^{\pi_\theta}} [A_R^{\pi_\theta}(s, a) \cdot \nabla_\theta \log \pi_\theta(a \mid s)], \tag{2}$$

where $A_R^{\pi_\theta}$ is the advantage function. The advantage function can be estimated using generalized advantage estimation $\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$, where $\lambda \in [0, 1]$ is a hyperparameter and $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$. One can update V_ϕ as follows:

$$\phi^{(i+1)} = \phi^{(i)} - \eta_c \nabla_{\phi^{(i)}} \mathbb{E}_{d^{\pi_\theta}} \left[\left(\hat{V}(s_t) - V_{\phi^{(i)}}(s_t) \right)^2 \right], \tag{3}$$

where η_c is the critic learning rate and $\hat{V}(s_t) := \hat{A}_t + V_{\phi^{(i)}}(s_t)$ is the value target. We simplify notation by dropping the policy term π_θ from value functions.

3 Our Algorithm: Intrinsic Reward Policy Optimization

Our algorithm uses K diffusion-maximizing intrinsic reward functions $\{\tilde{R}_k\}_{k=1}^K$, generated using the augmented Lagrangian Laplacian objective (ALLO) Gomez et al. (2023)—these intrinsic rewards encourage large changes in the state space. Each intrinsic reward $\tilde{R}_k$ is associated with a

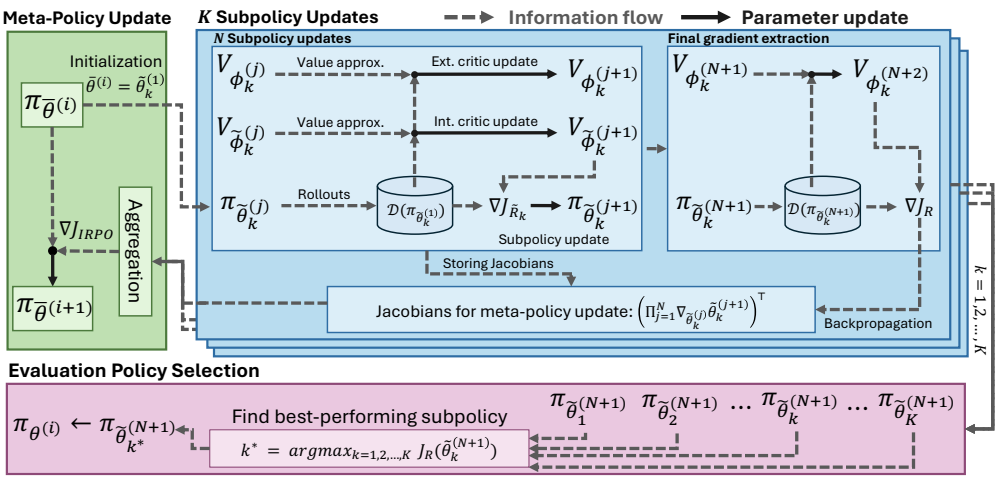

Figure 2: We pretrain K diffusion-maximizing intrinsic rewards a priori. Each subpolicy, paired with an intrinsic reward, uses an actor–critic framework with an intrinsic critic to update the subpolicy and an extrinsic critic to update the shared meta-policy. Finally, we select the best-performing final subpolicies as the evaluation policy.

subpolicy $\pi_{\tilde{\theta}_k}$. Building on the actor-critic framework, each subpolicy $\pi_{\tilde{\theta}_k}$ is associated with an intrinsic critic $V_{\tilde{\phi}_k}$, which approximates the value of $\pi_{\tilde{\theta}_k}$ with respect to intrinsic reward $\tilde{R}_k$, and an extrinsic critic V_{ϕ_k}, which approximates the value of $\pi_{\tilde{\theta}_k}$ with respect to the extrinsic reward R. We use intrinsic critics for subpolicy updates and extrinsic critics for meta-policy updates.

We first pretrain the diffusion-maximizing intrinsic rewards (Section 3.1). Then, for each iteration, IRPO performs the following: (1) updates subpolicies from the shared meta-policy (Section 3.2), (2) updates the meta-policy using experiences of the subpolicies (Section 3.3), and (3) selects the best-performing subpolicy as the evaluation policy (Section 3.4). At a high-level, IRPO uses subpolicies to explore and a meta-gradient (which we call the IRPO gradient) to update the meta-policy such that one of the final subpolicies optimizes the extrinsic reward. Figure 2 visualizes one iteration of IRPO, and Algorithm 1 summarizes the overall algorithm.

3.1 Pretraining Diffusion-Maximizing Intrinsic Rewards

Following Gomez et al. (2023), we first collect 200,000 samples by running a random-walk policy for $200{,}000/H$ episodes of horizon H, with initial states sampled uniformly from $\mathcal{S}$, i.e., $s_0 \sim \text{Unif}(\mathcal{S})$. We then train a neural network that takes the state as input and outputs a $(K/2)$-dimensional vector, where the positive and negative directions of each dimension define two intrinsic rewards (totaling K), by optimizing a surrogate of the graph-drawing objective using the collected samples (see Gomez et al. (2023) for details). We discuss our implementation details and visualize the resulting ALLO intrinsic rewards in Appendix F.

3.2 Subpolicy Updates (Inner-Updates)

Given the current meta-policy parameters $\bar{\theta}$, we first initialize subpolicies as duplicates, such that $\tilde{\theta}_k^{(1)} = \bar{\theta}^{(i)}$ for $k = 1, 2, \ldots, K$. Intrinsic and extrinsic critics for those subpolicies are initialized using their parameters from the last iteration. We then perform N updates to each subpolicy using its own intrinsic rewards.

Updating subpolicies. For the j-th update for subpolicy $\pi_{\tilde{\theta}_k^{(j)}}$, where $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, K$, we first collect rollouts from $\pi_{\tilde{\theta}_k^{(j)}}$. Each rollout contains experiences of the form $(s_t, a_t, r_t, \tilde{r}_{t,k})$, where $\tilde{r}_{t,k} = \tilde{R}_k(s_t, a_t)$. We use these experiences to update the intrinsic critic $V_{\tilde{\phi}_k^{(j)}}$ (based on intrinsic rewards) and the extrinsic critic $V_{\phi_k^{(j)}}$ (based on extrinsic rewards) for

Algorithm 1 IRPO: Intrinsic reward policy optimization

Initialize: Meta-policy $\pi_{\bar{\theta}}$ and K critic pairs $\{(V_{\tilde{\phi}_k}, V_{\phi_k})\}_{k=1}^{K}$
Pretrain K diffusion-maximizing intrinsic rewards $\{\tilde{R}_k\}_{k=1}^{K}$
for update $i = 1, 2, \ldots, T$ **do**
  `/* subpolicy updates */`
  Initialize subpolicies $\pi_{\tilde{\theta}_k^{(1)}} = \pi_{\bar{\theta}^{(i)}}$ for all $k = 1, 2, \ldots, K$
  for update $j = 1, 2, \ldots, N$ **do**
    for all k **do:** Collect rollouts from $\pi_{\tilde{\theta}_k^{(j)}}$ and update critics $V_{\tilde{\phi}_k}, V_{\phi_k}$
    for all k **do:** Update subpolicy $\pi_{\tilde{\theta}_k^{(j)}}$ and store Jacobian $\nabla_{\tilde{\theta}_k^{(j)}} \tilde{\theta}_k^{(j+1)}$
  end for
  `/* meta-policy update */`
  for all k **do:** Collect rollouts using $\pi_{\tilde{\theta}_k^{(N+1)}}$ and compute gradient $\nabla_{\tilde{\theta}_k^{(N+1)}} J_R(\tilde{\theta}_k^{(N+1)})$
  Compute the IRPO gradient via Equation (5) and update meta-policy via Equation (10)
  `/* evaluation policy selection */`
  $\theta^{(i)} = \tilde{\theta}_{k^*}^{(N+1)}$, where $k^* = \arg\max_{k=1,2,\ldots,K} J_R(\tilde{\theta}_k^{(N+1)})$
end for

each subpolicy using Equation (3). We use the intrinsic critic to estimate the policy gradient using Equation (2), which is used to update the subpolicy using Equation (1). We use the final extrinsic critic $V_{\phi_k^{(N+2)}}$ to update the meta-policy as discussed in Section 3.3.

Storing Jacobians for the meta-policy update. For each subpolicy update $j = 1, 2, \ldots, N$, we calculate and store the Jacobian between consecutive subpolicy parameters as follows:

$$\nabla_{\tilde{\theta}_k^{(j)}} \tilde{\theta}_k^{(j+1)} = \frac{\partial}{\partial \tilde{\theta}_k^{(j)}} \left(\tilde{\theta}_k^{(j)} + \tilde{\eta} \nabla_{\tilde{\theta}_k^{(j)}} J_{\tilde{R}_k}(\tilde{\theta}_k^{(j)}) \right) = I + \tilde{\eta} \nabla^2_{\tilde{\theta}_k^{(j)}} J_{\tilde{R}_k}(\tilde{\theta}_k^{(j)}), \tag{4}$$

where $\tilde{\eta}$ is the learning rate used for subpolicy updates, $J_{\tilde{R}_k}$ denotes the performance objective with respect to intrinsic reward $\tilde{R}_k$, and I is the identity matrix. We use this Jacobian to backpropagate the subpolicy gradients to update the meta-policy, as discussed in Section 3.3.

3.3 Meta-Policy Updates (Outer-Updates)

After performing N updates to each subpolicy, we update the meta-policy parameters by estimating the IRPO gradient defined in Equation (5) and using it in the trust-region update defined in Equation (10). For brevity, we denote the final subpolicy for intrinsic reward $\tilde{R}_k$ as $\tilde{\theta}_k = \tilde{\theta}_k^{(N+1)}$.

IRPO gradient. We define the IRPO gradient as a weighted sum of the backpropagated gradients:

$$\nabla J^{\tau}_{\text{IRPO}}(\bar{\theta}, \{\tilde{\theta}_k\}_{k=1}^{K}) \coloneqq \sum_{k=1}^{K} \omega_k^{\tau} \cdot \nabla_{\bar{\theta}} J_R(\tilde{\theta}_k), \tag{5}$$

where the weights ω_k^{τ} are given by a temperature-scaled softmax over subpolicy performance:

$$\omega_k^{\tau} \coloneqq \frac{\exp(J_R(\tilde{\theta}_k)/\tau)}{\sum_{k'=1}^{K} \exp(J_R(\tilde{\theta}_{k'})/\tau)}, \tag{6}$$

with $\tau \in (0, 1]$ being a temperature parameter. We anneal the temperature parameter $\tau \to 0_+$ over the course of training to emphasize exploitation (rather than exploration) as training progresses; we study the impact of this annealing in Section 4.2. The backpropagated gradients are defined as:

$$\nabla_{\bar{\theta}} J_R(\tilde{\theta}_k) = \mathbb{E}_{d^{\pi_{\tilde{\theta}_k}}} \left[Q_R^{\pi_{\tilde{\theta}_k}}(s, a) \, \nabla_{\bar{\theta}} \log \pi_{\tilde{\theta}_k}(a \mid s) \right], \tag{7}$$

where the gradient of the logarithm of the policy is defined as follows:

$$\nabla_{\bar{\theta}} \log \pi_{\tilde{\theta}_k}(a \mid s) = (\nabla_{\bar{\theta}} \tilde{\theta}_k)^{\top} \nabla_{\tilde{\theta}_k} \log \pi_{\tilde{\theta}_k}(a \mid s). \tag{8}$$

Estimating the IRPO gradient with backpropagation. We estimate the IRPO gradient by collecting rollouts from each final subpolicy and using those rollouts to estimate the policy gradients for each final subpolicy with respect to the extrinsic rewards $\nabla_{\tilde{\theta}_k} J_R(\tilde{\theta}_k)$, using Equation (2). These gradients are then backpropagated to estimate $\nabla_{\bar{\theta}} J_R(\tilde{\theta}_k)$ by applying the chain rule using the stored Jacobians from Section 3.2 as follows:

$$\nabla_{\bar{\theta}} J_R(\tilde{\theta}_k) = \left(\prod_{j=1}^{N} \nabla_{\tilde{\theta}_k^{(j)}} \tilde{\theta}_k^{(j+1)} \right)^{\top} \cdot \nabla_{\tilde{\theta}_k} J_R(\tilde{\theta}_k). \tag{9}$$

We discuss the practical computation of this gradient in Appendix C.2.

