Enhancing Exploration via Off-Reward Dynamic Reference Reinforcement Learning

Yamen Habib
Department of Communication
and Information Technologies
Universitat Pompeu Fabra
Barcelona, Spain
yamen.habib@upf.edu

Dmytro Grytskyy
Department of Communication
and Information Technologies
Universitat Pompeu Fabra
Barcelona, Spain
dmytro.grytskyy@gmail.com

Rubén Moreno-Bote
Serra Húnter Fellow Programme
Department of Communication
and Information Technologies
Universitat Pompeu Fabra
Barcelona, Spain
ruben.moreno@upf.edu

Abstract

In reinforcement learning (RL), balancing exploration and exploitation is essential for maximizing a target reward function. Traditional methods often employ regularizers to prevent the policy from becoming deterministic too early, such as penalizing deviations from a static reference policy. This paper introduces a novel approach by *jointly* training an off-reward dynamic reference policy (ORDRP) with the target policy, using a distinct reward function to guide exploration. We employ Kullback–Leibler divergence between the target policy and the dynamic reference policy as a regularization mechanism. Crucially, we provide a formal proof of convergence for the ORDRP iteration method, establishing its theoretical soundness. Our approach is validated within an actor-critic framework, with the ORDRP trained either using the maximum occupancy principle or Laplacian intrinsic off-rewards. Experimental results in challenging environments demonstrate that incorporating a jointly trained ORDRP enhances exploration, resulting in superior performance and higher sample efficiency compared to state-of-the-art baselines. These findings highlight the benefits of learning the reference policy alongside the main policy, leading to improved learning outcomes. Project page: https://yamenhabib.com/ORDRP/

1 Introduction

Balancing exploration and exploitation is a critical challenge in RL, as it directly influences the efficiency and success of the learning process [46]. Ideally, an RL agent should have two sources of knowledge to solve this problem: knowledge about what the best actions are to maximize reward given current learning state, and knowledge about which actions should be explored to enhance future learning. The latter can be separately crystallized in what we call a *reference* policy, that is, a policy that indicates to the agent which actions could be promising to explore given current learning state. Both pieces of knowledge are synthesized in a *target* policy, the policy that is used by the agent to actually generate actions with the only aim of maximizing reward. However, designing an effective

17th European Workshop on Reinforcement Learning (EWRL 2024).

reference policy that incorporates relevant prior information and that it adapts to the new action-state regions that become more likely through learning can be a quite challenging problem.

In traditional RL, exploration techniques such as ϵ-greedy or entropy maximization encourage agents to explore actions uniformly and randomly, yet these methods can result in inefficient and aimless exploration because they do not distinguish between second-best, third-best, and so on, actions [46, 56]. More recent model-free deep RL algorithms employ more sophisticated strategies. On-policy algorithms like TRPO [41] and PPO [43] explore by sampling actions according to the latest version of their target policy, which effectively serves as the reference policy. Early in training, the target policy exhibits high randomness, but as it learns to exploit rewards, it becomes less random, risking convergence to local optima due to early collapse of exploration. In these approaches, the target and reference policies become so similar that there is no clear separation between reward maximizing and exploration knowledge. In A3C [30], exploration is enhanced by parallel agents, entropy regularization, and on-policy learning with stochastic policy gradients. TRPO, PPO, and A3C suffer from poor sample efficiency due to on-policy learning, requiring new samples to be collected for nearly every policy update, which leads to rapidly escalating costs. REPS [36] improves the target policy search by avoiding large departures (in terms of relative entropy) between the target and reference action-state distributions, making it applicable for both on-policy and off-policy learning. However, REPS may still suffer from reduced exploration over time as the observed data distribution becomes prematurely biased towards reward exploitation. Off-policy methods such as DDPG [26] and TD3 [9] train deterministic policies and enhance exploration by adding noise to their actions during training. This noise, often Gaussian, ensures a wider variety of actions is explored [44], but it is aimless and difficult to tune, potentially leading to suboptimal exploration and learning efficiency. Soft Actor-Critic (SAC) [17], a state-of-the-art off-policy algorithm, incorporates entropy regularization into its objective function to balance exploration and exploitation. However, this regularization effectively introduces a uniform reference policy [35], impeding finer adjustments to the novel regions of action-state space being visited.

All this previous research has highlighted the importance of using reference policies to enhance exploration. These reference policies can be broadly categorized into two types: static and dynamic. A *static* reference policy, due to its simplicity, has been the more frequently employed approach ([17, 35]). Examples include the uniform policy in SAC [17] and G-learning [10], and MaxEnt [16]. While a static reference policy can consistently provide guidance throughout the learning process of the target policy, it risks becoming outdated as the target policy improves and the explored regions of the action-state space evolve. Alternatively, some approaches have used *dynamic* reference policies ([14, 41, 43]). This dynamic nature ensures that the reference policy evolves in response to the target policy's progress, preventing the exploration strategy from becoming obsolete. This adaptability can enhance sample efficiency by focusing exploration efforts on relevant areas of the state space that are actively visited by the target policy, thereby improving efficiency and stability of learning. However, the integration of prior knowledge about beneficial explorative actions and safety measures into dynamic reference policies remains a largely unexplored field. To address this, we propose a novel approach: injecting prior knowledge in the design of the dynamic reference policy by having it maximize a *different* reward function than the target policy. We introduce *off-reward dynamic reference policy* (ORDRP) search to accelerate learning and maintain active exploration over extended periods. By carefully designing the off-reward function, we can develop a dynamic reference policy that possesses more information about potentially useful exploration routines while being more specialized towards the trajectories generated by the target policy.

Figure 1 illustrates our intuition behind the superiority of ORDRP search. We assume that the target policy π is improved at each iteration to maximize reward, but there is also a penalty for deviating too much from the current reference policy μ. We will assume (more details to come) that the penalty increases with the Kullback-Leibler (KL) difference between the new target and the reference policy, $\mathcal{D}_{\text{KL}}\left(\pi(\cdot|s)||\mu(\cdot|s)\right)$. The reference policy μ can be static, like a uniform function U (Fig 1A), or dynamic, such as the latest target policy π_{n-1} obtained after $n-1$ update iterations (Fig 1B), or any other dynamic reference policy μ_n at current iteration step n (Fig 1C). The quantity $\pi_n(a_t|s_t)\log\frac{\pi_n(a_t|s_t)}{\mu(a_t|s_t)}$ in the KL penalty term changes its sign and magnitude depending on the difference in assigned probabilities between the target policy and the reference policy, providing adaptive guidance. When using a static uniform reference policy (Fig. 1A), the KL penalty tends to greatly reduce the likelihood of an action that has been discovered to be good by the target policy (peak of the Gaussian), while it increases the likelihood of harmful or useless actions due to the

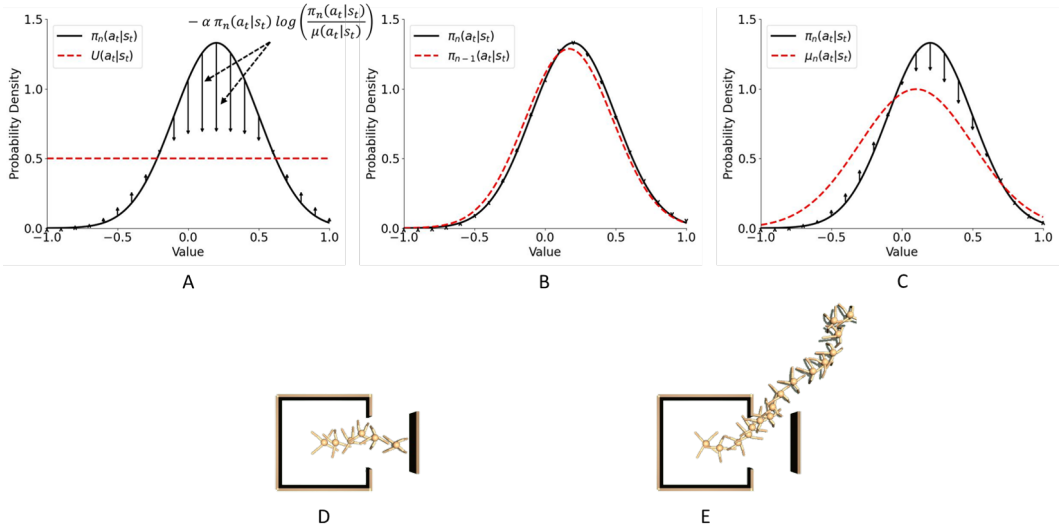

Figure 1: Visualization of the the strategic use of varying reference policies in ORDRP search, and how the target policy π is iteratively improved by balancing reward maximization against penalties for deviation from the reference policy. The penalties are informed by the Kullback-Leibler divergence, reflecting differences in policy probabilities. **A** demonstrates a static uniform policy U, strongly penalizing deviations that could otherwise optimize actions. **B** shows minimal divergence using the previous iteration's policy π_{n-1} as a reference, which stabilizes improvements with limited exploration. **C** introduces a dynamic policy μ_n, enabling flexible adaptations and encouraging explorioatn of new actions. Panels D and E detail the KL penalty's impact on action selection, emphasizing the strategic use of reference policies for effective policy optimization. (see videos here).

uninformative regularization term (left tail of the Gaussian). When using the current target policy as the reference (Fig. 1B), there are minimal differences between the new target policy and the reference policy, which do not significantly enhance exploration beyond what has already been learned. In contrast, a dynamic reference policy that lies between purely uninformative and purely target-aligned references (Fig. 1C) can inject relevant information about historically useful actions while remaining flexible. This suggests that for the reference policy to be effective, it should follow a different reward function than the target one (i.e., being off-reward), such that it retains knowledge about generally good and safe actions while dynamically adapting (i.e., being dynamic) as new regions of action-state space are explored during learning.

To empirically test these ideas, we used a high-dimensional control problem [7] with an ORDRP featuring an off-reward function based on the maximum occupancy principle (MOP) ([39, 32]), where the off-reward is defined as the cumulative future action entropy (see Sec. 5). While a uniform static reference causes the ant to get stuck against a wall (Fig. 1E), ORDRP search enables the agent to discover and exploit a more varied set of routes, allowing it to escape from the central zone (Fig. 1F) (see videos). These results demonstrate that even a simple variation of a standard benchmark (e.g., Mujoco [7]) renders exploration highly ineffective for standard approaches like SAC [17], highlighting the need for more sophisticated exploration techniques such as our ORDRP. In Sec. 7, we show that ORDRP surpasses stable state-of-the-art approaches even when tested on the unmodified standard benchmark.

Novelty. In this paper, we introduce an off-reward dynamic reference policy (ORDRP) KL regularization as a novel way to guide exploration in a more flexible way while injecting prior information about generally useful actions. We propose two off-reward functions: one based on MOP [39, 32], which maximizes the occupancy of action-state space by maximizing cumulative action entropy, and the other based on the graph Laplacian (Lap) [53, 28, 24], which leverages spectral information from the Laplacian of the transition matrix to guide exploration by encouraging the agent to visit spectrally distinct states. The superiority of ORDRP over previous regularization schemes is evident in high-dimensional environments with simple boundaries, demonstrated using both MOP and Lap

(Fig. 1E-F; Fig. 2). ORDRP search scales up to these high-dimensional control problems due to significant advancements in the actor-critic methodology. We also prove the convergence of our main ORDRP regularization scheme for any convergent dynamic reference policy. A key advantage of ORDRP is its potential for increased sample efficiency. By leveraging insights from the off-reward dynamic reference policy, our method reduces redundant exploration, making better use of available data and accelerating the learning process. ORDRP represents a notable departure from conventional exploration techniques like SAC [17], OAC [8], WCSAC [54], and GAC [47]. Unlike these methods, which typically tie the reference policy closely to the main target policy objective (e.g., TRPO [41], PPO [43], and GAC [47]), ORDRP employs a distinct off-reward function that is not directly connected to the primary target policy objective, thus promoting more diverse and effective exploration strategies. See Appendix 9.1 for a detailed comparison with other approaches.