Trust-region update. We update the meta-policy with a trust-region update Schulman et al. (2015):

$$\bar{\theta}^{(i+1)} = \bar{\theta}^{(i)} + \sqrt{\frac{2\delta_{\text{KL}}}{\nabla J_{\text{IRPO}}^{\top} F_{\bar{\theta}^{(i)}}^{-1} \nabla J_{\text{IRPO}}}} F_{\bar{\theta}^{(i)}}^{-1} \nabla J_{\text{IRPO}}, \tag{10}$$

where $F_{\bar{\theta}} = \mathbb{E}_{(s,a)\sim d^{\pi_{\bar{\theta}}}} \left[\nabla_{\bar{\theta}} \log \pi_{\bar{\theta}}(a \mid s) (\nabla_{\bar{\theta}} \log \pi_{\bar{\theta}}(a \mid s))^{\top} \right]$ is the Fisher information matrix (FIM) of the meta-policy $\pi_{\bar{\theta}}$ and $\nabla J_{\text{IRPO}} \coloneqq \nabla J_{\text{IRPO}}(\bar{\theta}^{(i)}, \{\tilde{\theta}_k\}_{k=1}^{K})$.

3.4 Evaluation Policy Selection

After performing subpolicy updates (which yield K subpolicies $\{\pi_{\tilde{\theta}_k^{(N+1)}}\}_{k=1}^{K}$) and a single meta-policy update, we select the best-performing subpolicy—that is, the one that achieves the highest performance on J_R—as the evaluation policy for that iteration. Intuitively, as the temperature is annealed to zero, updating the meta-policy with the IRPO gradient guides it toward a policy from which one final subpolicy (derived from N subpolicy updates) maximizes J_R.

3.5 Empirical and Theoretical Analysis

We illustrate IRPO's update mechanism in Figure 3, where we used a two-dimensional parameter $\theta \in \mathbb{R}^2$, with the following (strictly concave) extrinsic and intrinsic objectives: $J(\theta) \coloneqq -\|\theta\|_2^2$, $\tilde{J}_1(\theta) \coloneqq -\left\|\theta - [0, -2]^{\top}\right\|_2^2$. As expected, the subpolicy updates (red arrows) move towards the intrinsic optimum (red star), and meta-policy updates steer the meta-policy so that N subpolicy updates from it land at the extrinsic optimum. We also see that increasing the number of subpolicy updates N increases exploration (indicated by the length of red arrows). We provide similar analysis for multiple intrinsic rewards with varying temperature parameters τ in Appendix D.

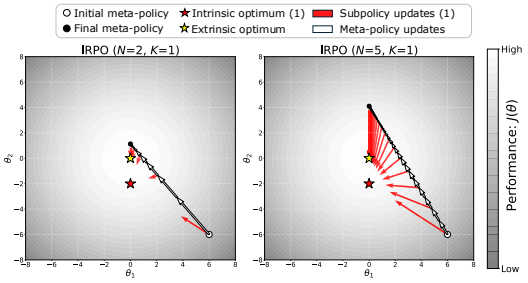

Figure 3: IRPO's update mechanism with a single intrinsic reward (i.e., $K = 1$). We plot results for different numbers of subpolicy updates with the extrinsic objective plotted on a contour.

We study the convergence properties of IRPO under exact gradients. Let an iteration be defined as one meta-policy update, which includes $K \times N$ subpolicy updates and an evaluation policy selection, and let the sum of N subpolicy gradients for the k-th intrinsic reward function be defined as follows:

$$G_k(\bar{\theta}) \coloneqq \sum_{j=1}^{N} \nabla_{\tilde{\theta}_k^{(j)}} J_{\tilde{R}_k}(\tilde{\theta}_k^{(j)}), \tag{11}$$

where $\tilde{\theta}_k^{(1)} = \bar{\theta}$ and $\tilde{\theta}_k^{(j+1)} = \tilde{\theta}_k^{(j)} + \tilde{\eta} \nabla_{\tilde{\theta}_k^{(j)}} J_{\tilde{R}_k}(\tilde{\theta}_k^{(j)})$. We assume the following.

Assumption 1 (Lipschitz Continuity)**.** Let G_k be defined as in Equation (11). Assume Lipschitz continuity of G_k and ∇G_k for all $k = 1, 2, \ldots, K$, and of the policy gradient ∇J_R on $\mathbb{R}^m$:

(a) There exists a constant $L_G > 0$ such that for all $k = 1, 2, \ldots, K$ and policy parameters $\bar{\theta}_1, \bar{\theta}_2 \in \mathbb{R}^m$: $\|G_k(\bar{\theta}_1) - G_k(\bar{\theta}_2)\|_2 \leq L_G \|\bar{\theta}_1 - \bar{\theta}_2\|_2$.

(b) There exists a constant $L'_G > 0$ such that for all $k = 1, 2, \ldots, K$ and policy parameters $\bar{\theta}_1, \bar{\theta}_2 \in \mathbb{R}^m$: $\|\nabla G_k(\bar{\theta}_1) - \nabla G_k(\bar{\theta}_2)\|_2 \leq L'_G \|\bar{\theta}_1 - \bar{\theta}_2\|_2$.

(c) There exists a constant $L_J > 0$ such that for all policy parameters $\theta_1, \theta_2 \in \mathbb{R}^m$: $\|\nabla J_R(\theta_1) - \nabla J_R(\theta_2)\|_2 \leq L_J \|\theta_1 - \theta_2\|_2$.

Assumption 2 (Positive Definiteness and Uniform Boundedness of FIM)**.** Assume that the inverse of the FIM $F_{\bar{\theta}}$ of the meta-policy is positive definite and uniformly bounded for all policy parameters $\bar{\theta} \in \mathbb{R}^m$ as follows:

$$\underline{\mu} I \preceq F_{\bar{\theta}}^{-1} \preceq \overline{\mu} I, \quad \forall \bar{\theta} \in \mathbb{R}^m, \quad \text{where } 0 < \underline{\mu} \leq \overline{\mu} < \infty. \tag{12}$$

This is a widely adopted assumption in the literature (e.g., Yuan et al. (2022); Bai et al. (2023)).

Assumption 3 (Scaled and Bounded Trust-Region Threshold)**.** Assume that the trust-region threshold δ_{KL} scales inversely with the total number of iterations T and is bounded from above as follows:

$$0 < \delta_{\text{KL}} = \frac{c}{T} < \frac{2\underline{\mu}(1 - \tilde{\eta} L_G)^2}{(\tilde{\eta} L'_G \overline{\mu})^2}, \tag{13}$$

where $c > 0$ is a constant that satisfies the inequality in Equation (13).

A similar threshold is employed in Shani et al. (2020) for convergence analysis of a trust region update.

Unlike standard policy gradient methods that directly optimize the evaluation policy via gradient ascent as in Equation (1), IRPO does so indirectly. As shown in Figure 4, IRPO updates the meta-policy, derives K subpolicies, and selects the best-performing subpolicy as the new evaluation policy $\theta^{(i)}$. Consequently, the evaluation policy does not follow a direct gradient ascent update, and it is therefore not obvious that IRPO theoretically converges like other gradient ascent methods Nesterov (2013); Xu et al. (2019); Shani et al. (2020). However, we show in Proposition 4 that we can in fact prove convergence to an ϵ-approximate first-order stationary point for IRPO.

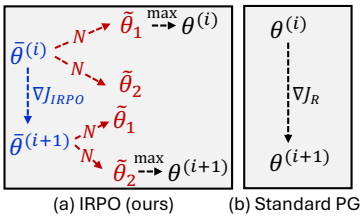

Figure 4: (a) how IRPO updates the evaluation policy (for $K = 2$). (b) a standard policy gradient update of the evaluation policy.

Proposition 4 (Convergence to an ϵ-Approximate First-Order Stationary Point)**.** *Let the temperature parameter be annealed as $\tau \to 0_+$ and the learning rate used for subpolicy updates satisfy $\tilde{\eta} < L_G^{-1}$. Under Assumptions 1 to 3, IRPO finds an ϵ-approximate first-order stationary point of J_R in $\mathbb{R}^m$ (i.e., a policy $\theta \in \mathbb{R}^m$ satisfying $\|\nabla J_R(\theta)\|_2 \leq \epsilon$) within $\mathcal{O}(1/\epsilon^2)$ updates.*

We provide the proof in Appendix A. The result in Proposition 4 matches that of standard first-order gradient methods on smooth nonconvex objectives Nesterov (2013); Agarwal et al. (2021).

4 Experiments

We compare IRPO to relevant baselines in discrete and continuous sparse-reward environments commonly used in the literature Machado et al. (2017a); Wu et al. (2018); Wang et al. (2021); Yang et al. (2024). We show the environments in Figure 5 and provide details in Appendix B. We study the impact of different intrinsic reward choices in Section 4.2. We consider the following baselines: (1) parameter space noise for exploration (PSNE) Plappert et al. (2018), (2) PPO Schulman et al. (2017) (3) PPO with distributional random network distillation (DRND) Yang et al. (2024), (4) PPO with one ALLO intrinsic reward, (5) hierarchical RL (HRL) with ALLO intrinsic rewards Machado et al. (2023); Gomez et al. (2023), (6) PPO with learning intrinsic rewards for policy gradient (LIRPG) intrinsic rewards Zheng et al. (2018), and (7) model-agnostic meta-learning (MAML) Finn et al. (2017). We used a Bayesian hyperparameter search to tune all algorithms. We justify our baselines and discuss algorithm implementation details in Appendix C.

4.1 Main Results

Environment	IRPO-ALLO (ours)	PSNE	PPO	PPO-DRND	PPO-ALLO	HRL-ALLO	PPO-LIRPG	MAML
Maze-v1	**0.83 ± 0.00**	0.17 ± 0.25	0.66 ± 0.25	0.74 ± 0.19	0.20 ± 0.30	0.82 ± 0.00	0.00 ± 0.00	0.02 ± 0.04
Maze-v2	**0.83 ± 0.00**	0.17 ± 0.25	0.33 ± 0.31	0.58 ± 0.28	0.40 ± 0.37	**0.83 ± 0.00**	0.00 ± 0.00	0.05 ± 0.11
FourRooms	**0.88 ± 0.00**	0.53 ± 0.32	0.70 ± 0.26	**0.88 ± 0.00**	**0.88 ± 0.00**	0.73 ± 0.09	0.20 ± 0.30	0.05 ± 0.11
PointMaze-v1	**0.85 ± 0.02**	0.23 ± 0.27	0.35 ± 0.33	0.40 ± 0.37	0.40 ± 0.37	0.79 ± 0.03	0.79 ± 0.30	0.34 ± 0.26
PointMaze-v2	**0.85 ± 0.01**	0.26 ± 0.27	0.44 ± 0.33	0.43 ± 0.33	0.30 ± 0.35	0.72 ± 0.18	0.30 ± 0.35	0.24 ± 0.20
FetchReach	**0.86 ± 0.02**	0.59 ± 0.35	0.75 ± 0.29	0.10 ± 0.07	0.13 ± 0.06	0.10 ± 0.22	0.10 ± 0.06	0.14 ± 0.05
AntMaze-v1	**0.99 ± 0.00**	**0.99 ± 0.00**	0.59 ± 0.67	0.39 ± 0.62	0.25 ± 0.32	0.71 ± 0.39	0.00 ± 0.00	-1.00 ± 0.00
AntMaze-v2	**0.97 ± 0.01**	0.79 ± 0.55	0.39 ± 0.67	0.16 ± 0.39	0.24 ± 0.28	0.69 ± 0.38	0.00 ± 0.00	-0.98 ± 0.03
AntMaze-v3	**0.97 ± 0.01**	0.00 ± 0.00	0.20 ± 0.55	0.07 ± 0.20	0.36 ± 0.20	0.61 ± 0.61	0.00 ± 0.00	-1.00 ± 0.00

Table 1: Peak evaluation return with mean and 95% confidence intervals over 10 random seeds.

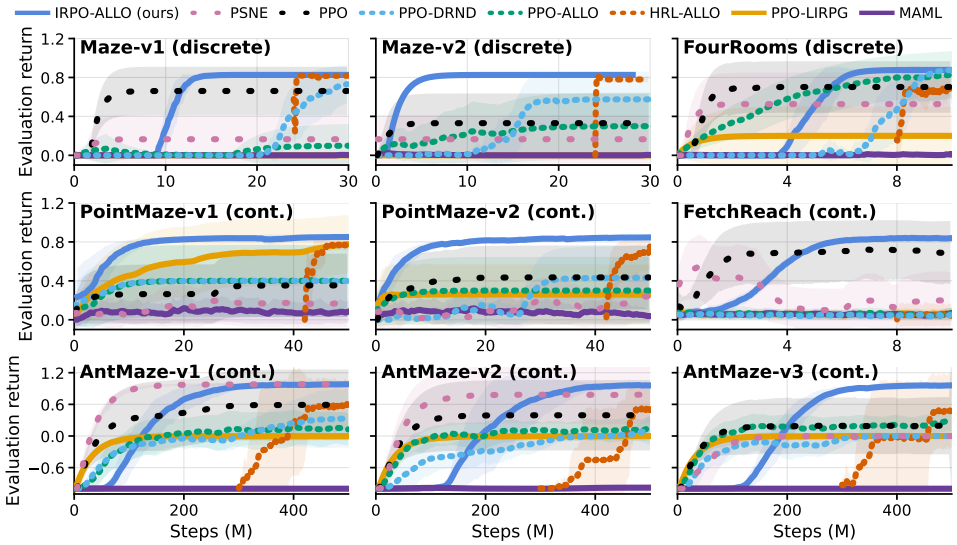

Figure 6: Learning curves with mean and 95% confidence intervals over 10 random seeds.

We show peak performance in Table 1 and learning curves in Figure 6. The learning curves include all steps (samples) used to generate ALLO intrinsic rewards (200,000 samples), perform subpolicy updates (for IRPO), and pretrain subpolicies (for HRL), which delays the curves of IRPO and HRL.

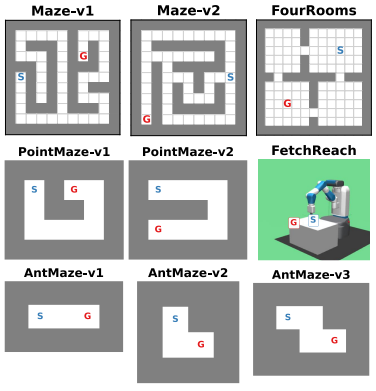

Figure 5: We consider three discrete environments (top row) and six continuous ones (middle and bottom rows).

We observe that IRPO outperforms all baselines (achieving the highest peak performance with the smallest confidence intervals, typically with fewer samples) in all environments, except PSNE in *AntMaze-v1*. This is likely because IRPO provides sufficient exploration (via intrinsic reward-driven subpolicy updates) while still optimizing for the original performance objective (via extrinsic reward-driven meta-policy updates). PSNE and PPO—which directly optimize extrinsic rewards—do not achieve consistently strong performance across environments, likely due to insufficient exploration. We also see that adding DRND and ALLO intrinsic rewards to PPO actually degrades performance, likely due to the suboptimality introduced by reward augmentation. HRL-ALLO does achieve relatively strong performance in many environments, though it consistently underperforms IRPO (both in peak evaluation return and sample efficiency). Moreover, HRL-ALLO fails in *FetchReach*, whereas IRPO succeeds using the same intrinsic rewards. We attribute this to HRL's use of temporally extended actions, which limits fine-grained decision-making. PPO-LIRPG—a meta-gradient exploration algorithm—shows poor performance in all environments except *PointMaze-v1*, because its learned intrinsic rewards depend on extrinsic rewards, which are sparse in these environments. MAML, which differs from IRPO in its backpropagated gradients and evaluation policy selection,

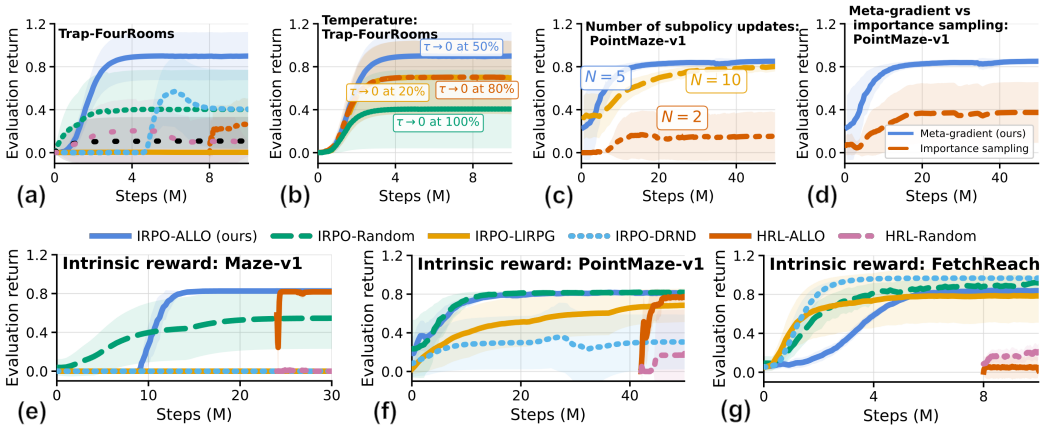

Figure 7: (a) Results for IRPO and baselines in *Trap-FourRooms* (legend matches Figure 6). (b) Temperature ablation for IRPO in *Trap-FourRooms*. (c) Number of subpolicy updates ablation for IRPO. (d) Meta-gradient comparison to importance sampling for IRPO. (e-g) Comparing intrinsic rewards for IRPO and HRL. All curves show mean and 95% confidence intervals over 10 seeds.

fails in all environments, showing that this distinction is key to solving sparse-reward environments. We provide a computational complexity analysis for IRPO in Appendix E, which shows that IRPO achieves similar wall-clock time to PPO and the relationship between the number of intrinsic rewards, the number of subpolicy updates, GPU memory usage, and wall-clock time.

4.2 Ablation Study

We show ablation results in Figure 7. We first evaluate IRPO's robustness to rewarding trap states in a *Trap-FourRooms* environment, created by adding six absorbing trap states (each with a reward of $+0.01$) to the top-left room of *FourRooms* and blocking the path from the top-left room to the bottom-left room (see Figure 8 in Appendix). Figure 7a shows that IRPO outperforms all baselines in *Trap-FourRooms*, due its ability to simultaneously explore different states. Figure 7b shows that IRPO's temperature annealing speed has a nonmonotonic impact on performance, where the highest performance corresponds to $\tau \to 0$ at 50% of the total training steps. This suggests that annealing speed governs an exploration–exploitation trade-off, similar to ϵ-greedy sampling Sutton et al. (1998). Figure 7c shows the impact of the number of subpolicy updates N, which also controls exploration. We see that $N = 2$ results in poor performance, likely due to insufficient exploration, while $N = 10$ results in similar converged performance to $N = 5$ but with more samples required.

Figure 7d compares our meta-gradient update to an alternative importance-sampling (IS) update using experiences of subpolicies, evaluated as $\nabla_\theta J_R(\theta) \approx \mathbb{E}_{d^{\pi_{\tilde{\theta}_k}}}[\rho(s,a)\, Q_R^{\pi_{\tilde{\theta}_k}}(s,a)\, \nabla_\theta \log \pi_\theta(a|s)]$, where $\rho(s,a) \coloneqq \pi_\theta(a|s)/\pi_{\tilde{\theta}_k}(a|s)$ denotes the IS ratio. The approximation sign arises from neglecting the discounted state-occupancy correction term, $d^{\pi_\theta}(s)/d^{\pi_{\tilde{\theta}_k}}(s)$, and from using a subpolicy critic (which we found to empirically yield better performance than using the policy critic). We observe that IS updates result in poor performance and high confidence intervals. We attribute this to the high variance of IS gradient estimates—as the action distributions of π_θ and $\pi_{\tilde{\theta}}$ diverge, the IS ratio ρ has more variance Sutton et al. (1998).

Finally, Figure 7(e-g) compares implementations of IRPO with different intrinsic rewards (LIRPG, DRND, and random). We also compare to HRL with random intrinsic rewards. We do not consider HRL with LIRPG and DRND intrinsic rewards because they need continuous updates throughout training and thus cannot be pretrained. We generated random rewards by using a randomly initialized neural network to map states to $(K/2)$-dimensional vectors, where the positive and negative directions of each dimension define two intrinsic rewards (totaling K and see the resulting rewards in Figure 11 in the Appendix). Figure 7e shows that random intrinsic rewards degraded the performance of IRPO in *Maze-v1*; IRPO was significantly more robust to random intrinsic rewards than HRL though (which failed with random intrinsics). However, IRPO struggled in *Maze-v1* with DRND intrinsic rewards, where PPO-DRND performed better, suggesting that IRPO is sensitive

to the choice of intrinsic rewards. Nonetheless, Figure 7g shows that IRPO-LIRPG and IRPO-DRND solved *FetchReach*, which PPO-LIRPG and PPO-DRND were unable to solve. Interestingly, IRPO-LIRPG, IRPO-DRND, and IRPO-Random outperformed IRPO-ALLO in *FetchReach*; HRL-Random similarly outperformed HRL-ALLO. We attribute this to the selective benefits of ALLO intrinsic rewards, which are effective for navigation in complex maze-like environments but less applicable to simpler environments. Moreover, IRPO-Random achieved somewhat reasonable performance in *Maze-v1* and *PointMaze-v1*, although it underperformed IRPO-ALLO, whereas HRL-Random significantly degraded relative to HRL-ALLO. These results suggest that IRPO is better than HRL at using intrinsic rewards for exploration and at solving the environments.

5 Related Work

Parameter noise injection enables consistent, structured exploration and has been shown to outperform action noise injection methods Fortunato et al. (2017); Plappert et al. (2018). However, Figure 6 shows that the method remains limited in sparse-reward environments. Moreover, the method scales poorly with neural-network size Plappert et al. (2018) and can introduce additional variance into policy gradient estimation Cohen et al. (2019).

Uncertainty-based methods use uncertainty in the environment or the agent to construct intrinsic rewards that guide the agent toward uncertain states by augmenting the extrinsic reward with these rewards. Schulman et al. (2017); Haarnoja et al. (2017) use entropy regularization to prevent the policy's stochasticity from collapsing during policy optimization and thereby promote continuous exploration, assuming uncertainty remains. Burda et al. (2018); Yang et al. (2024) interpret the model uncertainty of neural networks (i.e., high errors in unseen inputs) as intrinsic rewards to further explore those states. Bellemare et al. (2016) introduces a density model to approximate state visitation counts in continuous state spaces and uses this model to drive the agent to visit less-visited states. However, these methods change the performance objective and thus introduce suboptimality.

Hierarchical RL Sutton et al. (1999b); Machado et al. (2023) with diffusion-maximizing intrinsic rewards—derived from decomposing a graph representation of transition dynamics—has been widely used to solve sparse-reward environments. Graph Laplacians Wang et al. (2021); Wu et al. (2018); Machado et al. (2017a); Gomez et al. (2023) and successor representations Dayan (1993); Machado et al. (2017b); Ramesh et al. (2019) are prominent techniques to construct such representations. However, as shown in Figure 6, hierarchical RL suffers from suboptimality and sample inefficiency due to its hierarchical use of subpolicies.

Another line of work Xu et al. (2018); Schäfer et al. (2021) trains an exploration policy to fill an off-policy replay buffer and train an exploitation policy with that buffer. Zheng et al. (2018) uses meta-gradients to learn intrinsic rewards that guide the policy toward states that directly improve its performance. However, these methods induce suboptimality as they change the performance objective by augmenting extrinsic rewards with intrinsic rewards. Gupta et al. (2018); Norman & Clune (2024) use meta-learning to train context generators that condition policies on latent variables drawn from learned distributions or GPT-translated rollouts, providing appropriate exploration contexts for unseen tasks. Similar to IRPO, MAML Finn et al. (2017) updates a meta-policy using gradients from subpolicies. However, MAML backpropagates subpolicy gradients calculated with respect to task-specific (intrinsic) rewards and uses the meta-policy for few-shot adaptation to unseen tasks, rather than for exploration in sparse-reward environments. Table 1 and Figure 6 show that these distinctions are key to solving sparse-reward environments.

6 Conclusion

We propose IRPO, a meta-gradient algorithm that enables exploration through intrinsic rewards while optimizing for extrinsic rewards. We showed that IRPO outperforms relevant baselines in widely used sparse-reward environments, justified each IRPO component through ablation studies, and proved convergence to an ϵ-approximate first-order stationary point. Despite these promising results, we have not theoretically characterized how intrinsic reward quality affects IRPO's performance, such as through a suboptimality gap conditioned on intrinsic reward quality, and our experiments focus only on goal-reaching environments. Future work should examine the effect of intrinsic reward quality and evaluate IRPO across a broader range of environments.

AI USE STATEMENT

In this work, we used generative AI tools strictly for word choices, finding typos, and finding bugs in our code. We did not use AI to generate text from scratch, references, or experimental data. We have reviewed all AI-assisted outputs and take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we detail our environment and algorithm settings through Appendices B, C, and F. Additionally, we will make our source code publicly available upon the acceptance of this manuscript.

REFERENCES

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

Praveen Agarwal, Mohamed Jleli, and Bessem Samet. Banach contraction principle and applications. In *Fixed Point Theory in Metric Spaces: Recent Advances and Applications*, pp. 1–23. Springer, 2018.

Qinbo Bai, Amrit Singh Bedi, and Vaneet Aggarwal. Achieving zero constraint violation for constrained reinforcement learning via conservative natural policy gradient primal-dual algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6737–6744, 2023.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and trends in Machine Learning*, 8(3-4):231–357, 2015.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Eric Chen, Zhang-Wei Hong, Joni Pajarinen, and Pulkit Agrawal. Redeeming intrinsic rewards via constrained optimization. *Advances in neural information processing systems*, 35:4996–5008, 2022.