2 Preliminaries

Our RL problem is formulated as a target policy optimization task within a Markov decision process (MDP), characterized by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$. We consider both the state space $\mathcal{S}$ and the action space $\mathcal{A}$ to be discrete domains and finite in our proofs, but in the experiments we consider continuous action-state spaces. The function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1)$ denotes the probability density $p(s_{t+1}|s_t, a_t)$ for transitioning to a subsequent state $s_{t+1} \in \mathcal{S}$, given the present state $s_t \in \mathcal{S}$ and an action $a_t \in \mathcal{A}$. Each state transition is associated with a reward, as defined by the function $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$. Finally γ is a discounted factor in [0, 1). Furthermore, $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ represent respectively the time-dependent state and state-action marginal distributions of the trajectory induced by a policy $\pi(a_t|s_t)$. Recall that the standard RL objective is $\pi^* = \arg\max_\pi \mathbb{E}_{s_0 \sim \rho_\pi}[V(s_0)] = \arg\max_\pi \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{(s_t,a_t)\sim p_\pi}[r(s_t, a_t)]$. Regularized RL augments the reward with a regularization term, such that the optimal regularized policy aims to maximize

$$\pi^*_{\mathcal{R}} = \arg\max_\pi \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{(s_t,a_t)\sim p_\pi}[r(s_t, a_t) - \alpha\, \mathcal{R}(\pi(\cdot|s_t))]\,, \tag{1}$$

where α is a positive temperature parameter, determining the relative importance of the regularization term against the reward. In this work, we study the regularizer function $\mathcal{R}(\pi(\cdot|s_t)) = \mathcal{D}_{KL}\,[\pi||\mu]$, which ensures that policy π is not far from another policy μ that we call the reference policy. We start with the definition of the regularized state-value function of a state s under a policy π with a possibly dynamic reference policy μ, denoted $\tilde{V}$, as

$$\tilde{V}(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{(s_t,a_t)\sim p_\tau}[r(s_t, a_t) - \alpha \log \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}], \quad s_0 = s\ . \tag{2}$$

The optimal regularized value function $\tilde{V}^*(s)$ should satisfy the corresponding regularized optimal Bellman equation for all $s \in S$,

$$\tilde{V}^*(s_t) = \max_\pi \mathbb{E}_{a_t\sim\pi}\left[r(s_t, a_t) - \alpha \log \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} + \gamma\, \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})]\right]\ . \tag{3}$$

Theorem 1. ***Optimal Policy Solution***. *Given a static reference policy μ, the optimal policy π^* and optimal state value function $\tilde{V}^*$ that solve the maximization problem in 3 exist and are unique. They are defined as*

$$\pi^*(a_t|s_t) = \frac{\mu(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)}{\sum_{a_t \in A} \mu(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)}\ , \tag{4}$$

$$\tilde{V}^*(s_t) = \alpha \log \sum_{a_t \in A} \mu(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)\ . \tag{5}$$

Proof: See [3, 34, 27]. See proof in the appendix 9.2.1 for completeness.

3 Off-Reward Dynamic Reference Value Iteration

Implementing a separate reference policy offers significant advantages over the traditional approach of integrating bonus rewards for exploration or safety in RL. This methodology isolates the specific effects of reward functions on environmental dynamics, an analysis unfeasible when rewards are merely appended to the primary objective. By employing a reference policy, researchers can conduct a detailed analysis of its behavior—whether by deploying it independently within the environment or by examining, for instance, its state-action values. This not only clarifies how the reference policy impacts the target policy but also helps identify the most effective reward function for the given task, enhancing the clarity of the interaction between different incentives and agent behavior. Moreover, The use of distinct discount rates between the target and reference policies enables precise behavioral calibration tailored to training objectives. For example, a higher discount rate allows the target policy to focus on long-term rewards, while the reference policy can concentrate on immediate exploratory actions. This dual approach not only customizes the learning process but also strategically flexes the exploration-exploitation balance.

Furthermore, employing KL divergence to maintain a specified deviation between the target and reference policies—akin to adjusting the temperature in models like Soft Actor-Critic—provides dynamic adjustment capabilities. This method allows for precise tuning of the exploration bonus, removing the need to manually test multiple settings to achieve the optimal balance. The reusability of the reference policy across various tasks also enhances training efficiency for new agents in related scenarios. Overall, adopting a separate reference policy presents a more structured, insightful, and efficient method for advancing reinforcement learning endeavors.

Standard proofs of existence, uniqueness and convergence of Equation 3 assume that the reference policy is static. If the reference policy μ is simultaneously learnt as the target policy π is also optimized, convergence of the latter can be compromised. In this first section, we prove that if μ converges, so does the target policy π by using a standard value iteration algorithm. The advantage of this proof is that neither the reference nor the target policies need to frozen at any update step, and thus they can simultaneously be learnt.

In our approach, the dynamic reference policy μ has its own distinct off-reward function defined based on the same MDP. We assume that the series of training reference policies $\{\mu^n\}_{n=0}^{\infty}$ converge to an optimal policy μ^* defined as $\mu^* = \arg\max_\mu \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{(s_t,a_t)\sim p_\pi}[f(\mathcal{M})]$. Here, $f(\mathcal{M})$ represents a function defined over the Markov Decision Process (MDP), which is characterized by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, \text{off-}r, \gamma)$ with a given off-reward function off-r, which will be specified in Sec. 5.

We leverage value iteration, a dynamic programming technique that applies 'value backups' to create a series of value functions, which are functions defined over the state space, in a recursive process ([46, 33]). Our method ensures to concurrently learn both the state value function and the target policy π as the reference policy μ evolves, ensuring convergence without the need to freeze either policy during updates. Inspired by equation 5, we define $\tilde{V}^{n+1}(s_t)$ and the dynamic reference Bellman operator B_{μ^n}, for $\alpha > 0$, by

$$\tilde{V}^{n+1}(s_t) = (B_{\mu^n}\tilde{V}^n)(s_t) = \alpha \log \sum_{a_t \in A} \mu^n(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^n(s_{t+1})])\right). \tag{6}$$

Notice that we build state value function $\tilde{V}^{n+1}$ using the function $\tilde{V}^n$ and the reference policy μ^n.

Lemma 1. ***(Convergence)*** *Given a sequence of reference policies μ^n that converges to μ^∞, that is,*

$$\forall \epsilon > 0, \exists N \in \mathcal{N} \cup \{0\} : \forall n > N, \max_{a\in\mathcal{A}, s\in\mathcal{S}} |\mu^n(a|s) - \mu^\infty(a|s)| < \epsilon\,, \tag{7}$$

then the state value function $\tilde{V}^n$ defined in equation 6 converges as μ^n converges to μ^∞.

In Addition, we define the target policy π^n as

$$\pi^{n+1}(a_t|s_t) = \frac{\mu^n(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^n(s_{t+1})])\right)}{\sum_{a_t\in A}\mu^n(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^n(s_{t+1})])\right)}. \tag{8}$$

π^n converges to a limit policy π^∞.

Proof: See Appendix 9.2.2.

Theorem 2. ***(Dynamic Reference Value Iteration)*** *Given any initial state value function $\tilde{V}^0$, iteratively applying the dynamic reference Bellman operators B_{μ^n}, defined in equation 6, until convergence guarantees that the process reaches the fixed point of the Bellman operator B_{μ^∞}.*

Proof: See Appendix 9.2.2.

Equipped with the above proof, we have now guarantees that the problem of simultaneously learning the reference and target policies is well-defined. It lies the basis for the development of the dynamic reference policy iteration and dynamic reference actor-critic methods described next, important for practical algorithms that can be effectively implemented in complex RL scenarios.

4 Off-Reward Dynamic Reference Policy Iteration

Here we will only present the off-reward dynamic reference policy iteration, and later we will show the convergence of this method (Appendix 9.2.3) and how we build our off-reward dynamic reference actor-critic (Appendix 9.3.4). Our off-policy dynamic reference actor-critic algorithm is derived from a variant of the policy iteration method that takes dynamic reference policy learning into consideration. We will treat the temperature α as constant and later in Appendix 9.3.3 as a variable to control the similarity between the target and the reference policy defined by Kullback–Leibler divergence.

ORDRP iteration is derived directly from the vanilla policy iteration algorithm [46]. Let us define the objective function J for the target policy to take into consideration the dynamic reference policy as $J(\pi, \mu) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_{t=0}^{T}\gamma^t\left(r(s_t,a_t) - \alpha\log(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)})\right)\right]$. In the policy evaluation step of dynamic reference policy iteration, we wish to compute the value of a policy π according to the objective function $J(\pi, \mu)$. For a fixed target π policy and dynamic reference policy μ^n, the state action value function Q_{π/μ^n} can be computed iteratively, starting from any function $Q_{\pi/\mu^n} : S \times A \to \mathbb{R}$ and repeatedly applying a modified Bellman backup operator $\mathcal{T}$ given by

$$\mathcal{T}Q_{\pi/\mu^n}(s_t, a_t) \doteq r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\pi/\mu^n}(s_{t+1})]\,, \tag{9}$$

where

$$V_{\pi/\mu^n}(s_t) = \mathbb{E}_{a_t\sim\pi}[Q_{\pi/\mu^n}(s_t, a_t) - \alpha\log(\frac{\pi(a_t|s_t)}{\mu^n(a_t|s_t)})]\,. \tag{10}$$

During the policy improvement phase, and inspired by equation 4, we adjust the policy in the direction of the exponential of the state action value function $Q_{\pi/\mu}$ weighted by the dynamic reference policy μ. This update helps the policy to enhance with respect to its value $Q_{\pi/\mu}$. We Follow [17] by imposing additional limitations on the policy to a specific subset of policies Π which correspond to the parameterized group of Gaussians distributions. To integrate the stipulation that π is within Π, we map the enhanced policy onto the preferred subset of policies using Kullback-Leibler divergence. In other words, during the policy improvement phase, we update the policy state-by-state using the following update rule

$$\pi_{new} = \underset{\pi\in\Pi}{\operatorname{argmin}}\,\mathcal{D}_{\text{KL}}\left(\pi(\cdot|s_t)\middle|\middle|\frac{\mu_{\text{new}}(\cdot|s_t)\exp\left(\frac{1}{\alpha}Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,\cdot)\right)}{Z_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t)}\right)\,, \tag{11}$$

where $Z_{\pi_{\text{old}}/\mu_{\text{new}}}$ is the partition function that normalizes the distribution.

The full ORDRP iteration algorithm alternates between policy evaluation and policy improvement steps (Appendix 9.2.3), and it will provably converge to the optimal maximum policy among the policies in Π (Appendix 9.2.3).

5 Off-Reward Dynamic Reference Policies

While numerous exploration methods are available in RL —ranging from intrinsic rewards, count-based exploration, and uncertainty estimation— our research specifically aims to optimize the exploration-exploitation trade-off using two innovative methodologies. The first, known as the MOP reference and denoted by μ_{MOP}, focuses on maximizing the cumulative entropy of its policy, providing a robust exploration strategy [39, 32]. The second, the Lap reference, denoted by μ_{Lap},

leverages a Laplacian operator derived from the policy's state transition distribution, allowing for a sophisticated approach that integrates both the structural and dynamic aspects of the environment [53]. These methods are specifically chosen not only for their distinct exploration capabilities but also for their potential to effectively balance between exploring new possibilities and exploiting acquired knowledge. Importantly, our framework is designed to be compatible with any off-policy exploration method, offering flexibility and adaptability in application. Full implementation details of these reference policies are outlined in Appendix9.3.4.

Note that in either case the reference policy μ has to be learnt in order to optimize the off-rewards defined below, but the reference policy is not allowed to make any actions in the environment, which is only done by the target policy simultaneously learnt. It is also important to note that the off-reward functions defined below change with the course of learning. Therefore they are tuned to the priors that are injected into the reward function construction to aim at exploration, but they also adapt to the regions of action-state space that are most frequently visited during learning.

MOP reference

The Maximum Occupancy Principle (MOP) is a novel approach to modeling agent's behavior, which diverges from traditional reward-maximization frameworks [39, 32]. The goal of MOP is to maximize the occupancy of future action-state paths, rather than seeking extrinsic rewards. This principle posits that agents are intrinsically motivated to explore and visit rare or unoccupied action-states, thus ensuring a broad and diverse range of behaviors over time. The off-reward function in MOP is the entropy of the paths taken by the agent,

$$R(\tau) = -\sum_{t=0}^{\infty} \gamma^t \ln \left(\mu_{\text{MOP}}^{\alpha}(a_t|s_t) p^{\beta}(s_{t+1}|s_t, a_t) \right) \ ,$$

where $\alpha > 0$ and $\beta \geq 0$ are weights for actions and states, respectively, and γ is the discount factor. The agent maximizes this intrinsic reward by preferring low-probability actions and transitions, which encourages exploration and the occupancy of a wide range of action-states. This intrinsic motivation leads to behaviors that appear goal-directed and complex without the necessity of explicitly defined extrinsic rewards. In our experiments, we do not have access to the state transition model as we are using a model-free algorithm, so we set $\alpha = 1$ and $\beta = 0$, thereby considering only the entropy of the policy. The MOP reference policy helps the target policy to perform actions that are discovered to lead to many future action paths, improving exploration and sampling efficiency by avoiding strongly constrained regions, such as terminal states.