Andrew Cohen, Xingye Qiao, Lei Yu, Elliot Way, and Xiangrong Tong. Diverse exploration via conjugate policies for policy gradient methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3404–3411, 2019.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Diego Gomez, Michael Bowling, and Marlos C Machado. Proper laplacian representation learning. *arXiv preprint arXiv:2310.10833*, 2023.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pp. 2295–2304. PMLR, 2017a.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017b.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5125–5133, 2020.

Marlos C Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of machine learning research*, 24(80):1–69, 2023.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013.

Ben Norman and Jeff Clune. First-explore, then exploit: Meta-learning to solve hard exploration-exploitation trade-offs. *Advances in Neural Information Processing Systems*, 37:27490–27528, 2024.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731*, 2019.

Lukas Schäfer, Filippos Christianos, Josiah P Hanna, and Stefano V Albrecht. Decoupled reinforcement learning to stabilise intrinsically-motivated exploration. *arXiv preprint arXiv:2107.08966*, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv e-prints*, pp. arXiv–1707, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5668–5675, 2020.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999a.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999b.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pp. 11003–11012. PMLR, 2021.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Pan Xu, Felicia Gao, and Quanquan Gu. Sample efficient policy gradient methods with recursive variance reduction. *arXiv preprint arXiv:1909.08610*, 2019.

Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore via meta-policy gradient. In *International Conference on Machine Learning*, pp. 5463–5472. PMLR, 2018.

Kai Yang, Jian Tao, Jiafei Lyu, and Xiu Li. Exploration and anti-exploration with distributional random network distillation. In *International Conference on Machine Learning*, pp. 56397–56421. PMLR, 2024.

Rui Yuan, Robert M Gower, and Alessandro Lazaric. A general sample complexity analysis of vanilla policy gradient. In *International Conference on Artificial Intelligence and Statistics*, pp. 3332–3380. PMLR, 2022.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *Advances in neural information processing systems*, 31, 2018.

A PROOFS

We prove that IRPO optimizes J_R over $\mathbb{R}^m$ in Appendix A.1, and prove that IRPO under exact gradients finds an ϵ-approximate first-order stationary point of J_R over $\mathbb{R}^m$ in Appendix A.2.

Throughout the proofs, G_k denotes the sum of N subpolicy gradients, and we define the subpolicy update function f_k for the k-th intrinsic reward function as follows:

$$G_k(\bar{\theta}) \coloneqq \sum_{j=1}^{N} \nabla_{\tilde{\theta}_k^{(j)}} J_{\tilde{R}_k}(\tilde{\theta}_k^{(j)}), \qquad f_k(\bar{\theta}) \coloneqq \bar{\theta} + \tilde{\eta} G_k(\bar{\theta}) = \tilde{\theta}_k^{(N+1)}, \tag{14}$$

where $\tilde{\theta}_k^{(1)} = \bar{\theta}$, $\tilde{\theta}_k^{(j+1)} = \tilde{\theta}_k^{(j)} + \tilde{\eta} \nabla_{\tilde{\theta}_k^{(j)}} J_{\tilde{R}_k}(\tilde{\theta}_k^{(j)})$, and we omit explicit dependence on N, as N is fixed throughout training. The operator f_k represents the update of subpolicy k within one iteration. We also use Assumptions 1 to 3 as stated in Section 3.5.

A.1 IRPO Optimizes J_R over $\mathbb{R}^m$

We define all reachable subpolicies (i.e., the search space of IRPO) by using the image of the function f_k from Equation (14).

Definition 5 (All Reachable Subpolicies)**.** Let $\mathcal{F}_k \coloneqq \bigcup_{\bar{\theta}\in\mathbb{R}^m}\{f_k(\bar{\theta})\}$ be the image of f_k, which is defined in Equation (14). Then, the set of all reachable subpolicies for K intrinsic reward functions is defined as follows:

$$\mathcal{F} \coloneqq \bigcup_{k=1}^{K} \mathcal{F}_k = \bigcup_{k=1}^{K} \bigcup_{\bar{\theta}\in\mathbb{R}^m} \{f_k(\bar{\theta})\}. \tag{15}$$

Lemma 6 (Optimization of J_R over the Reachable Set $\mathcal{F}$)**.** *Let the temperature parameter be annealed as $\tau \to 0_+$. Then, the IRPO update is equivalent to optimizing J_R within $\mathcal{F}$.*

Proof. The objective of IRPO is the log-sum-exp of the performance of subpolicies with respect to extrinsic reward R as follows:

$$J^{\tau}_{\text{IRPO}}(\bar{\theta}) \coloneqq \tau \log \sum_{k=1}^{K} \exp\left(\frac{J_R(f_k(\bar{\theta}))}{\tau}\right) = \tau \log \sum_{k=1}^{K} \exp\left(\frac{J_R(\tilde{\theta}_k)}{\tau}\right). \tag{16}$$

Taking the gradient of J^{τ}_{IRPO} defined in Equation (16) with respect to the meta-policy parameters $\bar{\theta}$ yields:

$$\nabla_{\bar{\theta}} J^{\tau}_{\text{IRPO}}(\bar{\theta}) = \sum_{k=1}^{K} \frac{\exp(J_R(f_k(\bar{\theta}))/\tau)}{\sum_{j=1}^{K}\exp(J_R(f_j(\bar{\theta}))/\tau)} \nabla_{\bar{\theta}} J_R(f_k(\bar{\theta})). \tag{17}$$

Notice that the weighting term is precisely the softmax weight ω^{τ}_k defined in Equation (6).

By Definition 5, any subpolicy $\tilde{\theta}^{(N+1)}_k \in \mathcal{F}$ can be expressed as $f_k(\bar{\theta})$. Furthermore, the log-sum-exp function is a well-known smooth approximation of the maximum function, satisfying the following bounds for any $\bar{\theta} \in \mathbb{R}^m$:

$$\max_{k=1,\dots,K} J_R(f_k(\bar{\theta})) \le J^{\tau}_{\text{IRPO}}(\bar{\theta}) \le \tau \log K + \max_{k=1,\dots,K} J_R(f_k(\bar{\theta})). \tag{18}$$

Let the temperature parameter be annealed as $\tau \to 0_+$. Then, the softmax weights ω^{τ}_k concentrate on the index $q(\bar{\theta}) = \arg\max_k J_R(f_k(\bar{\theta}))$, so that the IRPO gradient reduces to $\nabla_{\bar{\theta}} J_R(f_{q(\bar{\theta})}(\bar{\theta}))$ to update the meta-policy such that N subpolicy updates with respect to $\tilde{R}_{q(\bar{\theta})}$ maximize J_R. Consequently, the IRPO update ascends $J_R(f_{q(\bar{\theta})}(\bar{\theta}))$ with respect to $\bar{\theta}$, and maximizing this objective over the meta-policy parameters poses the following optimization problem:

$$\max_{\bar{\theta}\in\mathbb{R}^m} J_R(f_{q(\bar{\theta})}(\bar{\theta})) = \max_{\tilde{\theta}\in\mathcal{F}} J_R(\tilde{\theta}), \tag{19}$$

where the final equality follows from Definition 5 and the definition of $q(\bar{\theta})$, since $J_R(f_{q(\bar{\theta})}(\bar{\theta})) = \max_{k=1,\dots,K} J_R(f_k(\bar{\theta}))$ and the reachable set $\mathcal{F}$ consists exactly of all policies of the form $f_k(\bar{\theta})$ for $\bar{\theta} \in \mathbb{R}^m$ and $k = 1, 2, \dots, K$. □

We now show that a sufficiently small learning rate $\tilde{\eta}$ for subpolicy updates ensures that IRPO does not artificially restrict the search space $\mathcal{F}$, thereby preserving all stationary points of the performance objective J_R as part of IRPO's search space (i.e., $\mathcal{F} = \mathbb{R}^m$).

Lemma 7 (Bijectivity of the Mapping f_k)**.** *Let $\mathcal{F}$ be defined in Definition 5 and the learning rate used for subpolicy updates satisfy $\tilde{\eta} < L_G^{-1}$. Under Assumption 1a, each $f_k : \mathbb{R}^m \to \mathbb{R}^m$ is bijective, satisfying $\mathcal{F} = \mathbb{R}^m$.*

Proof. We establish bijectivity of $f_k : \mathbb{R}^m \to \mathbb{R}^m$ by showing that for any $\tilde{\theta}_k \in \mathbb{R}^m$, there exists a unique $\bar{\theta} \in \mathbb{R}^m$ such that $f_k(\bar{\theta}) = \tilde{\theta}_k$.

Fix $\tilde{\theta}_k \in \mathbb{R}^m$ and define the function $T_k : \mathbb{R}^m \to \mathbb{R}^m$ by $T_k(\bar{\theta}) = \tilde{\theta}_k - \tilde{\eta} G_k(\bar{\theta})$. Note that $\bar{\theta}$ is a fixed point of T_k if and only if $f_k(\bar{\theta}) = \tilde{\theta}_k$.

We now show that T_k is a contraction mapping, which establishes the existence and uniqueness of its fixed point. For any $\bar{\theta}_1, \bar{\theta}_2 \in \mathbb{R}^m$:

$$\|T_k(\bar{\theta}_1) - T_k(\bar{\theta}_2)\|_2 = \|(\tilde{\theta}_k - \tilde{\eta}G_k(\bar{\theta}_1)) - (\tilde{\theta}_k - \tilde{\eta}G_k(\bar{\theta}_2))\|_2 \tag{20}$$

$$= \tilde{\eta}\|G_k(\bar{\theta}_1) - G_k(\bar{\theta}_2)\|_2 \tag{21}$$

$$\leq \tilde{\eta}L_G\|\bar{\theta}_1 - \bar{\theta}_2\|_2. \tag{22}$$

Given $\tilde{\eta} < L_G^{-1}$, the constant $\tilde{\eta}L_G < 1$, making T_k a strict contraction on the complete metric space $(\mathbb{R}^m, \|\cdot\|_2)$. By the Banach Fixed-Point Theorem Agarwal et al. (2018), T_k has a *unique* fixed point $\bar{\theta}^\star \in \mathbb{R}^m$. This implies $f_k(\bar{\theta}^\star) = \tilde{\theta}_k$ is the unique solution for $\tilde{\theta}_k$, establishing that f_k is a bijection and its image satisfies $\mathcal{F}_k = \mathbb{R}^m$. Since $\mathcal{F}$ is the union of $\mathcal{F}_k$ (see Definition 5), $\mathcal{F} = \mathbb{R}^m$ holds. □

Corollary 8 (IRPO Optimizes J_R over $\mathbb{R}^m$)**.** *Let the temperature parameter be annealed as $\tau \to 0_+$, and let the learning rate used for subpolicy updates satisfy $\tilde{\eta} < L_G^{-1}$. Then, the IRPO update optimizes J_R within $\mathbb{R}^m$.*

Proof. Recall from Lemma 6 that IRPO optimizes J_R over the reachable set $\mathcal{F}$. The result then follows directly from Lemma 7, which guarantees that $\mathcal{F} = \mathbb{R}^m$ under the condition $\tilde{\eta} < L_G^{-1}$. □

A.2 Convergence to an ϵ-Approximate First-Order Stationary Point

We first establish two useful lemmas to prove convergence.

Lemma 9 (Non-Singularity and Uniform Boundedness of Jacobian $\nabla_{\bar{\theta}} f_k(\bar{\theta})$)**.** *Let the learning rate used for subpolicy updates satisfy $\tilde{\eta} < L_G^{-1}$, and let $\mathcal{J}_k(\bar{\theta}) \coloneqq \nabla_{\bar{\theta}} f_k(\bar{\theta})$ denote the Jacobian of the subpolicy update f_k. Under Assumption 1a, the Jacobian $\mathcal{J}_k(\bar{\theta})$ is non-singular and uniformly bounded for all $k = 1, 2, \ldots, K$ as follows:*

$$0 \;<\; 1 - \tilde{\eta}L_G \;\leq\; \sigma_{\min}\big(\mathcal{J}_k(\bar{\theta})\big) \;\leq\; \sigma_{\max}\big(\mathcal{J}_k(\bar{\theta})\big) \;\leq\; 1 + \tilde{\eta}L_G, \quad \forall \bar{\theta} \in \mathbb{R}^m, \tag{23}$$

where $\sigma_{\min}(\cdot)$ and $\sigma_{\max}(\cdot)$ denote the minimum and maximum singular values of a matrix, respectively.