Lap reference

Learning state representations in RL is important for effective exploration. A well-designed state representation captures the essential features and structure of the environment, enabling the agent to differentiate between meaningful and redundant information [53]. This clarity allows the agent to make more informed decisions about where to explore, focusing on areas of the state space that are likely to yield valuable information and potential rewards. Further, if this state representations is low-dimensional, it can enhance computational efficiency, speed up learning, improve generalization, and most importantly simplify policy and value function optimization. We can think of learning a low-dimensional state representation as building a mapping between the original space and a low-dimensional representation such that near/distant points in the original space are near/distant in the representation. This problem is referred to as *graph drawing* problem [51, 23].

In the approach taken here, valid for large and continuous spaces, we want to build a set of features $\phi(s) = (f_1(s), ..., f_d(s)) \in \mathbb{R}^d$ that faithfully represent the input space $s \in S$. As we are interested in agents that traverse that space S as they follow a current target policy π, we would like to consider nearby states as those states s_{t+1} that are visited from another state s_t in consecutive time steps. This graph drawing problem [51, 23] can be then formalized as finding the set of features such that the functional

$$G(f_1, \ldots, f_d) = \frac{1}{2} \mathbb{E}_{s_t \sim \rho_\pi, s_{t+1} \sim p} \left[\sum_{k=1}^{d} (f_k(s_t) - f_k(s_{t+1}))^2 \right]$$

is minimized under the constraint that the features are normalized vectors and that they are orthogonal to each other (i.e, orthonormal), that is $\sum_{j,k} \left(\mathbb{E}_{s \sim \rho} \left[f_j(s) f_k(s) \right] - \delta_{jk} \right)^2 = 0$, where δ_{jk} is the Kronecker delta. The first condition ensures that features do not become zero, and the second

condition promotes that the features are as different as possible from each other. Note that a trivial feature that would minimize the above functional is simply a constant function over states, and therefore this feature is removed altogether from the admissible list. It can be shown that the features minimizing the graph drawing functional correspond to the eigenvectors with the lowest eigenvalues of the graph Laplacian. Specifically, for a function $x \in \mathbb{R}^{\mathcal{S}}$ defined on the states, and given that each pair of states (s_t, s_{t+1}) has an associated transition probability $p(s_{t+1}|s_t) > 0$, the graph Laplacian quadratic form is given by $x^T L_G x = \sum_{(s_t, s_{t+1})} p(s_{t+1}|s_t)(x(s_t) - x(s_{t+1}))^2$[1]. Minimizing the graph drawing functional corresponds to finding similar representations for pairs of states s_t and s_{t+1} that are closely related in time under the current target policy, while the constraints favor that the representations of pairs of states randomly sampled are as different as possible.

In high-dimensional, complex environments, it is necessary to find these eigenvectors efficiently, even if it involves some loss of accuracy due to function approximations. Following [53], we use a stochastic gradient descent method along with relaxed constraints to solve the minimum graph drawing problem. In our case we chose to reduce the original $n-$ dimensional state space (e.g., 27 in Ant-v4, 376 in Humanoid-v4 [7]) into $d = 4$ dimensions. In Appendix 9.3.2, we present our implementation details.

Once the embedding function $\phi(s) = [f_1(s), \ldots, f_d(s)]$ has been learned, we ask our off-reward dynamic reference policy to maximize the expected cumulative reward $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$, where the instantaneous off-reward $R(s_t, a_t)$ is defined as

$$R(s_t, a_t) = \frac{||\phi(s_{t+1}) - \phi(s_t)||_2^2}{||\phi(s_{t+1})||_2^2 + ||\phi(s_t)||_2^2} \, .$$

This novel off-reward function consistently encourages the agent to transition to states that are furthest from the current one, which leads to improved exploration. This approach effectively prevents the agent from wasting time on inefficient transitions that result in minimal changes to the embedded state space, leading to a more sample efficient algorithm.

6 Implementation

Our ORDRP algorithm is implemented in a standard manner. We allow the target policy to interact with the environment, collect transitions, and store them in a replay buffer. The off-reward dynamic reference policy does not interact with the environment. During updates, the target policy, its embedding function (in the case of Lap reference) and the reference policy have access to this replay buffer. Every few interaction steps, we sample a batch from the replay buffer to update both the target and reference policies in an off-policy manner. We use SAC [17] to train our reference policy with the previously discussed reward functions. The implementation details for the MOP reference are provided in Appendix 9.3.1, and for the Lap reference in Appendix 9.3.2. Similarly to SAC [17], we employ the actor critic framework to train our target policy (Appendix 9.3.4). This off-policy actor critic framework allows us to separate the training of the target and reference policies (along with their necessary embedding functions), making our method generalizable and easily adaptable to any other reference policy as a plug-and-play component.

7 Results

In this section, we experimentally evaluate the impact of incorporating ORDRPs on exploration efficiency and learning outcomes. By using two distinct types of ORDRPs, MOP and Lap dynamic reference policies, we aim to demonstrate the increased sample efficiency of our RL algorithm across a range of tasks. Details on the hyper-parameters utilized can be found in Appendix 9.4.

Escape Room: Both reference policies in our algorithm are intended to improve the agent's exploratory behavior. To test this, we created a novel environment within the Mujoco engine to rigorously test our algorithm in a high-dimensional continuous domain, where effective exploration is crucial for task completion. This custom environment ensures that performance improvements are due to enhanced exploratory behavior and not other factors. Escape Room consists of a square room

[1]For discrete state space, the Laplacian matrix $L = D - M$, where D is the diagonal matrix of vertex degrees, and M is the adjacency matrix of the graph.

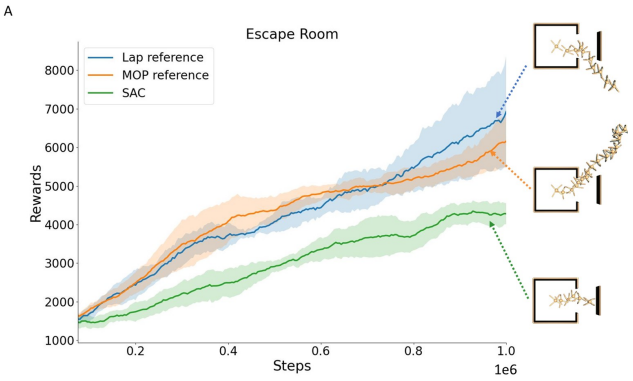

Figure 2: Effectiveness of different exploration policies on accumulated rewards:(A) The graph shows accumulated rewards, demonstrating that ORDRPs enhance exploration more effectively than SAC. Late frames of the agent's path using the Lap reference, MOP reference, and the agent's path using the SAC policy, showing less effective exploration in the last case.

with a door in one of its walls (Fig. 2). The agent's reward equals its distance from the room's center, incentivizing it to move as far away from the center as possible. To exit the room, the agent must navigate through the door, which requires it to explore and learn the layout. To add complexity and increase the exploration required, we placed an additional wall in front of the door, creating a more challenging path for the agent to find its way out. The outcomes are illustrated in (Fig.2 A), showing that both dynamic references assist the main target policy in learning how to escape the room (Fig. 2 (Lap reference), (MOP reference)) and earn higher rewards, leading to a more sample-efficient algorithm. Although SAC and MOP both aim to maximize entropy, having a separate reference contributes to improved algorithm performance (Fig.2 (MOP reference), (SAC)).

ORDRPs Vs Exploration Bonus: In this study, we compare two methods for encouraging exploration in reinforcement learning: adding an entropy-based exploration bonus to the main reward function and using the Kullback-Leibler (KL) divergence between the main target policy and a continuously updating reference policy that aims to maximize entropy. Notably, this KL divergence cannot be simplified to entropy regularization because the reference policy keeps updating. By employing a separate reference policy designed to maximize entropy, we facilitate more targeted and effective exploration strategies without compromising the primary objectives of the main policy, effectively separating exploration from goal-directed behaviors and preventing potential conflicts in reward optimization. The key distinction between using an entropy exploration bonus and employing a reference policy lies in their influence on the agent's behavior: the entropy bonus discourages the agent from becoming overly deterministic by preventing it from assigning high probabilities to specific actions (as illustrated in Fig. 1A), whereas the entropy-maximizing reference policy actively encourages the agent to explore states that maximize the expected future occupancy of action-state paths. To evaluate the differences in performance between these two approaches, we conducted experiments across five different Mujoco environments, training each with 10 seeds over 500,000 steps. Our results indicate that the reference policy approach is more sample-efficient and demonstrates improved performance metrics.

Table 1: MOP Reference vs. SAC Performance

Environment	MOP Reference	SAC
Ant	6166.43 ± 568.79	5754.86 ± 183.80
Humanoid	4972.94 ± 58.96	4761.54 ± 592.75
Walker2d	3656.10 ± 689.60	3347.21 ± 821.24
Hopper	2209.39 ± 48.26	2292.46 ± 891.73
Half Cheetah	8348.35 ± 1351.01	8048.29 ± 263.37
Mean	**5070.64 ± 543.32**	**4840.87 ± 550.58**

MuJoCo Benchmark Evaluation: We benchmark our approach against previously established methods, evaluating performance across an array of complex continuous control tasks sourced from the OpenAI Gym benchmark suite [7]. Our aim with this experimental evaluation is to assess how our method's sample complexity and stability stack up against earlier off-policy and on-policy deep RL algorithms in standard problems without any modifications. We compare our method with SAC [17], deep deterministic policy gradient (DDPG)[26], considered one of the most efficient off-policy

Table 2: Performance Across Mujoco benchmarks

Metric/Method	SAC	PPO	TD3	DDPG	ORDRP with Lap reference
Ant-v4	6427.60 ± 369.41	1929.61 ± 365.52	4378.93 ± 2180.22	413.12 ± 57.74	6695.32 ± 27.29
Humainoid-v4	5316.68 ± 63.88	661.88 ± 56.70	93.33 ± 0.01	127.60 ± 1.82	5489.51 ± 58.63
Walker2d-v4	4297.73 ± 1255.16	3444.32 ± 624.90	4172.49 ± 6.06	1449.51 ± 116.59	4558.64 ± 19.56
Hopper-v4	2766.41 ± 700.70	935.58 ± 204.37	3294.23 ± 95.69	1721.71 ± 2768.47	2394.10 ± 535.31
Cheetah-v4	9378.05 ± 136.61	5463.19 ± 420.92	10305.00 ± 485.72	11607.61 ± 534.42	10516.12 ± 105.35
	5637.30 ± 505.15	2486.92 ± 334.48	4448.80 ± 553.54	3063.91 ± 695.81	5930.74 ± 149.23

deep RL algorithms, and proximal policy optimization (PPO)[43], a stable and effective on-policy policy gradient method. Our evaluation also includes twin delayed deep deterministic policy gradient (TD3)[9].

Table 2 presents average outcomes from ten different agents across each method, with values reflecting the mean of the last five recordings taken at 4,000-step intervals. It is evident that the ORDRP methods excel in four of the five environments, with Lap reference policy achieving the highest average reward across all experiments. While the DDPG method surpasses Lap reference in the cheetah-v4 environment, its performance is suboptimal in all other scenarios, indicating that Lap reference policies is not only more stable but also exhibits better generalization across diverse tests.

8 Discussion and conclusion

The findings from our experiments highlight that even state-of-the-art methods often struggle with directionless exploration or overcommitment to suboptimal policies (see Figs. 1,2). Our ORDRP approach addresses these challenges by providing a reference policy that evolves in response to the learning environment. This adaptive nature allows for more structured exploration, reducing the risk of inefficient learning trajectories. Our study explored a novel approach to RL by introducing an off-reward dynamic reference policy to guide the main policy through complex learning environments. This reference policy evolves alongside the target policy, allowing for more targeted exploration and reducing the inefficiencies of traditional exploration methods. Our approach demonstrates that incorporating an off-reward reference policy into RL improves convergence rates and enhances performance in various tasks. The off-reward functions that we used, based on MOP and graph Laplacian, demonstrated to be much more efficient in exploring, learning and optimizing target rewards in slightly more involved environments than the ones typically considered (Fig. 1-2). The experiments conducted in this project, including the Escape Room scenario and MuJoCo Benchmark Evaluation, demonstrate the success of our method. Figure 4 (see Appendix 9.4) presents the learning curves of our method compared to our baselines, demonstrating that dynamic reference exhibits greater stability across various environments. The results indicated that our ORDRP generally outperformed traditional algorithms, leading to faster learning and higher final rewards.

Limitations. A primary limitation of ORDRP is its increased demand for computational resources, as it necessitates training an additional reference policy alongside the target policy. This requirement doubles the computational load, which may restrict the method's applicability in environments where resources are limited or rapid decision-making is essential. Unlike traditional policies that interact directly with the environment, the reference policy does not collect trajectories or make actions, requiring a fundamentally different approach to its design and evaluation. This divergence from conventional methods can complicate the engineering process, as the reference policy must effectively guide exploration without engaging in the environment itself. Moreover, ORDRP, in its current form, assumes a fixed Markov Decision Process (MDP), which implies stationary, fully observable dynamics. This assumption limits its effectiveness in non-stationary or partially observable environments. Extending ORDRP to handle non-stationary dynamics or partially observable scenarios remains an area for future research. Furthermore, crafting the off-reward structure for the reference policy is particularly challenging. It must be distinct enough from the target policy to drive meaningful exploration yet carefully calibrated to prevent convergence to suboptimal solutions. This delicate balance necessitates out-of-the-box thinking and a departure from established RL paradigms, potentially slowing adoption and integration into existing frameworks.

References

[1] J. Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

[2] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pages 151–160. PMLR, 2019.

[3] M. G. Azar, V. Gómez, and H. J. Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 13(103):3207–3245, 2012.

[4] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[5] D. Bertsekas. *Dynamic Programming and Optimal Control*. Number v. 1 in Athena scientific optimization and computation series. Athena Scientific, 1995.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. MIT License.

[8] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann. Better exploration with optimistic actor critic. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[9] S. Dankwa and W. Zheng. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In *Proceedings of the 3rd international conference on vision, image and signal processing*, pages 1–5, 2019.

[10] R. Fox, A. Pakman, and N. Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[11] K. Friston, J. Kilner, and L. Harrison. A free energy principle for the brain. *Journal of physiology-Paris*, 100(1-3):70–87, 2006.

[12] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[13] A. Galashov, S. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.

[14] A. Galashov, S. M. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess. Information asymmetry in kl-regularized RL. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[15] M. Geist, B. Scherrer, and O. Pietquin. A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR, 2019.

[16] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1352–1361. JMLR.org, 2017.

[17] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

[18] H. Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[19] H. v. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press, 2016.

[20] T. Jung, D. Polani, and P. Stone. Empowerment for continuous agent—environment systems. *Adaptive Behavior*, 19(1):16–39, 2011.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[23] S. Kirmani and K. Madduri. Spectral graph drawing: Building blocks and performance analysis. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 269–277, 2018.

[24] M. Klissarov and M. C. Machado. Deep Laplacian-based options for temporally-extended exploration. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17198–17217. PMLR, 23–29 Jul 2023.

[25] A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pages 128–135. IEEE, 2005.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[27] J. Ma. The point to which soft actor-critic converges, 2023.

[28] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2295–2304. JMLR.org, 2017.

[29] A. Miller, N. Foti, A. D' Amour, and R. P. Adams. Reducing reparameterization gradient variance. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.

[32] R. Moreno-Bote and J. Ramirez-Ruiz. Empowerment, free energy principle and maximum occupancy principle compared. In *NeurIPS 2023 workshop: Information-Theoretic Principles in Cognitive Systems*, 2023.

[33] R. Munos and C. Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(27):815–857, 2008.

[34] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2772–2782, Red Hook, NY, USA, 2017. Curran Associates Inc.

[35] E. Noorani and J. S. Baras. Risk-sensitive reinforcement learning and robust learning for control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2976–2981, 2021.

[36] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1607–1612. AAAI Press, 2010.

[37] A. Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020. MIT License.

[38] R. Raileanu and T. Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

[39] J. Ramírez-Ruiz, D. Grytskyy, and R. Moreno-Bote. Seeking entropy: complex behavior from intrinsic motivation to occupy action-state path space. *arXiv preprint arXiv:2205.10316*, 2022.

[40] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

[41] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[42] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[44] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR.

[45] S. Still and D. Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131:139–148, 2012.

[46] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[47] V. Tangkaratt, A. Abdolmaleki, and M. Sugiyama. Guide actor-critic for continuous control. *arXiv preprint arXiv:1705.07606*, 2017.

[48] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12163–12174, 2020.

[49] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. In *Neural Information Processing Systems*, 2020.

[50] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[51] K. Wang, K. Zhou, Q. Zhang, J. Shao, B. Hooi, and J. Feng. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11003–11012. PMLR, 18–24 Jul 2021.

[52] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[53] Y. Wu, G. Tucker, and O. Nachum. The laplacian in rl: Learning representations with efficient approximations. *ArXiv*, abs/1810.04586, 2018.

[54] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10639–10646, 2021.

[55] G. Zhu, Z. Lin, G. Yang, and C. Zhang. Episodic reinforcement learning with associative memory. 2020.

[56] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

9 Appendix

9.1 Related Work

Actor-critic methods, initially introduced by [4] and [46], have been enhanced by integrating deep learning [30]. Further enhancements include structural improvements [50], advanced value approximations [42], and the incorporation of off-policy elements [52]. Refinements such as trust region application [41] and bias mitigation in off-policy methods [55] have also advanced the theory and application of actor-critic techniques [15, 2, 48].

The Soft Actor-Critic (SAC) approach addresses model-free RL by utilizing a stochastic policy framework that optimizes both the entropy and expected return, fostering a balance between exploration and exploitation [17]. The Optimistic Actor-Critic (OAC) algorithm enhances model-free RL by leveraging an upper confidence bound of the critic to improve exploration, effectively mitigating issues like pessimistic underexploration [8]. Both SAC and OAC implement a KL constraint between the reference policy and the target policy. In SAC, this is done using a uniform reference policy, while in OAC, the reference policy is a Gaussian distribution $N(\mu_E, \Sigma_E)$, where μ_E is shifted from the target policy mean μ_T to optimize the upper bound of the Q-function. The covariance Σ_E remains the same as the target policy's Σ_T, ensuring consistent exploration variance. Additionally, the Worst-Case Soft Actor Critic (WCSAC) introduces a safety critic (instead of a reference policy) that optimizes policies using Conditional Value-at-Risk (CVaR), providing a robust framework for high-safety environments [54]. Finally, the Guide Actor-Critic (GAC) method leverages second-order information from the critic to refine policy updates effectively. It uses a guide actor, learned to locally maximize the critic under a Kullback-Leibler (KL) divergence constraint, to enhance the learning process [47]. All these methods aim to enhance the performance of the target policy using either another critic or another policy, but to the best of our knowledge, none has tried a completely separate policy with its own off-reward function as we propose in this work.

G-learning [10], an off-policy RL algorithm that incorporates a reference policy to regularize value estimates, helps to mitigate bias caused by deterministic policies during the initial learning phases. G-learning achieves smoother exploration and better convergence rates than Q-learning by including a penalty term for divergence from a simple stochastic prior policy. A KL-regularized reward objective in RL that incorporates a static reference policy along with the learned target policy promotes structured learning and faster training [13]. KL regularization not only facilitates smoother updates but also averages Q-values, enhancing learning stability and convergence [49]. Other work has used a dynamic reference policy [41, 49] designed to be the most recent version of the learned target policy. In contrast to these methods, we use a dynamic reference policy that is learned to maximize a different off-reward function from the one that the target policy is designed to maximize. This methodological variation allows for a broader exploration of policy space and more tailored optimization strategies, promoting enhanced learning outcomes in complex environments (see results in section 7).

Our method is compatible with any state-of-the-art model-free off-policy exploration approach, enhancing the relevance and impact of our findings. For this study, however, we selected two novel methods for evaluation. The first method, MOP [39, 32], diverges from traditional techniques like empowerment [25, 20, 45] and the free energy principle [11], which typically favor deterministic policies. MOP supports dynamic, goal-oriented behavior by incorporating behavioral variability and accounting for the physical limitations of embodied agents, thus facilitating more adaptive and engaging interactions with the environment. It dynamically adjusts to environmental feedback to maximize the entropy of action-state paths, promoting diverse behaviors over the deterministic tendencies of uniform policies. Importantly, MOP does not merely maximize entropy at each time step; instead, it balances entropy maximization with the agent's objectives to encourage effective exploration. By leveraging MOP's objective, we were able to compare the effects of incorporating entropy through two distinct mechanisms: adding an entropy-based exploration bonus to the main reward function, and introducing a continuously updating reference policy that aims to maximize entropy. This comparison allowed us to evaluate how different methods of integrating entropy influence the agent's exploration behavior and overall performance. Our second method employs a Laplacian representation to generate an intrinsic reward based on the differences between consecutive state representations. While closely related to RIDE [38], which utilizes forward and inverse dynamics models to learn state representations that focus solely on agent-influenced environmental elements, our method leverages the Laplacian representation to capture the spectral information of the state space without considering the agent's influence.

9.2 Proofs

9.2.1 Optimally and Properties of Predefined reference policies

In this section, we reintroduce Theorem 1 from the main body and provide its proof.

Theorem 1. ***Optimal Policy Solution***. *Given a static reference policy μ, the optimal policy π^* and optimal state value function $\tilde{V}^*$ that solve the maximization problem in 3 exist and are unique. They are defined as*

$$\pi^*(a_t|s_t) = \frac{\mu(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)}{\sum_{a_t\in A}\mu(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)}\,, \tag{12}$$

$$\tilde{V}^*(s_t) = \alpha\log\sum_{a_t\in A}\mu(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)\,. \tag{13}$$

Proof: A similar proof was presented in [3, 34, 27]. We write ours here for completeness. Following the methodologies outlined in [27], we adopt a Lagrangian approach to tackle the constrained optimization problem essential to defining the optimal policy π^* and the state value function $\tilde{V}^*$. Let's write out our constrained optimization problem

$$\underset{\pi}{\text{maximize}} \quad \sum_{a_t\in A}\pi(a_t|s_t)\left[r(s_t,a_t)-\alpha\log\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}+\gamma\mathbb{E}_{s_{t+1}\sim p}\left[\tilde{V}^*(s_{t+1})\right]\right] \tag{14}$$

$$\text{s.t.} \quad \pi(a_t|s_t) > 0, \qquad \forall s_t, a_t \tag{15}$$

$$\sum_{a_t\in A}\pi(a_t|s_t) = 1. \qquad \forall s_t \tag{16}$$

We define the relaxed Lagrangian function

$$L(s_t;\nu) = \sum_{a_t\in A}\pi(a_t|s_t)[r(s_t,a_t)-\alpha\log\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})]]-\nu(\sum_{a_t\in A}\pi(a_t|s_t)-1)\,. \tag{17}$$

Since $-x\log(x)$ is strictly concave function, then we have a convex optimization problem. We can solve this problem by taking the derivatives of the relaxed Lagrangian function and setting them equal to zero given by

$$0 = \frac{\partial L(s_t;\nu)}{\partial\pi(a_t|s_t)} = r(s_t,a_t)-\alpha\log\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}-\alpha+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})]-\nu\,. \tag{18}$$

The solution is

$$\pi^*(a_t|s_t) = \mu(a_t|s_t)\exp\left(\frac{-\nu}{\alpha}-1\right)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)\,. \tag{19}$$

With the equality constraint

$$\sum_{a_t\in A}\pi^*(a_t|s_t) = 1\,, \tag{20}$$

and by applying log transformation on both sides, we can solve for the multiplier as

$$\nu = \alpha\log\sum_{a_t\in A}\mu(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)-\alpha\,, \tag{21}$$

and inserting this into Equation 21 we get

$$\pi^*(a_t|s_t) = \frac{\mu(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)}{\sum_{a_t\in A}\mu(a_t|s_t)\exp\left(\frac{1}{\alpha}(r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right)}\,. \tag{22}$$

Finally, inserting this result into Equation 3, we obtain

$$\tilde{V}^*(s_t) = \alpha \log \sum_{a_t \in A} \mu(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^*(s_{t+1})])\right). \tag{23}$$

The solution $\pi^*(a_t|s_t)$ strictly satisfies the constrains of the original problem and KKT conditions (where $\gamma = 0$ for the inequality condition), thus it's the optimal solution.