Proof. By definition of f_k in Equation (14), the Jacobian is given by $\mathcal{J}_k(\bar{\theta}) = \nabla_{\bar{\theta}}(\bar{\theta} + \tilde{\eta}G_k(\bar{\theta})) = I + \tilde{\eta}\nabla G_k(\bar{\theta})$. Because G_k is globally L_G-Lipschitz continuous (Assumption 1a), its Jacobian satisfies $\|\nabla G_k(\bar{\theta})\|_2 \leq L_G$. Applying the triangle inequality yields an upper bound on the maximum singular value:

$$\sigma_{\max}(\mathcal{J}_k(\bar{\theta})) = \|\mathcal{J}_k(\bar{\theta})\|_2 \leq \|I\|_2 + \tilde{\eta}\|\nabla G_k(\bar{\theta})\|_2 \leq 1 + \tilde{\eta}L_G. \tag{24}$$

Similarly, by the reverse triangle inequality, we lower-bound the minimum singular value by taking the minimum over all unit vectors v:

$$\sigma_{\min}(\mathcal{J}_k(\bar{\theta})) = \min_{\|v\|_2=1} \big\|\big(I + \tilde{\eta}\nabla G_k(\bar{\theta})\big)\, v\big\|_2 \geq 1 - \max_{\|v\|_2=1} \|\tilde{\eta}\nabla G_k(\bar{\theta})v\|_2 \geq 1 - \tilde{\eta}L_G. \tag{25}$$

Given $\tilde{\eta} < L_G^{-1}$, we have $1 - \tilde{\eta}L_G > 0$, implying $\mathcal{J}_k(\bar{\theta})$ is non-singular and every singular value is uniformly bounded from above. □

The following lemma reformulates the descent lemma (see Lemma 3.4 in Bubeck (2015)) as an ascent lemma for the maximization framework of RL.

Lemma 10 (Ascent Lemma)**.** *Under Assumption 1c, we have the following lower bound on the performance difference for any subpolicy parameters $\theta_1, \theta_2 \in \mathbb{R}^m$:*

$$J_R(\theta_2) - J_R(\theta_1) \geq \nabla J_R(\theta_1)^\top (\theta_2 - \theta_1) - \frac{L_J}{2}\|\theta_2 - \theta_1\|_2^2. \tag{26}$$

Proof. By the fundamental theorem of calculus, the exact difference in the objective function evaluated at any two points can be expressed using a line integral as follows:

$$J_R(\theta_2) - J_R(\theta_1) = \int_0^1 \nabla J_R(\theta_1 + t(\theta_2 - \theta_1))^\top (\theta_2 - \theta_1) dt.$$

Adding and subtracting $\nabla J_R(\theta_1)^\top(\theta_2-\theta_1)$ yields:

$$J_R(\theta_2)-J_R(\theta_1)=\nabla J_R(\theta_1)^\top(\theta_2-\theta_1)+\int_0^1\left(\nabla J_R(\theta_1+t(\theta_2-\theta_1))-\nabla J_R(\theta_1)\right)^\top(\theta_2-\theta_1)dt.$$

Applying the Cauchy-Schwarz inequality to lower-bound the inner product, and leveraging the L_J-smoothness of J_R to bound the gradient difference, we obtain:

$$\begin{aligned}J_R(\theta_2)-J_R(\theta_1)&\geq\nabla J_R(\theta_1)^\top(\theta_2-\theta_1)-\int_0^1\left\|\nabla J_R(\theta_1+t(\theta_2-\theta_1))-\nabla J_R(\theta_1)\right\|_2\|\theta_2-\theta_1\|_2dt\\&\geq\nabla J_R(\theta_1)^\top(\theta_2-\theta_1)-\int_0^1\left(tL_J\|\theta_2-\theta_1\|_2\right)\|\theta_2-\theta_1\|_2dt\\&=\nabla J_R(\theta_1)^\top(\theta_2-\theta_1)-L_J\|\theta_2-\theta_1\|_2^2\int_0^1tdt.\end{aligned}$$

Evaluating the integral $\int_0^1 tdt=\frac{1}{2}$ completes the proof. □

Proposition 4 (Convergence to an ϵ-Approximate First-Order Stationary Point)**.** *Let the temperature parameter be annealed as $\tau\to0_+$ and the learning rate used for subpolicy updates satisfy $\tilde{\eta}<L_G^{-1}$. Under Assumptions 1 to 3, IRPO finds an ϵ-approximate first-order stationary point of J_R in $\mathbb{R}^m$ (i.e., a policy $\theta\in\mathbb{R}^m$ satisfying $\|\nabla J_R(\theta)\|_2\leq\epsilon$) within $\mathcal{O}(1/\epsilon^2)$ updates.*

Proof. Let $q(i)=\arg\max_k J_R(f_k(\bar{\theta}^{(i)}))$ denote the index of the subpolicy selected for evaluation at iteration i (see the selection procedure in Section 3.4) after the temperature parameter is annealed as $\tau\to0_+$, and let $\tilde{\theta}^{(i)}\coloneqq f_{q(i)}(\bar{\theta}^{(i)})$ denote the selected subpolicy parameters (evaluation policy parameters). Here, the superscript (i) indexes the iteration (which also indexes the meta-policy update), whereas the superscript (j) used in Section 3.2 indexes the inner subpolicy updates. After iteration $i+1$, the parameters of subpolicy $q(i)$ can be represented as follows:

$$f_{q(i)}(\bar{\theta}^{(i+1)})=f\left(\bar{\theta}^{(i)}+\sqrt{\frac{2\delta_{\text{KL}}}{\nabla J_{\text{IRPO}}^\top F_{\bar{\theta}^{(i)}}^{-1}\nabla J_{\text{IRPO}}}}F_{\bar{\theta}^{(i)}}^{-1}\nabla J_{\text{IRPO}}\right).\tag{27}$$

We first analyze the change in parameters of the evaluation policy after one iteration, i.e., $f_{q(i)}(\bar{\theta}^{(i+1)})-\tilde{\theta}^{(i)}$.

Let $\mathcal{J}^{(i)}\coloneqq\nabla_{\bar{\theta}^{(i)}}f_{q(i)}(\bar{\theta}^{(i)})$ denote the Jacobian of $f_{q(i)}$ at $\bar{\theta}^{(i)}$ and $h^{(i)}\coloneqq\nabla_{\tilde{\theta}^{(i)}}J_R(\tilde{\theta}^{(i)})$ denote the policy gradient of the selected (evaluation) subpolicy with respect to extrinsic rewards, with which we can rewrite the IRPO gradient (Equation (5)) as $\nabla J_{\text{IRPO}}={\mathcal{J}^{(i)}}^\top h^{(i)}$.

Let the meta-policy update be denoted as $\Delta\bar{\theta}^{(i)}\coloneqq\bar{\theta}^{(i+1)}-\bar{\theta}^{(i)}$. The first-order Taylor expansion of Equation (27) at $\bar{\theta}^{(i)}$ gives

$$f_{q(i)}(\bar{\theta}^{(i+1)})=\tilde{\theta}^{(i)}+\alpha^{(i)}\,M^{(i)}\,h^{(i)}+r^{(i)},\tag{28}$$

$$\alpha^{(i)}\coloneqq\sqrt{\frac{2\delta_{\text{KL}}}{{h^{(i)}}^\top M^{(i)}h^{(i)}}}\in\mathbb{R}_{>0},\quad M^{(i)}\coloneqq\mathcal{J}^{(i)}F_{\bar{\theta}^{(i)}}^{-1}{\mathcal{J}^{(i)}}^\top,\tag{29}$$

$$r^{(i)}\coloneqq f_{q(i)}(\bar{\theta}^{(i+1)})-f_{q(i)}(\bar{\theta}^{(i)})-\mathcal{J}^{(i)}\Delta\bar{\theta}^{(i)}=\int_0^1\left(\nabla f(\bar{\theta}^{(i)}+t\Delta\bar{\theta}^{(i)})-\nabla f(\bar{\theta}^{(i)})\right)\Delta\bar{\theta}^{(i)}\,dt,\tag{30}$$

where $\mathcal{J}^{(i)}$ in $M^{(i)}$ arises from the Taylor expansion of $f_{q(i)}$, and the remainder $r^{(i)}$ is defined by the integral form of Taylor's theorem for vector-valued functions. Note that $f_{q(i)}(\bar{\theta}^{(i+1)})$ need not equal the policy $\tilde{\theta}^{(i+1)}$ selected at the next iteration, since the selected index may change; we account for this throughout our analysis. We now bound $M^{(i)}$ and $r^{(i)}$.

Spectral bounds on $M^{(i)}$. By Lemma 9, $\mathcal{J}^{(i)} = I + \tilde{\eta}\nabla G(\bar{\theta}^{(i)})$ is uniformly non-singular. Because a matrix and its transpose share singular values, we have for any $x \in \mathbb{R}^m$:

$$(1 - \tilde{\eta}L_G)\|x\|_2 \;\le\; \left\|{\mathcal{J}^{(i)}}^\top x\right\|_2 \;\le\; (1 + \tilde{\eta}L_G)\|x\|_2. \tag{31}$$

Let $y = {\mathcal{J}^{(i)}}^\top x$. Using the uniform bounds $\underline{\mu} I \preceq F_{\bar{\theta}^{(i)}}^{-1} \preceq \overline{\mu} I$ (Assumption 2), the quadratic form $x^\top M^{(i)} x = y^\top F_{\bar{\theta}^{(i)}}^{-1} y$ is bounded via Equation (31) as follows:

$$\underline{\mu}(1 - \tilde{\eta}L_G)^2\|x\|_2^2 \;\le\; x^\top M^{(i)} x \;\le\; \overline{\mu}(1 + \tilde{\eta}L_G)^2\|x\|_2^2. \tag{32}$$

Because this holds for all x, $M^{(i)}$ is strictly positive definite and uniformly bounded from above:

$$0 \prec (1 - \tilde{\eta}L_G)^2\underline{\mu} I \preceq M^{(i)} \preceq (1 + \tilde{\eta}L_G)^2\overline{\mu} I. \tag{33}$$

Bound on the Taylor remainder $r^{(i)}$. Recall that $f_{q(i)}(\bar{\theta}) = \bar{\theta} + \tilde{\eta}G_{q(i)}(\bar{\theta})$, which implies its Jacobian is $\nabla f_{q(i)}(\bar{\theta}) = I + \tilde{\eta}\nabla G_{q(i)}(\bar{\theta})$. By Assumption 1b, the Jacobian $\nabla G(\bar{\theta})$ is L'_G-Lipschitz continuous, which guarantees that $\nabla f(\bar{\theta})$ is $(\tilde{\eta}L'_G)$-Lipschitz continuous. Taking the ℓ_2-norm of the remainder $r^{(i)}$ and applying the Lipschitz continuity of $\nabla f(\bar{\theta})$, we bound the remainder as follows:

$$\|r^{(i)}\|_2 \le \int_0^1 \left\|\nabla f_{q(i)}(\bar{\theta}^{(i)} + t\Delta\bar{\theta}^{(i)}) - \nabla f_{q(i)}(\bar{\theta}^{(i)})\right\|_2 \left\|\Delta\bar{\theta}^{(i)}\right\|_2 dt \tag{34}$$

$$\le \int_0^1 \left(\tilde{\eta}L'_G t\left\|\Delta\bar{\theta}^{(i)}\right\|_2\right)\left\|\Delta\bar{\theta}^{(i)}\right\|_2 dt \tag{35}$$

$$= \tilde{\eta}L'_G\|\Delta\bar{\theta}^{(i)}\|_2^2 \int_0^1 t\, dt = \frac{\tilde{\eta}L'_G}{2}\|\Delta\bar{\theta}^{(i)}\|_2^2. \tag{36}$$

To express this remainder bound in terms of the trust-region threshold δ_{KL}, we bound the squared ℓ_2-norm of the meta-policy update, $\|\Delta\bar{\theta}^{(i)}\|_2^2$. Using the analytical trust-region update $\Delta\bar{\theta}^{(i)} = \alpha^{(i)} F_{\bar{\theta}^{(i)}}^{-1}\nabla J_{\text{IRPO}}$, its squared norm is:

$$\|\Delta\bar{\theta}^{(i)}\|_2^2 \;=\; (\alpha^{(i)})^2\,\nabla J_{\text{IRPO}}^\top F_{\bar{\theta}^{(i)}}^{-2}\nabla J_{\text{IRPO}} \;=\; \frac{2\delta_{\text{KL}}\cdot\nabla J_{\text{IRPO}}^\top F_{\bar{\theta}^{(i)}}^{-2}\nabla J_{\text{IRPO}}}{\nabla J_{\text{IRPO}}^\top F_{\bar{\theta}^{(i)}}^{-1}\nabla J_{\text{IRPO}}}. \tag{37}$$

From Assumption 2, the maximum eigenvalue of $F_{\bar{\theta}^{(i)}}^{-1}$ is bounded by $\overline{\mu}$. Because $F_{\bar{\theta}^{(i)}}^{-1}$ is symmetric positive definite, this implies the matrix inequality $F_{\bar{\theta}^{(i)}}^{-2} \preceq \overline{\mu} F_{\bar{\theta}^{(i)}}^{-1}$. Consequently, we can upper-bound Equation (37) as follows:

$$\|\Delta\bar{\theta}^{(i)}\|_2^2 \;\le\; \frac{2\delta_{\text{KL}}\overline{\mu}\cdot\nabla J_{\text{IRPO}}^\top F_{\bar{\theta}^{(i)}}^{-1}\nabla J_{\text{IRPO}}}{\nabla J_{\text{IRPO}}^\top F_{\bar{\theta}^{(i)}}^{-1}\nabla J_{\text{IRPO}}} \;=\; 2\delta_{\text{KL}}\overline{\mu}. \tag{38}$$

Substituting this into our Taylor remainder inequality yields $\|r^{(i)}\|_2 \le \tilde{\eta}L'_G\overline{\mu}\delta_{\text{KL}}$.

Analysis for subpolicy update. Let us define the following to simplify the proof:

$$\underline{m} \coloneqq (1 - \tilde{\eta}L_G)^2\underline{\mu}, \quad \overline{m} \coloneqq (1 + \tilde{\eta}L_G)^2\overline{\mu}, \quad \rho \coloneqq \tilde{\eta}L'_G\overline{\mu}\delta_{\text{KL}}. \tag{39}$$

Because IRPO selects the maximizing index at every iteration, the performance of the selected subpolicy at the $(i+1)$-th iteration satisfies $J_R(\tilde{\theta}^{(i+1)}) = J_R(f_{q(i+1)}(\bar{\theta}^{(i+1)})) \ge J_R(f_{q(i)}(\bar{\theta}^{(i+1)}))$. Given that ∇J_R is L_J-Lipschitz continuous (Assumption 1c), we apply Lemma 10 with $\theta_1 = \tilde{\theta}^{(i)}$ and $\theta_2 = \tilde{\theta}^{(i+1)}$ to obtain the following inequality:

$$J_R(\tilde{\theta}^{(i+1)}) - J_R(\tilde{\theta}^{(i)}) \tag{40}$$

$$= J_R(f_{q(i+1)}(\bar{\theta}^{(i+1)})) - J_R(f_{q(i)}(\bar{\theta}^{(i)})) \tag{41}$$

$$\ge J_R(f_{q(i)}(\bar{\theta}^{(i+1)})) - J_R(f_{q(i)}(\bar{\theta}^{(i)})) \tag{42}$$

$$\ge {h^{(i)}}^\top\left(f_{q(i)}(\bar{\theta}^{(i+1)}) - \tilde{\theta}^{(i)}\right) - \frac{L_J}{2}\left\|f_{q(i)}(\bar{\theta}^{(i+1)}) - \tilde{\theta}^{(i)}\right\|_2^2 \tag{43}$$

$$= \alpha^{(i)}{h^{(i)}}^\top M^{(i)}h^{(i)} + {h^{(i)}}^\top r^{(i)} - \frac{L_J}{2}\left\|\alpha^{(i)}M^{(i)}h^{(i)} + r^{(i)}\right\|_2^2 \tag{44}$$

$$\ge \sqrt{2\delta_{\text{KL}}{h^{(i)}}^\top M^{(i)}h^{(i)}} - \rho\|h^{(i)}\|_2 - \frac{L_J}{2}\left(\|\alpha^{(i)}M^{(i)}h^{(i)}\|_2^2 + 2\rho\|\alpha^{(i)}M^{(i)}h^{(i)}\|_2 + \rho^2\right). \tag{45}$$

To bound the squared norm $\|\alpha^{(i)} M^{(i)} h^{(i)}\|_2^2$, we use the following spectral bound $\lambda_{\max}(M^{(i)}) \leq \overline{m}$ from Equation (33):

$$\|\alpha^{(i)} M^{(i)} h^{(i)}\|_2^2 = \frac{2\delta_{\mathrm{KL}}}{{h^{(i)}}^\top M^{(i)} h^{(i)}} \|M^{(i)} h^{(i)}\|_2^2 \;\leq\; 2\delta_{\mathrm{KL}} \lambda_{\max}(M^{(i)}) \;\leq\; 2\delta_{\mathrm{KL}} \overline{m}, \tag{46}$$

where the first inequality uses $\|M^{(i)} h^{(i)}\|_2^2 \leq \lambda_{\max}(M^{(i)})\, {h^{(i)}}^\top M^{(i)} h^{(i)}$, which holds because $M^{(i)}$ is positive definite by Equation (33). Given the identity $x^\top M x \leq \lambda_{\max}(M)\|x\|_2^2$ for positive definite M, applying Equation (46) and the lower spectral bound on $M^{(i)}$, the inequality simplifies to:

$$J_R(\tilde{\theta}^{(i+1)}) - J_R(\tilde{\theta}^{(i)}) \geq \left(\sqrt{2\delta_{\mathrm{KL}}\underline{m}} - \rho\right) \|h^{(i)}\|_2 - \frac{L_J}{2}\left(2\delta_{\mathrm{KL}}\overline{m} + 2\sqrt{2\delta_{\mathrm{KL}}\overline{m}}\,\rho + \rho^2\right) \tag{47}$$

$$= \left(\sqrt{2\delta_{\mathrm{KL}}\underline{m}} - \rho\right) \|h^{(i)}\|_2 - \mathcal{O}(\delta_{\mathrm{KL}}) - \mathcal{O}(\delta_{\mathrm{KL}}^{3/2}) - \mathcal{O}(\delta_{\mathrm{KL}}^2) \tag{48}$$

$$= \left(\sqrt{2\delta_{\mathrm{KL}}\underline{m}} - \rho\right) \|h^{(i)}\|_2 - \mathcal{O}(\delta_{\mathrm{KL}}). \tag{49}$$

Let $\Omega(\sqrt{\delta_{\mathrm{KL}}}) \coloneqq \sqrt{2\delta_{\mathrm{KL}}\underline{m}} - \rho$. Rearranging Equation (49) to isolate $\|h^{(i)}\|_2$ yields:

$$\Omega\left(\sqrt{\delta_{\mathrm{KL}}}\right) \|h^{(i)}\|_2 \leq\; J_R(\tilde{\theta}^{(i+1)}) - J_R(\tilde{\theta}^{(i)}) + \mathcal{O}(\delta_{\mathrm{KL}}). \tag{50}$$

By Assumption 3, δ_{KL} is sufficiently small such that $\Omega(\sqrt{\delta_{\mathrm{KL}}}) > 0$. Furthermore, since the discount factor $\gamma < 1$ and the reward $R \in [0, R_{\max}]$, the objective is bounded from above by $J_{\max} = R_{\max}/(1-\gamma)$. Summing the inequality in Equation (50) from $i = 1$ to T, the objective differences telescope to yield:

$$\underbrace{\Omega(\sqrt{\delta_{\mathrm{KL}}})}_{>0} \sum_{i=1}^{T} \|h^{(i)}\|_2 \;\leq\; \underbrace{\left(J_R(\tilde{\theta}^{(T+1)}) - J_R(\tilde{\theta}^{(1)})\right)}_{\leq D \coloneqq J_{\max} - J_R(\theta^{(1)})} + T \cdot \mathcal{O}(\delta_{\mathrm{KL}}). \tag{51}$$

Dividing both sides by T and the coefficient $\Omega(\sqrt{\delta_{\mathrm{KL}}})$ isolates the average gradient norm $\frac{1}{T}\sum_{i=1}^{T} \|h^{(i)}\|_2$.

Recall that $h^{(i)} = \nabla_{\tilde{\theta}^{(i)}} J_R(\tilde{\theta}^{(i)})$ is the policy gradient of the selected (evaluation) subpolicy at iteration i with respect to extrinsic rewards. We now show that $h^{(i)}$ vanishes over iterations 1 to T. Because the minimum of a finite sequence is at most its average, there exists an update $i^\star = \arg\min_{1 \leq i \leq T} \|h^{(i)}\|_2$ bounded by:

$$\|h^{(i^\star)}\|_2 \;\leq\; \frac{1}{T}\sum_{i=1}^{T} \|h^{(i)}\|_2 \;\leq\; \mathcal{O}\left(\frac{1}{T\sqrt{\delta_{\mathrm{KL}}}}\right) + \mathcal{O}(\sqrt{\delta_{\mathrm{KL}}}). \tag{52}$$

Finally, we substitute the scaled KL threshold $\delta_{\mathrm{KL}} = \mathcal{O}(1/T)$ from Assumption 3, yielding:

$$\|h^{(i^\star)}\|_2 \;\leq\; \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) \;=\; \mathcal{O}\left(\frac{1}{\sqrt{T}}\right). \tag{53}$$

Analysis for meta-policy update. Equation (53) bounds the gradient of the selected (evaluation) subpolicy at iteration i (i.e., $\tilde{\theta}^{(i)}$) with respect to extrinsic rewards, and Equation (31) transfers this bound to the IRPO gradient that drives the meta-policy. Setting $x = h^{(i)}$ in Equation (31) and recalling the factorization $\nabla J_{\mathrm{IRPO}}(\bar{\theta}^{(i)}, \{f_k(\bar{\theta}^{(i)})\}_{k=1}^K) = {\mathcal{J}^{(i)}}^\top h^{(i)}$ gives the following for every iteration i:

$$(1 - \tilde{\eta} L_G)\, \|h^{(i)}\|_2 \;\leq\; \|\nabla J_{\mathrm{IRPO}}(\bar{\theta}^{(i)}, \{f_k(\bar{\theta}^{(i)})\}_{k=1}^K)\|_2 \;\leq\; (1 + \tilde{\eta} L_G)\, \|h^{(i)}\|_2, \tag{54}$$

with $1 - \tilde{\eta} L_G > 0$ because $\tilde{\eta} < L_G^{-1}$. Hence, evaluating Equation (54) at update $i^\star$ from Equation (53) directly yields:

$$\|\nabla J_{\mathrm{IRPO}}(\bar{\theta}^{(i^\star)}, \{f_k(\bar{\theta}^{(i^\star)})\}_{k=1}^K)\|_2 \;\leq\; (1 + \tilde{\eta} L_G) \|h^{(i^\star)}\|_2 \;=\; \mathcal{O}\left(\frac{1}{\sqrt{T}}\right). \tag{55}$$

Environment	γ	Horizon	State Dim.	Action Dim.	Number of Intrinsic rewards (K)
Maze-v1	0.99	300	4	4	6
Maze-v2	0.99	300	4	4	6
FourRooms	0.99	100	4	4	4
PointMaze-v1	0.999	500	8	2	4
PointMaze-v2	0.999	500	8	2	4
FetchReach	0.99	50	16	4	4
AntMaze-v1	0.9999	1000	27	8	3
AntMaze-v2	0.9999	1000	27	8	3
AntMaze-v3	0.9999	1500	27	8	3

Table 2: Environment parameters used in our experiments. K also denotes the number of subpolicies of IRPO and HRL.

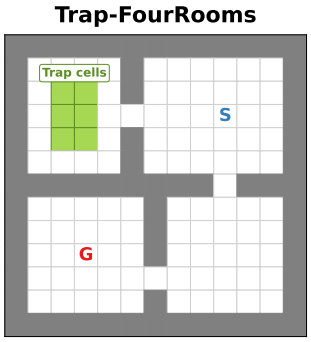

Figure 8: We depict the *Trap-FourRooms* environment, in which we add six absorbing trap states, each with a reward of $+0.01$, to the top-left room of *FourRooms* and block the path between the top-left and bottom-left rooms.

Conclusion. We derive a convergence result from Equations (53) and (55) by requiring their right-hand sides to be at most ϵ, which yields $T = \mathcal{O}(1/\epsilon^2)$. Combined with Corollary 8, which establishes that the IRPO update optimizes J_R over $\mathbb{R}^m$, this result shows that IRPO finds an ϵ-approximate first-order stationary point of J_R over $\mathbb{R}^m$ within $\mathcal{O}(1/\epsilon^2)$ updates. □

B Environment Details

In each environment in Figure 5, an agent is initialized at a fixed starting location (denoted as S), and the goal (denoted as G) is placed at a distant location from the initial state. For discrete environments, the state consists of the Cartesian coordinates of the agent and the goal, and the action space is $\mathcal{A} = \{\text{left}, \text{up}, \text{right}, \text{down}\}$. For continuous environments, the state consists of the Cartesian coordinates of the agent and the goal, as well as the agent's kinematic information, such as velocity. We adopt the default action set provided by `Gymnasium-Robotics` de Lazcano et al. (2024) for each environment with continuous state and action spaces. For all environments, the agent receives a $+1$ reward for reaching the goal. In *AntMaze* environments, we also give a -1 penalty and transition the agent to an absorbing state with reward 0 if the agent jumps aggressively or turns over, ensuring it first learns to walk effectively. Additionally, we increased the goal radius in *AntMaze* environments from 0.35 to 0.5, as the default radius required unnatural jumps to reach the goal. We summarize important environment parameters, including discount factor, evaluation horizon, and state and action dimensions in Table 2. We visualize the *Trap-FourRooms* in Figure 8.