We now highlight several important observations that provide new insights into the theoretical framework of Markov Decision Processes (MDPs):

Remark 1: Continuity of π^* and $\tilde{V}^*$

The optimal policy π^* and the value function $\tilde{V}^*$ exhibit continuity with respect to the policy sequence μ. This continuity is crucial for understanding the behavior of iterative policy refinement methods, where policies are progressively updated. It signifies that as the sequence of policies μ^n converges towards its asymptotic limit μ^∞, the corresponding optimal policies $(\pi^n)^*$ and value functions $(V^n)^*$ derived at each iteration also converge. This result ensures the stability and reliability of using iterative methods for policy improvement in practical applications, where incremental modifications to the policy do not lead to disproportionate changes in the resultant policy performance.

Remark 2: Convergence of Optimal Solutions in Sequential MDPs

Each policy μ^n effectively defines a distinct Markov Decision Process, denoted as MDP_n. The progression of these MDPs mirrors a path towards an ultimate target problem, MDP_∞. Our analysis establishes that the optimal solutions for these intermediate problems converge towards the optimal solution of MDP_∞. This observation not only validates the theoretical underpinnings of evolving MDP models but also reassures that practical implementations where policies evolve or are learned over time are based on a solid theoretical foundation.

9.2.2 Off-Reward Dynamic Reference Value Iteration

In the main body of this paper, we introduced a dynamic approach to training reference policies μ concurrently with the main target policy π within an MDP framework. This method is designed to effectively adapt to changing environment dynamics and addresses the limitations of static reference policies by ensuring ongoing alignment and optimization with respect to the target policy's objectives. In this appendix, we delve deeper into the technical implementation and mathematical proof of dynamic reference value iteration method and how it can be concurrently optimized to achieve our stated goals. We base our proof on the analytical framework provided by [34].

Now consider the dynamic reference Bellman operator B_{μ^n}, for $\alpha > 0$, by

$$(B_{\mu^n}\tilde{V}^n)(s_t) = \alpha \log \sum_{a_t \in A} \mu^n(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^n(s_{t+1})])\right).$$

Lemma 2 is important in demonstrating the bounded nature of the state value function $\tilde{V}^n$, which is essential for ensuring the convergence and stability of our value iteration method.

Lemma 2. *The state value function $\tilde{V}^n$ defined in equation 6 is bounded.*

Proof: Let's start by a randomly initialized state value function $\tilde{V}^0$ where $\forall s \in \mathcal{S}$, $\tilde{V}^0(s) \leq C$ and $C > 0$.

We notice that $\tilde{V}^1$ is bounded from above because

$$\tilde{V}^1(s_t) = \alpha \log \sum_{a_t \in A} \mu^0(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^0((s_{t+1})])\right) \tag{24}$$

$$\leq \alpha \log \sum_{a_t \in A} \mu^0(a_t|s_t) \exp\left(\frac{1}{\alpha}(r_{\max} + \gamma C)\right) \tag{25}$$

$$= r_{\max} + \gamma C\,, \tag{26}$$

and the same for V_{π^2/μ^2}

$$\tilde{V}^2(s_t) = \alpha \log \sum_{a_t \in A} \mu^1(a_t|s_t) \exp\left(\frac{1}{\alpha} r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}^1((s_{t+1})])\right) \tag{27}$$

$$\leq \alpha \log \sum_{a_t \in A} \mu^0(a_t|s_t) \exp\left(\frac{1}{\alpha}(r_{\max} + \gamma r_{\max} + \gamma^2 C)\right) \tag{28}$$

$$= r_{\max} + \gamma r_{\max} + \gamma^2 C\,. \tag{29}$$

Therefore we can conclude that $\forall n > 1\,, \tilde{V}^n((s_t) \leq \frac{1-\gamma^n}{1-\gamma} r_{\max} + \gamma^n C$ and $\tilde{V}^n$ is bounded from above for $\gamma < 1$.

Lemma 3. ***(Contraction)*** *For $\alpha > 0$, there exists a unique fixed point for the dynamic reference Bellman operator $B_{\mu^n} V = V$.*

Proof: Similar proof was presented in [34]:Lemma 15. Starting from two different value functions $\tilde{V}_1^1$ and $\tilde{V}_2^1$, we know that after n updates we have

$$\tilde{V}_1^{n+1}(s_t) = \alpha \log \sum_{a_t \in A} \mu^n(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}_1^n(s_{t+1})])\right)\,. \tag{30}$$

$$\tilde{V}_2^{n+1}(s_t) = \alpha \log \sum_{a_t \in A} \mu^n(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\tilde{V}_2^n(s_{t+1})])\right)\,. \tag{31}$$

Let $\mathcal{V}^n = \|\tilde{V}_1^n - \tilde{V}_2^n\|_\infty = \max_{s\in S} |\tilde{V}_1^n(s) - \tilde{V}_2^n(s)|$ and denote

$$A(s, a; \tilde{V}_1^n) = \frac{1}{\alpha}\left(r(s, a) + \gamma \mathbb{E}_{s'\sim p}[\tilde{V}_1^n(s')]\right)$$

$$\begin{aligned}
\exp A(s, a; \tilde{V}_1^n) &= \exp\left(\frac{1}{\alpha}\left(r(s, a) + \gamma \mathbb{E}_{s'\sim p}[\tilde{V}_1^n(s')]\right)\right) \\
&\leq \exp\left(\frac{1}{\alpha}\left(r(s, a) + \gamma\left(\mathbb{E}_{s'\sim p}[\tilde{V}_2^n(s')] + \mathcal{V}^n\right)\right)\right) \\
&= \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \cdot \exp\left(\frac{1}{\alpha}\left(r(s, a) + \gamma\left(\mathbb{E}_{s'\sim p}[\tilde{V}_2^n(s')]\right)\right)\right) \\
&\leq \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \cdot \exp A(s, a; \tilde{V}_2^n)\,.
\end{aligned} \tag{32}$$

we have for $\forall s \in \mathcal{S}$,

$$\begin{aligned}
\tilde{V}_1^{n+1}(s) - \tilde{V}_2^{n+1}(s) =& \alpha \log\left(\frac{\sum_a \mu^n(a|s) \exp A(s, a; \tilde{V}_1^n)}{\sum_a \mu^n(a|s) \exp A(s, a; \tilde{V}_2^n)}\right) \\
&\leq \alpha \log\left(\frac{\sum_a \mu^n(a|s) \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \cdot \exp A(s, a; \tilde{V}_2^n)}{\sum_a \mu^n(a|s) \exp A(s, a; \tilde{V}_2^n)}\right) \\
&= \alpha \log\left(\exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \frac{\sum_a \mu^n(a|s) \exp A(s, a; \tilde{V}_2^n)}{\sum_a \mu^n(a|s) \exp A(s, a; \tilde{V}_2^n)}\right) \\
&= \gamma \mathcal{V}^n
\end{aligned}$$

Then $\mathcal{V}^{n+1} = \|\tilde{V}_1^{n+1} - \tilde{V}_2^{n+1}\|_\infty \leq \gamma \mathcal{V}^n$. Lemma 2 with the contraction mapping fixed-point theorem [5] ensures the existence and uniqueness of V if $\gamma < 1$.

As the reference policies converge towards an optimal limit, the corresponding state value functions evolve crucially towards stabilization, ensuring the effectiveness of our value iteration approach. The following theorem introduces a proof strategy that establishes the convergence of the state value functions in our dynamic reference policy setting, highlighting the direct influence of evolving reference policies on the optimization process.

Lemma 1. ***(Convergence)*** *Given a sequence of reference policies μ^n that converges to μ^∞, that is,*

$$\forall \epsilon > 0, \exists N \in \mathcal{N} \cup \{0\} : \forall n > N, \max_{a \in \mathcal{A}, s \in \mathcal{S}} |\mu^n(a|s) - \mu^\infty(a|s)| < \epsilon \,, \tag{33}$$

then the state value function V_{π^n/μ^n} defined in equation 6 converges as μ^n converges to μ^∞.

In addition, we define the target policy π^n as

$$\pi^n(a_t|s_t) = \frac{\mu^n(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[V_{\pi^n/\mu^n}(s_{t+1})])\right)}{\sum_{a_t \in A} \mu^n(a_t|s_t) \exp\left(\frac{1}{\alpha}(r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[V_{\pi^n/\mu^n}(s_{t+1})])\right)} \,. \tag{34}$$

π^n converges to its limit policy π^∞.

Proof: The convergence of $\tilde{V}^n$ and π^n is straightforward due to the convergence of the dynamic reference policies μ^n to μ^∞. Given that $\mu^n \to \mu^\infty$, we can use the continuity properties of the state value function and the definition of the target policy to show that $\tilde{V}^n$ and π^n will also converge. Essentially, because the policy μ^n converges, the iterative process defined by the value iteration naturally leads to the convergence of the state value function and the target policy.

Let $\mathcal{V}^n = ||\tilde{V}^n - \tilde{V}^\infty|| = \max_{s \in S} |\tilde{V}^n(s) - \tilde{V}^\infty(s)|$ and let's define $A(s,a;V) = \frac{1}{\alpha}(r(s,a) + \gamma \mathbb{E}_{s'\sim p}[V(s')])$. Then we can show as in the lemma 3 that

$$\begin{aligned}
\exp A(s,a;\tilde{V}^n) &= \exp\left(\frac{1}{\alpha}\left(r(s,a) + \gamma \mathbb{E}_{s'\sim p}[\tilde{V}^n(s')]\right)\right) \\
&\leq \exp\left(\frac{1}{\alpha}\left(r(s,a) + \gamma\left(\mathbb{E}_{s'\sim p}[\tilde{V}^\infty(s')] + \mathcal{V}^n\right)\right)\right) \\
&= \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \times \exp\left(\frac{1}{\alpha}\left(r(s,a) + \gamma\left(\mathbb{E}_{s'\sim p}[\tilde{V}^\infty(s')]\right)\right)\right) \\
&\leq \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \times \exp A(s,a;\tilde{V}^\infty) \,.
\end{aligned} \tag{35}$$

Finally, $\forall s \in S, \tilde{V}^{n+1}(s) - \tilde{V}^\infty(s)$, and by assumption of convergence of the series μ^n there exists n such that

$$\begin{aligned}
\mathcal{V}^{n+1} &= \alpha \log\left(\frac{\sum_a \mu^n(a|s) \exp A(s,a;\tilde{V}^n)}{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)}\right) \\
&\leq \alpha \log\left(\frac{\sum_a (\mu^\infty(a|s) + \epsilon) \exp A(s,a;\tilde{V}^n)}{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)}\right) \\
&= \alpha \log\left(\frac{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^n)}{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)} + \epsilon \frac{\sum_a \exp A(s,a;\tilde{V}^n)}{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)}\right) \\
&\leq \alpha \log\left(\frac{\sum_a \mu^\infty(a|s) \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \exp A(s,a;\tilde{V}^\infty)}{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)} + \epsilon \frac{\sum_a \exp\left(\frac{\gamma}{\alpha}\mathcal{V}_{\mu^n}\right) \exp A(s,a;\tilde{V}^\infty)}{\sum_a \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)}\right) \\
&= \alpha \log\left(\exp\left(\frac{\gamma}{\alpha}\mathcal{V}^n\right) + \epsilon \exp\left(\frac{\gamma}{\alpha}\mathcal{V}^n\right) \times C\right) \\
&= \alpha \log\left(\exp\left(\frac{\gamma}{\alpha}\mathcal{V}^n\right)(1 + \epsilon C)\right) \\
&= \gamma \mathcal{V}^n + \alpha \log(1 + \epsilon C) \\
&\leq \gamma \mathcal{V}^n + \alpha \epsilon \, C \,,
\end{aligned}$$

where $C = \max_{s \in \mathcal{S}} \left[\frac{\sum_{a \in A} \exp A(s,a;\tilde{V}^\infty)}{\sum_{a \in A} \mu^\infty(a|s) \exp A(s,a;\tilde{V}^\infty)}\right]$. The existence of this value is guaranteed by the bounded nature of the value function and therefore $A(s,a;\tilde{V})$ as established in Lemma 2.