C Algorithm Details

Each baseline algorithm is selected to compare against a specific aspect of IRPO. We use PPO and PSNE to evaluate whether exploration through action-space or parameter-space stochasticity alone

General Hyperparameters				
Parameter	**Value**	**Parameter**	**IRPO**	**Baseline**
Actor hidden layers	$(128, 128)$	Target KL divergence	1×10^{-3}	1×10^{-2}
Critic hidden layers	$(128, 128)$	Actor learning rate	1×10^{-2}	3×10^{-4}
Activation	Tanh	Critic learning rate	1×10^{-3}	3×10^{-4}
PPO clip ratio	0.2	Discount factor	See in Table 2	

Table 3: Algorithm hyperparameters used for our experiments. General parameters apply to all algorithms. We show algorithm-specific hyperparameters for IRPO and baseline algorithms.

Parameter	**Search Range**
Actor layers	$\{(64, 64), (128, 128), (256, 256)\}$
Critic layers	$\{(64, 64), (128, 128), (256, 256)\}$
Activation	$\{\text{ReLU}, \text{Tanh}\}$
PPO clip	$\{0.1, 0.2, 0.3\}$
Target KL	$[1 \times 10^{-4}, 1 \times 10^{-1}]$
Actor lr	$[1 \times 10^{-5}, 1 \times 10^{-2}]$
Critic lr	$[1 \times 10^{-5}, 1 \times 10^{-2}]$

Table 4: Hyperparameter search space.

is sufficient to solve sparse-reward environments. We use PPO-DRND and PPO-ALLO to demonstrate the suboptimality introduced by reward augmentation, which IRPO avoids. We use HRL to demonstrate the architectural superiority of IRPO. We use PPO-LIRPG as a meta-gradient baseline. We use MAML to demonstrate that IRPO's structural differences from MAML are necessary for addressing sparse-reward exploration.

C.1 Hyperparameters

We present the general hyperparameters used in our experiments in Table 3, which were tuned through Bayesian hyperparameter sweeps over the search space in Table 4. We used WandB hyperparameter sweep[1] for this hyperparameter search. The sweeps were conducted in *Maze-v1*, *PointMaze-v1*, and *AntMaze-v1* to identify broadly applicable hyperparameters for the remaining environments. We used PPO to represent the baselines, as HRL, PPO-LIRPG, PPO-ALLO, and PPO-DRND all use PPO for their own policy optimization. For algorithm-specific hyperparameters of IRPO and HRL governing the number of intrinsic rewards, and of PPO-LIRPG and PPO-DRND governing intrinsic-reward learning, we conducted Bayesian hyperparameter sweeps for each environment category, using *Maze-v1* for *Mazes*, *FourRooms* for *FourRooms*, *PointMaze-v1* for *PointMazes*, *FetchReach* for *FetchReach*, and *AntMaze-v1* for *AntMazes*, as environments within each category are similar.

We provide further algorithm-specific implementation details in the following subsections.

C.2 IRPO implementation

When we optimize a meta-policy using the IRPO gradient defined in Equation (5), we found that annealing the temperature τ from 1 to 0 over the first 10% of the total training time steps was shown to be best for the environments in Figure 5 (50% was shown to be best for *Trap-FourRooms*).

Additionally, directly computing the policy gradient in Equation (9) is computationally expensive, as it requires computing the full Jacobian. To address this, we leverage the vector-Jacobian product of automatic differentiation libraries such as `PyTorch` by leveraging the computational graph history across subpolicy updates. In turn, this reduces the computational complexity from $\mathcal{O}(m^2)$ to $\mathcal{O}(m)$ per backward pass, where m is the number of policy parameters.

[1] `https://docs.wandb.ai/models/sweeps`

From Table 3, we adopt a relatively high learning rate for subpolicy updates because the subpolicy update performs a single-batch gradient update, unlike other algorithms that rely on multiple mini-batch updates. However, we use a relatively small target KL divergence of $1\mathrm{e}-3$ for trust-region updates of the IRPO gradient, as it showed more stable convergence than a higher KL divergence threshold. We hypothesize that the chosen small KL-divergence threshold satisfies Assumption 3, which is the key assumption to guarantee convergence.

To reduce the high VRAM usage of IRPO due to backpropagation through vector–Jacobian products, which requires saving computational graphs in VRAM through all subpolicy updates, we estimate the IRPO gradient over minibatches of subpolicies as follows:

$$\nabla J^{\tau}_{\text{IRPO}} = \sum_{d=1}^{D} \sum_{k \in \mathcal{D}_d} \omega^{\tau}_k \nabla_{\tilde{\theta}} J_R(\tilde{\theta}_k), \tag{56}$$

where $\{\mathcal{D}_d\}_{d=1}^{D}$ partitions the K subpolicies into minibatches. While this may increase wall-clock time by performing D backpropagations rather than one, it reduces peak VRAM usage and remains more efficient, as the computational complexity caused by the minibatch approach scales linearly with respect to D while computing the full Jacobian scales quadratically with respect to m.

For IRPO-LIRPG and IRPO-DRND, whose intrinsic rewards need continuous updates throughout the training, we used the same number of subpolicies and subpolicy updates as IRPO-ALLO by assigning each subpolicy an independent intrinsic reward module and updating it with samples from the final subpolicy ($\pi_{\tilde{\theta}_k^{(N+1)}}$).

C.3 PPO-LIRPG IMPLEMENTATION

We directly adopt the LIRPG codebase[2] and optimize a policy using PPO with rewards augmented by LIRPG intrinsic rewards.

C.4 MAML IMPLEMENTATION

We use a set of diffusion-maximizing intrinsic rewards (see Appendix F) as task-specific rewards to perform task-specific subpolicy updates and backpropagate the policy gradients evaluated on those rewards to update the meta-policy. Then, we perform N few-shot updates with respect to extrinsic rewards—N is the number of subpolicy updates used for IRPO—to derive an evaluation policy.

C.5 HIERARCHICAL RL IMPLEMENTATION

Our HRL baseline Sutton et al. (1999b) pretrains subpolicies using intrinsic rewards with PPO, resulting the following set of subpolicies:

$$\Pi = \{\pi_{\tilde{\theta}_k}\}_{k=1}^{K} \cup \{\pi_{\text{rw}}\}, \tag{57}$$

where π_{rw} denotes an additional random walk policy for a nearby fine-grained movement Machado et al. (2017b; 2023). We then optimize a policy π_θ for extrinsic rewards with action space defined as $\mathcal{A} = \Pi$. This makes the policy π_θ use subpolicies for exploration and exploitation. Following a common HRL implementation Machado et al. (2023), any subpolicy can be initiated over the full state space $\mathcal{S}$ and terminates either upon reaching its local optimum or after a fixed number of time steps, which was set to 10 in our experiments.

C.6 PPO IMPLEMENTATION

We follow the standard PPO implementation as described in Huang et al. (2022).

C.7 PPO-ALLO IMPLEMENTATION

For PPO-ALLO, we augment the extrinsic reward with a single intrinsic reward drawn from the same ALLO Gomez et al. (2023) set used by IRPO and HRL as follows:

$$\hat{R}(s,a) = R(s,a) + \alpha\, \tilde{R}_{k^\star}(s,a), \tag{58}$$

[2] https://github.com/Hwhitetooth/lirpg

where we discuss how we chose $k^\star$ and c subsequently.

Choosing $k^\star$. We choose the index $k^\star$ such that the intrinsic reward difference between the initial state s_0 and the goal state s_g is maximized as follows:

$$k^\star = \underset{k=1,2,\ldots,K}{\arg\max} \left(\tilde{R}_k(s_0) - \tilde{R}_k(s_g) \right), \tag{59}$$

where we justify this design as ALLO provides geometry-aware intrinsic rewards and, thus, provides smooth intrinsic rewards toward the goal (see the selected intrinsic reward in Figure 11). Also, this allows for a systematic selection process rather than relying on expert knowledge.

Choosing α. Let V and $\tilde{V}_{k^\star}$ denote the value functions with respect to the extrinsic reward and the chosen intrinsic reward, respectively. The scalar c is chosen such that $\tilde{V}_{k^\star}$ does not outweigh V (i.e., $V(s) > \alpha \tilde{V}_{k^\star}(s)$ for all $s \in \mathcal{S}$) for a policy that successfully reaches the goal, thereby preventing the augmented intrinsic rewards from fooling the policy into ignoring the goal.

Following Machado et al. (2023), we define the ALLO intrinsic reward for index $k^\star$ as follows (see Equation (69) for the full definition):

$$\tilde{R}_{k^\star}(s, s') \coloneqq \psi_{k^\star}(s') - \psi_{k^\star}(s), \tag{60}$$

where $\psi_{k^\star}$ is the ALLO intrinsic reward network parameterized by ψ.

Let $\tilde{\mathtt{R}}_{k^\star} = \max_{s \in \mathcal{S}} |\psi_{k^\star}(s)|$. Because $\tilde{R}_{k^\star}$ takes a potential-based form, the value with respect to $\tilde{R}_{k^\star}$ over any trajectory is strictly bounded as follows:

$$\begin{aligned}
\left| \tilde{V}_{k^\star}(s) \right| &= \left| \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \big(\psi_{k^\star}(s_{t+1}) - \psi_{k^\star}(s_t) \big) \right] \right| && (61)\\
&= \left| \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} \psi_{k^\star}(s_t) \right] - \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t \psi_{k^\star}(s_t) \right] - \psi_{k^\star}(s_0) \right| && (62)\\
&= \left| \mathbb{E}_\pi \left[-\psi_{k^\star}(s_0) + (1-\gamma) \sum_{t=1}^{\infty} \gamma^{t-1} \psi_{k^\star}(s_t) \right] \right| && (63)\\
&\leq \mathbb{E}_\pi \left[|-\psi_{k^\star}(s_0)| + (1-\gamma) \sum_{t=1}^{\infty} \gamma^{t-1} |\psi_{k^\star}(s_t)| \right] && (64)\\
&\leq \mathbb{E}_\pi \left[\tilde{\mathtt{R}}_{k^\star} + (1-\gamma) \sum_{t=1}^{\infty} \gamma^{t-1} \tilde{\mathtt{R}}_{k^\star} \right] && (65)\\
&= \tilde{\mathtt{R}}_{k^\star} + (1-\gamma) \tilde{\mathtt{R}}_{k^\star} \left(\frac{1}{1-\gamma} \right) = 2\tilde{\mathtt{R}}_{k^\star}, \quad \forall s \in \mathcal{S}. && (66)
\end{aligned}$$

Meanwhile, the worst-case value of reaching the goal (+1 reward) within H timesteps is γ^H. To guarantee $V(s) > \alpha \tilde{V}_{k^\star}(s)$ for all $s \in \mathcal{S}$, we set α as follows:

$$\alpha = \frac{\gamma^H}{(2+\nu)\tilde{\mathtt{R}}_{k^\star}}, \tag{67}$$

where $\nu > 0$, and, for each random seed, we empirically estimate $\tilde{\mathtt{R}}_{k^\star}$.

C.8 PPO-DRND IMPLEMENTATION

We directly adopt the DRND codebase[3] and optimize a policy using PPO with rewards augmented by DRND intrinsic rewards.

[3] https://github.com/yk7333/DRND

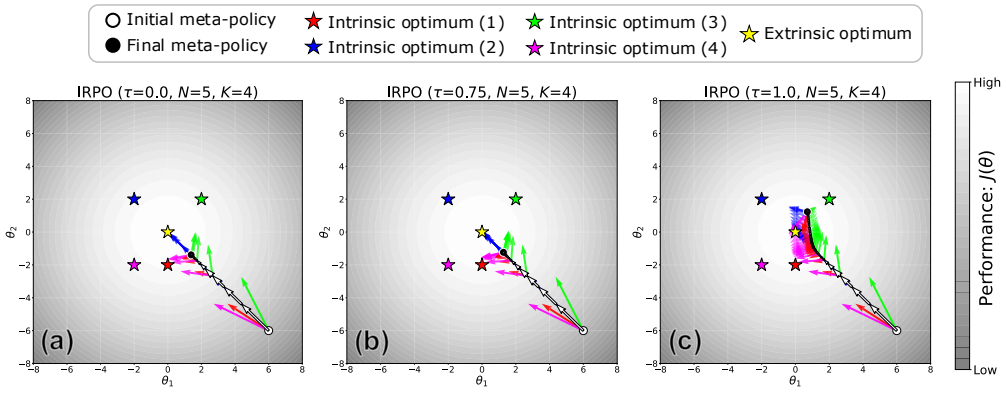

Figure 9: We visualize the update mechanism of IRPO for K intrinsic rewards and N subpolicy updates, with a varying temperature parameter τ.

C.9 PSNE IMPLEMENTATION

We sample K independent Gaussian noises, $\epsilon_k \sim \mathcal{N}(0, I)$, and inject them into the policy parameters θ using reparameterization trick as $\tilde{\theta}_k = \theta + \Sigma^{1/2}\epsilon_k$, where Σ is the learnable covariance. The noise scale is adaptively adjusted to satisfy the given KL divergence between the policy π_θ and noise-injected policies $\pi_{\tilde{\theta}_k}$. Rollouts are collected using $\{\pi_{\tilde{\theta}_k}\}_{k=1}^{K}$, and both θ and Σ are updated to optimize their average performance.

D EMPIRICAL ANALYSIS WITH MULTIPLE INTRINSIC REWARD FUNCTIONS

We introduce three additional intrinsic performance objectives (see others in Section 3.5) as follows:

$$\tilde{J}_2(\theta) = -\left\|\theta - \begin{bmatrix}-2\\2\end{bmatrix}\right\|_2^2, \quad \tilde{J}_3(\theta) = -\left\|\theta - \begin{bmatrix}2\\2\end{bmatrix}\right\|_2^2, \quad \tilde{J}_4(\theta) = -\left\|\theta - \begin{bmatrix}-2\\-2\end{bmatrix}\right\|_2^2. \tag{68}$$

We show IRPO's update mechanism with multiple intrinsic rewards and varying temperature parameter $\tau \in \{0.0, 0.75, 1.0\}$ in Figure 9.

Figure 9ab show that the meta-policy converges to a policy from which one subpolicy (shown in blue) maximizes J after N subpolicy updates, consistent with Section 3.5. In contrast, Figure 9c shows that a high temperature ($\tau = 1.0$) steers the meta-policy toward a policy whose subpolicies, after N updates, achieve higher average extrinsic performance. This matches IRPO's initial exploration phase discussed in Section 4.2, which explores the state space by prioritizing gradients that yield higher extrinsic performance while neglecting regions with no improvement in extrinsic rewards.

E COMPUTATIONAL COMPLEXITY

One potential concern may be the computational complexity of IRPO. Specifically, IRPO requires updating $N \times K$ actors (subpolicy updates), $2(N \times K)$ critics, and computing the Jacobian, along with a trust-region update (meta-policy update) per iteration. For this reason, we analyze the computational complexity of IRPO by evaluating its wall-clock time and the contribution of each IRPO component to GPU memory usage and wall-clock time. We conduct this analysis using a Ryzen 9 7900X CPU, 32GB DDR5 RAM, and an RTX 4070 SUPER GPU.

Table 5 shows that the mean wall-clock time over 5 random seeds is similar for IRPO and PPO in the *PointMaze-v1* and *AntMaze-v1* environments, which is achieved by parallelizing the subpolicy updates across K subpolicies. Additionally, the runtime breakdown of IRPO shows that most of IRPO's computational cost comes from subpolicy updates and sample collection. Noticeably, the complexity of the trust-region update and backpropagation is negligible compared to the time required for subpolicy updates.

Total Time (hours)

Env.	IRPO	PPO
AntMaze-v1	28.06 ± 0.32	28.62 ± 0.66
PointMaze-v1	3.15 ± 0.02	2.95 ± 0.20

IRPO Runtime Breakdown (%)

Component	AntMaze-v1	PointMaze-v1
Subpolicy Update	58.27 ± 0.11	71.66 ± 0.21
Sample Time	41.05 ± 0.11	26.57 ± 0.21
Meta-Policy Update	0.20 ± 0.00	0.47 ± 0.01
Backpropagation	0.03 ± 0.00	0.06 ± 0.00
Remainder	0.44 ± 0.15	1.25 ± 0.30

Table 5: Mean wall-clock time and runtime breakdown over 5 random seeds. The remainder in the runtime breakdown table reflects spawning parallel environments, transferring CPU data to the GPU for neural network updates, and etc.

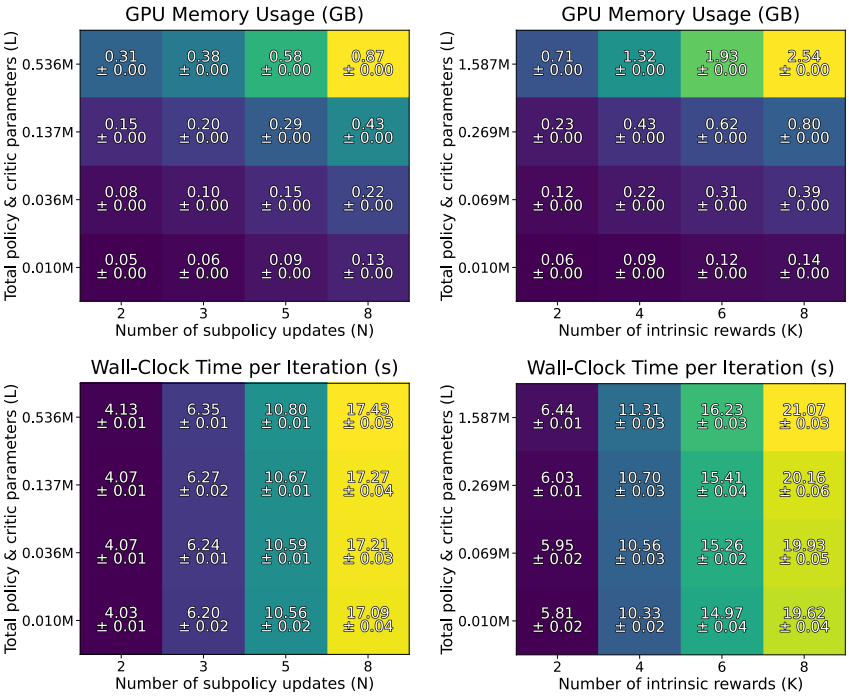

Figure 10: Heatmaps showing the interdependencies of GPU memory usage and wall-clock time with respect to the number of policy and critic parameters per subpolicy (L), the number of intrinsic rewards (K), and the number of subpolicy updates (N).

We also provide heatmaps showing how the number of policy and critic parameters per subpolicy (L), the number of intrinsic rewards (K), and the number of subpolicy updates (N) are correlated with GPU memory usage and per-iteration wall-clock time in Figure 10. The mean and 95% confidence intervals are measured over 10 iterations of IRPO in the *PointMaze-v1* environment, where one iteration includes the meta-policy update, subpolicy updates, and evaluation policy selection. Notably, Figure 10 shows that GPU memory usage is heavily dependent on L due to the Jacobian computations required for backpropagation, while per-iteration wall-clock time has minimal dependence on L, as backpropagation is not a major contributor to computation wall-clock time, consistent with Table 5. Another observation is that the per-iteration wall-clock time is shown to linearly increase with K and N.

F INTRINSIC REWARDS FROM ALLO

We use the ALLO codebase[4] for our algorithm with its default hyperparameters Gomez et al. (2023). These include a discount sampling parameter of 0.9 for discrete environments and 0.99 for continuous ones. Additionally, we employ a neural network with hidden layers of $(256, 256, 256, 256)$, a learning rate of $3\text{e} - 4$, and ReLU activations. The Cartesian coordinates of the agent, say x, which is part of the state, are used as an input, and the network is trained for $50,000$ epochs with a batch size of 1024.

Let $\psi : \mathbb{R}^{d_x} \to \mathbb{R}^K$ denote the ALLO network, and d_x denote the dimension of x. The ALLO intrinsic rewards are then generated using a potential-based formulation:

$$\tilde{R}_k(s, a) = (-1)^{k-1} \, \mathbb{E}_{s' \sim P(\cdot|s,a)} \Big[\psi_{\lceil k/2 \rceil}(x') - \psi_{\lceil k/2 \rceil}(x) \Big], \tag{69}$$

where $(-1)^{k-1}$ alternates the sign ($+1$ for odd k, -1 for even k), and the index $\lceil k/2 \rceil$ extracts the corresponding dimension of the network output ψ. The resulting ALLO intrinsic rewards used in our experiments are visualized in Figure 11. We generally see that each ALLO intrinsic reward can guide the agent toward different distinct regions of the state space; together, they provide sufficient exploration in our environments.

[4] https://github.com/tarod13/laplacian_dual_dynamics

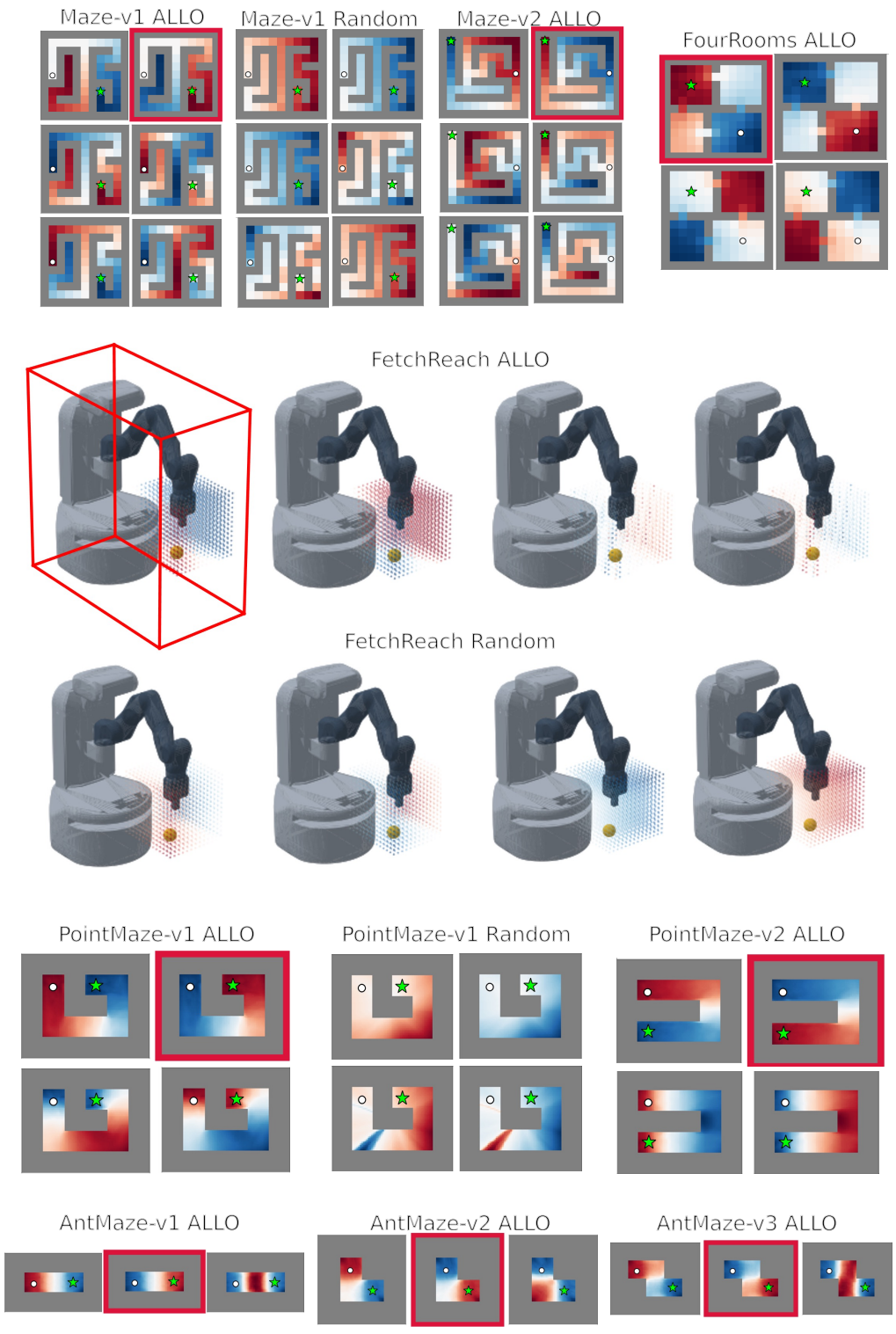

Figure 11: We visualize the intrinsic rewards generated by ALLO, normalized to the range $[-1, +1]$. In the colormap, red corresponds to $+1$, black to 0, and blue to -1. We indicate the chosen intrinsic reward by PPO-ALLO with a red box, and show sets of random intrinsic rewards used in our experiments.