Similarly we can show that

$$\forall s \in S, \tilde{V}^\infty(s) - \tilde{V}^{n+1}(s) \leq \gamma \mathcal{V}^n + \alpha \epsilon \, C \,. \tag{36}$$

This indicates that

$$\begin{aligned}\mathcal{V}^{n+1} = \max_{s\in S} |\tilde{V}^{\infty}(s) - \tilde{V}^{n+1}(s)| &\leq \gamma \mathcal{V}^{n} + \alpha\epsilon\, C \\ &\leq \gamma^2 \mathcal{V}^{n-1} + \alpha\epsilon\,\gamma C + \alpha\epsilon\, C \\ &\leq \gamma^{n-N+1}\mathcal{V}^{N} + \frac{1-\gamma^{n-N+1}}{1-\gamma}\alpha\epsilon\, C\,,\end{aligned}$$

which implies that $\lim_{n\to\infty} \mathcal{V}^n = 0$ and $\forall s \in S, \lim_{n\to\infty} \tilde{V}^n(s) = \tilde{V}^{\infty}(s)$. Finally, the convergence of sequence π^n follows from the convergence of $\tilde{V}^n$ and μ^n.

Theorem 2. ***(Dynamic Reference Value Iteration)*** *Given any initial state value function $\tilde{V}^0$, iteratively applying the dynamic reference Bellman operators B_{μ^n}, defined in equation 6, until convergence guarantees that the process reaches the fixed point of the Bellman operator B_{μ^∞}.*

Proof: Lemma 3 establishes the contractive property of the dynamic reference Bellman operator, ensuring that successive applications of B_{μ^∞} on any initial value function V result in a sequence converging to a unique fixed point. Theorem 1 shows that as the sequence of reference policies μ^n converges to μ^∞, the corresponding state value functions $\tilde{V}^n$ converge to $\tilde{V}^\infty$, which is the fixed point of B_{μ^∞}. Therefore, starting with any state value function and applying the dynamic reference Bellman operators until convergence, the process is assured to stabilize at the fixed point of B_{μ^∞}.

9.2.3 Off-Reward Dynamic Reference Policy Iteration

In this section, we present the proofs for the dynamic reference policy iteration algorithm, which serves as the theoretical basis for our dynamic reference actor-critic method. We describe the application of dynamic reference policy evaluation using both the main target policy and the dynamic reference policy, as well as the methodology for directing the improvement of the target policy accurately. We utilize Lemma 5 to illustrate the discrepancies in the value function under the same policy π but with different references. We conclude with a proof of the convergence of the dynamic reference policy iteration algorithm. First we remind you that our state value function is defined as

$$\begin{aligned}V_{\pi/\mu^n}(s) &:= \mathbb{E}_{\substack{s_0=s, a_t\sim\pi(\cdot|s_t)\\ s_{t+1}\sim p(\cdot|s_t,a_t)}}\left[\sum_{t=0}^{\infty}\gamma^t(r(s_t,a_t) - \alpha\log\frac{\pi(a_t|s_t)}{\mu^n(a_t|s_t)}\right] \\ &= \mathbb{E}_{\substack{a_t\sim\pi(\cdot|s_t)\\ s_{t+1}\sim p(\cdot|s_t,a_t)}}\left[Q_{\pi/\mu^n}(s_t,a_t) - \alpha\log(\frac{\pi(a_t|s_t)}{\mu^n(a_t|s_t)})\right].\end{aligned}$$

where

$$Q_{\pi/\mu^n}(s_t,a_t) := r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t)}[V_{\pi/\mu^n}(s_{t+1})]\,, \tag{37}$$

Lemma 4. ***(Dynamic Reference Policy Evaluation)*** *Consider the modified Bellman backup operator $\mathcal{T}$ in equation 37 and a mapping $Q^0_{\pi/\mu^n} : S\times A \to \mathbb{R}$ with $|A| < \infty$ and $\mathbb{E}_{(a_t,s_t)\sim\rho_\pi}\left[\log\left(\frac{\pi(a_t|s_t)}{\mu^n(a_t|s_t)}\right)\right] < C$, and define $Q^{k+1}_{\pi/\mu^n} = \mathcal{T}Q^k_{\pi/\mu^n}$. Then the sequence Q^k_{π/μ^n} will converge to the dynamic reference Q-function Q^*_{π/μ^n} of π as $k\to\infty$.*

Proof: The proof follows directly from established results in the literature, specifically leveraging the framework and techniques discussed in [17]. Define the regularized-augmented reward, denoted by $r_{\pi,\mu,t}(s_t,a_t)$, as

$$r_{\pi,\mu^n,t}(s_t,a_t) = r(s_t,a_t) - \gamma\alpha\,\mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi(\cdot|s_{t+1})}\left[\log\left(\frac{\pi(a_{t+1}|s_{t+1})}{\mu^n(a_{t+1}|s_{t+1})}\right)\right]. \tag{38}$$

Then we can rewrite the modified Bellman equation 37 into the standard Bellman equation from the true Q^π as follows

$$\mathcal{T}Q_{\pi/\mu^n}(s_t,a_t) = r_{\pi,\mu^n,t}(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi}[Q_{\pi/\mu^n}(s_{t+1},a_{t+1})]\,. \tag{39}$$

Under the assumption of a bounded action space and $\mathbb{E}_{(a_t,s_t)\sim\rho_\pi}\left[\log\left(\frac{\pi(a_t|s_t)}{\mu^n(a_t|s_t)}\right)\right] < C$, the reward $r_{\pi,\mu,t}$ is bounded and the convergence is guaranteed as the usual policy evaluation.

Lemma 5. *Given a reference policy μ^n from a sequence of reference policies μ that converges to its optimal policy μ^∞ and satisfies $\forall(a,s) \in \mathcal{A}\times\mathcal{S}, \forall k \in \mathbb{N}\cup\{\infty\}, \mu^k(a|s) > 0$ and the condition in* (7)*, then for every $\epsilon > 0$, there exists $N \in \mathbb{N}$ such that for all $n \geq N$, we have $\|Q_{\pi/\mu^n} - Q_{\pi/\mu^\infty}\|_\infty \leq \epsilon$.*

Proof: Let's denote $\Delta\mu = \max_{\forall a,s\in\mathcal{A}\times\mathcal{S}} |\mu^\infty(a|s) - \mu^n(a|s)|$. From Equation 7, we know that for any $\delta > 0$, there exists an $N \in \mathcal{N}$ such that for all $n > N$, we have $\Delta\mu < \delta$. Let $N_0 \in \mathcal{N}$ such that $0 < \delta < (1-\gamma\alpha)\cdot\epsilon\cdot\min_{\forall a,s\in\mathcal{A}\times\mathcal{S}}(\mu^\infty(a|s))$ and notice that $\forall s,a$ we have

$$
\begin{aligned}
&Q_{\pi/\mu^n}(s_t,a_t)\\
&= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V_{\pi/\mu^n}(s_{t+1})\right]\\
&= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi}\left[Q_{\pi/\mu^n}(s_{t+1},a_{t+1}) - \alpha\log\left(\frac{\pi(a_{t+1}|s_{t+1})}{\mu^n(a_{t+1}|s_{t+1})}\right)\right]\right]\\
&\leq r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi}\left[Q_{\pi/\mu^n}(s_{t+1},a_{t+1}) - \alpha\log\left(\frac{\pi(a_{t+1}|s_{t+1})}{\mu^\infty(a_{t+1}|s_{t+1}) + \Delta\mu}\right)\right]\right]\\
&= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi}\left[Q_{\pi/\mu^n}(s_{t+1},a_{t+1}) - \alpha\log\left(\frac{\pi(a_{t+1}|s_{t+1})}{\mu^\infty(a_{t+1}|s_{t+1})\left(1+\frac{\Delta\mu}{\mu^\infty(a_{t+1}|s_{t+1})}\right)}\right)\right]\right]\\
&= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi}\left[Q_{\pi/\mu^n}(s_{t+1},a_{t+1}) - \alpha\log\left(\frac{\pi(a_{t+1}|s_{t+1})}{\mu^\infty(a_{t+1}|s_{t+1})}\right)\right]\right]\\
&+ \gamma\alpha\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi}\left[\log\left(1+\frac{\Delta\mu}{\mu^\infty(a_{t+1}|s_{t+1})}\right)\right]\right]\\
&= \ldots\\
&= Q_{\pi/\mu^\infty}(s_t,a_t) + \sum_l \gamma^l\alpha^l\mathbb{E}_{s_{l+1}\sim p}\left[\mathbb{E}_{a_{l+1}\sim\pi}\left[\log\left(1+\frac{\Delta\mu}{\mu^\infty(a_{l+1}|s_{l+1})}\right)\right]\right]\\
&\leq Q_{\pi/\mu^\infty}(s_t,a_t) + \sum_l \gamma^l\alpha^l\mathbb{E}_{s_{l+1}\sim p}\left[\mathbb{E}_{a_{l+1}\sim\pi}\left[\log\left(1+(1-\gamma\alpha)\epsilon\right)\right]\right]\\
&\leq Q_{\pi/\mu^\infty}(s_t,a_t) + (1-\gamma\alpha)\sum_l \gamma^l\alpha^l\mathbb{E}_{s_{l+1}\sim p}\left[\mathbb{E}_{a_{l+1}\sim\pi}\left[\epsilon\right]\right]\\
&= Q_{\pi/\mu^\infty}(s_t,a_t) + (1-\gamma\alpha)\sum_l \gamma^l\alpha^l\epsilon\\
&= Q_{\pi/\mu^\infty}(s_t,a_t) + \epsilon\,.
\end{aligned}
$$

We have obtained that $\forall a_t, s_t \in \mathcal{A}\times\mathcal{S}, Q_{\pi/\mu^n}(s_t,a_t) - Q_{\pi/\mu^\infty}(s_t,a_t) \leq \epsilon$. By following the exact same steps we find that also $\forall a_t, s_t \in \mathcal{A}\times\mathcal{S}, Q_{\pi/\mu^\infty}(s_t,a_t) - Q_{\pi/\mu^n}(s_t,a_t) \leq \epsilon$. Therefore, we conclude that $\|Q_{\pi/\mu^n} - Q_{\pi/\mu^\infty}\| \leq \epsilon$.

Lemma 6. ***(Dynamic Reference Policy Improvement):***

Let $\pi_{old} \in \Pi$ and let π_{new} be the optimizer of the minimization problem defined in Equation 11. Then $Q_{\pi_{new}/\mu_{new}}(s_t,a_t) \geq Q_{\pi_{old}/\mu_{new}}(s_t,a_t)$ for all $(s_t,a_t) \in S\times A$ with $|A| < \infty$.

Proof: A similar proof has been introduced in [17] and [46]. Here, we present a variation with additional details. Starting with the definition of the KL-divergence, we have

$$J_{\pi_{\text{old}}}(\pi(\cdot|s_t)) = D_{\text{KL}}\left(\pi(\cdot|s_t)\middle|\middle|\frac{\mu_{\text{new}}(\cdot|s_t)\exp\left(\frac{1}{\alpha}Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,\cdot)\right)}{Z_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t)}\right) \tag{40}$$

$$= \mathbb{E}_{a\sim\pi(\cdot|s_t)}\left[\log\left(\frac{\pi(a|s_t)}{\frac{\mu_{\text{new}}(a|s_t)\exp\left(\frac{1}{\alpha}Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,a)\right)}{Z_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t)}}\right)\right]. \tag{41}$$

Rearranging the terms inside the expectation

$$J_{\pi_{\text{old}}}(\pi(\cdot|s_t)) = \mathbb{E}_{a\sim\pi(\cdot|s_t)}\left[\log(\pi(a|s_t)) - \log\left(\mu_{\text{new}}(a|s_t)\right) - \frac{1}{\alpha}Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,a) + \log(Z_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t))\right]. \tag{42}$$

Now, let's denote the minimizing policy as π_{new} equals

$$\pi_{\text{new}} = \arg\min_{\pi'\in\Pi} J_{\pi_{\text{old}}}(\pi'(\cdot|s_t)). \tag{43}$$

The term $\log(Z_{\pi_{\text{old}}/\mu_{\text{new}}})$ is a function only of the states, and therefore, will not have an impact on the argmin operation. Following the definition we have

$$\mathbb{E}_{a\sim\pi_{\text{new}}(\cdot|s_t)}\left[Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,a) - \alpha\log\left(\frac{\pi_{\text{new}}(a|s_t)}{\mu_{\text{new}}(a|s_t)}\right)\right] \geq \mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s_t)}\left[Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,a) - \alpha\log\left(\frac{\pi_{\text{old}}(a|s_t)}{\mu_{\text{new}}(a|s_t)}\right)\right] \tag{44}$$

Repeatedly expanded $Q_{\pi_{\text{old}}/\mu_{\text{new}}}$ on the RHS by applying the Bellman equation and the bound in Equation 44 we have

$$\begin{aligned}
Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_t,a_t) &= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V_{\pi_{\text{old}}/\mu_{\text{new}}}(s_{t+1})\right] \\
&= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi_{\text{old}}}\left[Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_{t+1},a_{t+1}) - \alpha\log\left(\frac{\pi_{\text{old}}(a_{t+1}|s_{t+1})}{\mu_{\text{new}}(a_{t+1}|s_{t+1})}\right)\right]\right] \\
&\leq r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi_{\text{new}}}\left[Q_{\pi_{\text{old}}/\mu_{\text{new}}}(s_{t+1},a_{t+1}) - \alpha\log\left(\frac{\pi_{\text{new}}(a_{t+1}|s_{t+1})}{\mu_{\text{new}}(a_{t+1}|s_{t+1})}\right)\right]\right] \\
&\leq ... \\
&\leq Q_{\pi_{\text{new}},\mu_{\text{new}}}(s_t,a_t).
\end{aligned}$$

Theorem 3. *Given a sequence of reference polices μ^n convergent to μ^∞. Repeated application of dynamic reference policy evaluation and dynamic reference policy improvement from any $\pi\in\Pi$ converges to a policy π^* such that $Q_{\pi^*/\mu^\infty}(s_t,a_t) \geq Q_{\pi/\mu^\infty}(s_t,a_t)$ for all $\pi\in\Pi$ and $(s_t,a_t)\in S\times A$, assuming $|A|<\infty$ and $\mathbb{E}_{(a_t,s_t)\sim\rho_\pi}\left[\log\left(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}\right)\right] < C$ and the policy sequence μ^n satisfies the condition in equation 7.*

Proof: The proof of this theorem relies on Lemma 5 and the vanilla policy iteration method [46, 17] and lemma 6 and lemma 4. Our dynamic reference policy iteration algorithm starts with an initial random policy π_0 and iteratively improves it. The process can be described as follows

$$\pi_0 \xrightarrow[\mu_1]{\text{E}} Q_{\pi_0/\mu_1} \xrightarrow[\mu_1]{\text{I}} \pi_1 \xrightarrow[\mu_2]{\text{E}} Q_{\pi_1/\mu_2} \xrightarrow[\mu_2]{\text{I}} \pi_2 \xrightarrow[\mu_3]{\text{E}} \dots \xrightarrow[\mu^\infty]{\text{I}} \pi^\infty \xrightarrow[\mu_\infty]{\text{E}} Q_{\pi^\infty/\mu^\infty}. \tag{45}$$

In this sequence, E and I denote the evaluation and improvement steps, respectively, for the dynamic reference policy. We evaluate Q-values based on the current target policy and the reference policy μ, then improve the target policy based on these Q-values.

We know from the properties of vanilla policy iteration that for a fixed μ^∞, the algorithm converges to the optimal $Q_{\pi^\infty/\mu^\infty}$ [17]. This implies that for any $\epsilon>0$, there exists $N_0\in\mathcal{N}$ such that for all $n>N_0$,

$$\|Q_{\pi^\infty/\mu^\infty} - Q_{\pi^n/\mu^\infty}\|_\infty < \epsilon.$$

From Lemma 5, it follows that there exists $N_1 \in \mathcal{N}$ such that for all $n > N_1$,

$$\|Q_{\pi^n/\mu^\infty} - Q_{\pi^n/\mu^n}\|_\infty < \epsilon.$$

Combining these results, for $n > \max\{N_0, N_1\}$, we obtain:

$$\|Q_{\pi^\infty/\mu^\infty} - Q_{\pi^n/\mu^n}\|_\infty \leq \|Q_{\pi^\infty/\mu^\infty} - Q_{\pi^n/\mu^\infty}\|_\infty + \|Q_{\pi^n/\mu^\infty} - Q_{\pi^n/\mu^n}\|_\infty < 2\epsilon.$$

Finally, from the properties of the vanilla policy iteration, we know that $Q_{\pi^\infty/\mu^\infty}(s_t, a_t) \geq Q_{\pi/\mu^\infty}(s_t, a_t)$ for all $\pi \in \Pi$ and $(s_t, a_t) \in S \times A$. This establishes that the dynamic reference policy iteration converges, proving the theorem.

9.3 Implementation Details

9.3.1 MOP reference

The Maximum Occupancy Principle (MOP) [39, 32] is an effective exploration strategy because it emphasizes intrinsic motivation to maximize future action-state paths rather than relying on extrinsic rewards. This approach inherently promotes behavioral variability, preventing agents from getting stuck in local optima and encouraging the exploration of diverse and novel states. By eliminating the need for arbitrary and task-dependent reward functions, MOP simplifies the exploration process and adapts naturally to various environments. Despite being off-reward, MOP agents exhibit goal-directed behaviors, making them versatile and adaptive.

The implementation of MOP in our scenario is straightforward. We do not account for state transition entropy, allowing us to implement it with using SAC [17] with an alpha parameter set to 1 and no rewards, $r(s_t, a_t) = 0$ for all $s_t, a_t \in \mathcal{A} \times \mathcal{S}$ throughout the training process.

Algorithm 1 MOP Reference

1: Initialize policy $\mu(a|s)$.
2: Initialize SAC Agent with fixed $\alpha = 1$.
3: **function** UPDATEREFERENCEPOLICY($\mathcal{D}$)
4: Update policy $\mu(a_t|s_t)$ using SAC [17] with reward $R(s_t, a_t) = 0$
5: **end function**

9.3.2 Lap Reference

Laplacian eigenfunctions provide an effective method for learning state representations in RL. This technique uses the eigen-decomposition of the graph Laplacian to find the smallest eigenvectors, which capture the geometry of a weighted graph formed by state transitions [53, 28, 24]. The main objective is to create robust state representations by capturing how states connect and transition under a given policy.

As shown in section 5 from the main body, our goal is to identify the first d eigenfunctions $f_1, \ldots, f_d$ associated with the smallest d eigenvalues of L. The mapping $\phi : S \to \mathbb{R}^d$ defined by $\phi(u) = [f_1(u), \ldots, f_d(u)]$ creates an embedding or representation of the space S.

The eigenfunctions $f_1, \ldots, f_d$ of the graph Laplacian possess several key properties:

- **Orthogonality:** The eigenfunctions are orthogonal with respect to the stationary distribution ρ, i.e., $\mathbb{E}_{u\sim\rho}[f_i(u)f_j(u)] = \delta_{ij}$, where δ_{ij} is the Kronecker delta.
- **Eigenvalues:** Each eigenfunction corresponds to an eigenvalue λ_i of the graph Laplacian, with $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_d$.
- **Dimensionality:** The eigenfunctions collectively define a d-dimensional embedding of the state space, encapsulating the underlying geometric and transition dynamics.

The paper [53] suggests a method to train the functions f_i, we minimize a graph drawing objective [51, 23] using stochastic gradient descent:

$$G(f_1, \ldots, f_d) = \frac{1}{2} \sum_{s_t\sim\rho, s_{t+1}\sim P^\pi(\cdot|s_t)} \left[\sum_{k=1}^{d} (f_k(s_t) - f_k(s_{t+1}))^2 \right],$$

where $P^\pi(s_{t+1} \mid s_t)$ is the transition distributions and ρ stationary state distribution of P^π.

As discussed before we want to ensure orthonormality of the functions f_i which impose this constrain $\sum_{j,k} \left(\mathbb{E}_{s\sim\rho}\left[f_j(s)f_k(s)\right] - \delta_{jk}\right)^2$, where δ_{jk} is the Kronecker delta. The orthonormality constraint is relaxed to a soft constraint and incorporated as a penalty in the objective (see [53]):

$$\tilde{G}(f_1,\ldots,f_d) = G(f_1,\ldots,f_d) + \lambda\mathbb{E}_{s\sim\rho,s'\sim\rho}\left(\left(\phi(s)^\top\phi(s')\right)^2 - \|\phi(s)\|_2^2 - \|\phi(s')\|_2^2 + d\right),$$

During training, we need to sample states s_t and s_{t+1} from P^π. This can be done using the current target policy. However, since we are restricted to the off-policy paradigm, we sample s_t and s_{t+1} from transitions (s_t, a_t, r_t, s_{t+1}) stored in the replay buffer. Additionally, to compute the constraint, we randomly sample states from the replay buffer. To ensure this approximation does not negatively impact representation learning, one can reduce the replay buffer size. Nevertheless, we found that even with a large replay buffer, the representation method still provides valuable information for the target policy to explore. We selected $d = 4$ to minimize computational resources while training additional reference policy concurrently with the target policy. The dynamic reference policy and embedding function are trained simultaneously.

Our off-reward dynamic reference policy maximizes $\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)$, where the off-reward $R(s_t, a_t)$ is defined as

$$R(s_t, a_t) = \frac{||\phi(s_{t+1}) - \phi(s_t)||_2^2}{||\phi(s_{t+1})||_2^2 + ||\phi(s_t)||_2^2}.$$

This reward function, as stated in the main body 5, encourages the agent to transition to states furthest from the current one, leading to improved exploration.

Algorithm 2 Lap reference

1: Initialize policy $\mu(a|s)$, representation function $\phi(s)$, Lagrange multiplier λ. Target embedding size d, and a small relaxation constant ϵ.
2: Initialize SAC Agent with fixed $\alpha = 0.2$.
3: **function** UPDATEREFERENCEPOLICY($\mathcal{D}$)
4: Compute The Objective:

$$Obj = \mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{D}}\left[\left(\phi(s') - \phi(s)\right)^2\right]$$

5: Compute The Relaxed Constraint:

$$Cstn = \mathbb{E}_{(s,s')\sim\mathcal{D}}\left[\left(\phi(s)^\top\phi(s')\right)^2 - \|\phi(s)\|_2^2 - \|\phi(s')\|_2^2 + d - \epsilon\right]$$

6: Update $\phi(s)$ using gradient descent:

$$\nabla_\phi[Obj + \lambda \cdot Cstn]$$

7: Update λ using gradient ascent:

$$\nabla_\lambda[\lambda \cdot Cstn]$$

8: Compute reward

$$R(s_t, a_t) = \frac{\|\phi(s_{t+1}) - \phi(s_t)\|_2^2}{\|\phi(s_{t+1})\|_2^2 + \|\phi(s_t)\|_2^2}$$

9: Update policy $\mu(a_t|s_t)$ using SAC [17] with reward $R(s_t, a_t)$
10: **end function**

9.3.3 Dynamically Tuning the Similarity Weight

In the previous section, we developed an off-policy learning algorithm that seeks to maximize cumulative rewards while keeping the new policy similar to a reference policy. The level of similarity is controlled by a temperature parameter α. Unfortunately, determining the optimal temperature is not straightforward, and it requires fine-tuning for each specific task. Rather than having users manually set this temperature, we can redefine our learning objective by treating similarity as a constraint. This removes the need to find the perfect temperature and instead ensures that the average

similarity between the two policies stays below a predefined hyperparameter $\mathcal{C}$. This approach allows the target policy to maintain some resemblance to the reference policy while providing room for more deterministic behavior, helping to distinguish between desirable and undesirable actions.

Given reference polices μ^n and for $t \in \{0, ..., T\}$, the optimization problem is formally stated as

$$\text{maximize} \quad E_{\rho_\pi}\left[\sum_{t=0}^{\infty} r(s_t, a_t)\right] \tag{46}$$

$$\text{subject to} \quad E_{(s_t,a_t)\sim\rho_\pi}\left[\log \frac{\pi(a_t|s_t)}{\mu^n(a_t|s_t)}\right] \leq \mathcal{C}\,. \tag{47}$$

Adopting a similar procedure as described in [17], we can change our constrained maximization problem for the last time step T to the dual problem as

$$\max_{\pi_T} E_{(s_T,a_T)\sim\rho_{\pi_T}}\left[r(s_T, a_T)\right] = \min_{\alpha_T \geq 0} \max_{\pi} \underbrace{E_{(s_T,a_T)\sim\rho_{\pi_t}}\left[r(s_T, a_T) - \alpha_T \log \frac{\pi_T(a_T|s_T)}{\mu^n(a_T|s_T)} + \alpha_T \mathcal{C}\right]}_{L(\pi_T,\alpha_T)}\,. \tag{48}$$

We could compute the optimal π_T and α_T iteratively. First given the current α^T, get the best policy π_T^* that maximizes $L(\pi_T^*, \alpha_T)$. Then plug in π_T^* and compute α_T^* that minimizes $L(\pi_T^*, \alpha_T)$.

$$\pi_T^* = \underset{\pi}{argmax}\; E_{(s_t,a_t)\sim\rho_\pi}\left[r(s_T, a_T) - \alpha_T \log \frac{\pi_T(a_T|s_T)}{\mu^n(a_T|s_T)} + \alpha_T \mathcal{C}\right]\,,$$

$$\alpha_T^* = \underset{\alpha_T \geq 0}{argmin}\; E_{(s_t,a_t)\sim\rho_{\pi_T^*}}\left[-\alpha_T \log \frac{\pi_T^*(a_T|s_T)}{\mu^n(a_T|s_T)} + \alpha_T \mathcal{C}\right]\,.$$

To find α_t for $t \leq T$, recall that our Q-function is defined as following

$$Q_t^*(s_t, a_t; \pi_{t+1:T}, \mu^n, \alpha_{t+1:T}^*) = r(s_t, a_t) + \mathbb{E}_{\rho_{\pi_{t+1}^*}}\left[Q_{t+1}^*(s_{t+1}, a_{t+1}) - \alpha_{t+1}^* \log \frac{\pi_{t+1}^*(a_{t+1}|s_{t+1})}{\mu^n(a_{t+1}|s_{t+1})}\right]\,, \tag{49}$$

with $Q_T^*(s_T, a_T) = \mathbb{E}\left[r(s_T, a_T)\right]$. Notice that

$$\begin{aligned} Q_{T-1}^*(s_{T-1}, a_{T-1}) &= r(s_{T-1}, a_{T-1}) + \mathbb{E}_{\rho_{\pi_T^*}}\left[r(s_T, a_T) - \alpha_T^* \log \frac{\pi_T^*(a_T|s_T)}{\mu^n(a_T|s_T)}\right] \\ &= r(s_{T-1}, a_{T-1}) + \max_{\pi_T} \mathbb{E}\left[r(s_T, a_T)\right] - \alpha_T^* \mathcal{C}\,. \end{aligned}$$

Where the second equality comes from equation 48. We can find our α_{T-1} using the Kullback–Leibler divergence constraint and the dual problem as the following

$$\max_{\pi_{T-1}} \mathbb{E}\left[r(s_{T-1}, a_{T-1}) + \max_{\pi_T} \mathbb{E}\left[r(s_T, a_T)\right]\right] \tag{50}$$

$$= \max_{\pi_{T-1}} \left[Q_{T-1}^*(s_{T-1}, a_{T-1}) + \alpha_T^* \mathcal{C}\right] \tag{51}$$

$$= \min_{\alpha_{T-1} \geq 0} \max_{\pi_{T-1}} \left(\mathbb{E}\left[Q_{T-1}^*(s_{T-1}, a_{T-1}) - \alpha_{T-1} \log \frac{\pi_{T-1}(a_{T-1}|s_{T-1})}{\mu^n(a_{T-1}|s_{T-1})} + \alpha_{T-1}^* \mathcal{C}\right]\right) + \alpha_T^* \mathcal{C}\,. \tag{52}$$

In this manner, we can iteratively optimize our objective by proceeding backwards through time. Once we have computed Q_t^* and π_t^*, we can subsequently determine the optimal value of the dual variable, α_t^* as

$$\underset{\alpha_t > 0}{argmin}\; E_{(s_t,a_t)\sim\rho_\pi^*}\left[-\alpha_t \log \frac{\pi_t^*(a_t|s_t)}{\mu^n(a_t|s_t)} + \alpha_t \mathcal{C}\right]\,. \tag{53}$$

9.3.4 Off-Reward Dynamic Reference Actor Critic

While the previous sections establish the theoretical basis and convergence properties of our proposed dynamic reference policy iteration, it becomes imperative to transition these theoretical constructs into practical, feasible solutions for real-world applications. This need arises from the inherent complexities and computational demands encountered in expansive continuous domains, where exact policy iteration may be impractical or impossible. Therefore, the introduction of function approximators, as discussed in this section, serves not only as a bridge between theory and application but also emphasises the essential role of approximation techniques in deploying robust RL algorithms in practical settings. Consequently, function approximators are employed for both the Q-function and the policy. Rather than achieving convergence through evaluation and improvement, we transition towards optimizing all networks via stochastic gradient descent. Let's denote the parameterized Q-function as $Q_\theta(s_t, a_t)$, target policy as $\pi_\phi(a_t|s_t)$ and reference policy $\mu_\psi(a_t|s_t)$ with θ, ϕ and ψ representing their respective parameters. The Q-function will be realized through feed-forward neural networks, while the target and the reference policy is represented as a Gaussian—its mean and covariance guided by neural networks. In the subsequent section, we will describe the update procedures for these parameter vectors.

To optimize the Q-function, parameters can be refined by minimizing the Bellman difference

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\theta}}(s_{t+1})]\right)\right)^2\right]. \tag{54}$$

Where the value function, V, is indirectly determined by the Q-function's parameters as per Equation (10) and $\mathcal{D}$ is the replay buffer (pool) to store the collection of transition tuples (Algorithm 3).

In this schema, the update employs a target Q-function, defined by parameters $\bar{\theta}$. These parameters originate from the primary parameters, albeit with a delay [31]. This method has been demonstrated to enhance stability throughout the training process.

The parameters of the policy are refined by minimizing the expected KL-divergence as denoted in Equation (11), with a factor of α, while omitting the constant log-partition function, as it remains unchanged during the optimization process and thus does not influence the outcome.

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha\log\left(\frac{\pi_\phi(a_t|s_t)}{\mu_\psi(a_t|s_t)}\right) - Q_\theta(s_t,a_t)\right]. \tag{55}$$

Since the Q-function function is symbolized by a neural network and is differentiable, we opt to employ the reparameterization trick, similar to [17, 22, 40]. The reparameterization trick, by parameterizing the stochasticity in the policy through a fixed noise distribution and a deterministic function, allows for direct backpropagation through the stochastic nodes of the network. This results in gradients that exhibit lower variance compared to those obtained through a score function estimator, hence facilitating more stable and efficient learning [29]. Consequently, the policy undergoes reparameterization through a neural network transformation given by $a_t = f_\phi(\varepsilon_t; s_t)$ where ε_t signifies a noise vector sourced from a consistent distribution, possibly a standard normal distribution. Now we write our objective $J_\pi(\phi)$ as

$$a_t = f_\phi(\varepsilon_t; s_t) = \tanh(\mu_\phi(s_t) + \sigma_\phi(s_t)\odot\varepsilon_t), \quad \varepsilon_t \sim \mathcal{N}(0, I) \tag{56}$$

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\varepsilon_t\sim N}\left[\alpha\log\frac{\pi_\phi\left(f_\phi(\varepsilon_t;s_t)|s_t\right)}{\mu_\psi\left(f_\phi(\varepsilon_t;s_t)|s_t\right)} - Q_\theta(s_t, f_\phi(\varepsilon_t;s_t))\right]. \tag{57}$$

Finally, similar to [17], we learn α using approximating dual gradient descent [6]. We compute the gradient of α with respect to the objective function $J_\alpha(\alpha)$ defined by

$$J_\alpha(\alpha) = \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\phi}\left[-\alpha\log\frac{\pi_\phi(a_t|s_t)}{\mu_\psi(a_t|s_t)} + \alpha\,\mathcal{C}\right], \tag{58}$$

and then perform gradient descent to optimize α for minimizing this objective. Equation 58is derived from equation 53 where $\mathcal{C}$ is a hyper-parameter that control the similarity between π_ϕ and μ_ψ (see section 9.3.3 in Appendix).

In developing our algorithm, we also employed twin Q-functions, in line with previous works [17, 19, 12], to counteract the positive bias in the policy improvement step—a known factor that

impacts the success of value-based methods [18]. We adapted a similar parameterization strategy for the two Q-functions, characterized by parameters θ_i. These were independently trained to optimize $J_Q(\theta_i)$. Consistent with [12], the lesser value among the Q-functions was selected for the stochastic gradient detailed in Equation 54 and the policy gradient in Equation 55. We use feed-forward neural networks to parameterize both the policy and value functions, each network having two hidden layers with 256 neurons per layer. Check the full algorithm pseudo-code for all details (Algorithm 3). Finally, we employ the method for enforcing action bounds as outlined in [17], using an unbounded Gaussian distribution for actions and applying the tanh function to ensure actions are within a finite interval. This involves transforming the likelihoods accordingly, as detailed in the equations provided.

Algorithm 3 Dynamic Refernece Actor-Critic

Input: initial policy parameters ϕ, Q-function parameters θ_1, θ_2, empty replay buffer D, target update frequency ν
Initialize target value network $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
Initialize reference policy parameters ψ
Initialize entropy coefficient $\alpha \leftarrow 1$, log-alpha $\eta \leftarrow \log(\alpha)$
Obtain initial observation state s_0
for $t = 0, 1, 2, \ldots$ **do**
    Select action $a_t \sim \pi_\phi(\cdot|s_t)$ with a stochastic policy (Equation 56)
    Execute a_t in the environment
    Observe next state s_{t+1}, reward r_t, and done signal d_t
    Store $(s_t, a_t, r_t, s_{t+1}, d_t)$ in replay buffer D
    if t is a multiple of N (for some N) **then**
        for $j = 1, \ldots, M$ (for some M) **do**
            Sample mini-batch $\mathcal{B} = (s, a, r, s', d)$ from $\mathcal{D}$
            $\mu_\psi \leftarrow$ UPDATEREFERENCEPOLICY($\mathcal{D}$)
            Compute targets:

$$y(r, s', d) = r + \gamma(1-d)\left(\min_{i=1,2} Q_{\bar{\theta}_i}(s', \tilde{a}') - \alpha \log\left(\frac{\pi_\phi(\tilde{a}'|s';\eta)}{\mu_\psi(\tilde{a}'|s';\eta)}\right)\right),\ \tilde{a}' \sim \pi_\phi(.|s')$$

            Update Q-functions by one step of gradient descent:

$$\nabla_{\theta_i} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s',d)\in\mathcal{B}} (Q_{\theta_i}(s,a) - y(r,s',d))^2 \qquad \text{for } i \in 1, 2$$

            Update main target policy π_ϕ by one step of gradient ascent:

$$\nabla_\phi \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}} \left(\min_{i=1,2} Q_{\bar{\theta}_i}(s, \tilde{a}) - \alpha \log\left(\frac{\pi_\phi(\tilde{a}|s)}{\mu_\psi(\tilde{a}|s)}\right)\right), \quad \tilde{a} \sim \pi_\phi(.|s)$$

            Update α by one step of gradient descent:

$$\nabla_\alpha \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}} [-\alpha \log \frac{\pi_\phi(\tilde{a}|s)}{\mu_\psi(\tilde{a}|s)} + \alpha C], \quad \tilde{a} \sim \pi_\phi(.|s)$$

        end for
        if t is a multiple of ν **then**
            Update the target networks: $\bar{\theta}_i \leftarrow \theta_i$
        end if
    end if
end for
Output: ϕ, θ_1, θ_2

9.4 Experiments

In the following, we provide a comprehensive description of the setups used for the escape room and MuJoCo experiments. For training each set of 5 agents, we utilized a cluster node equipped

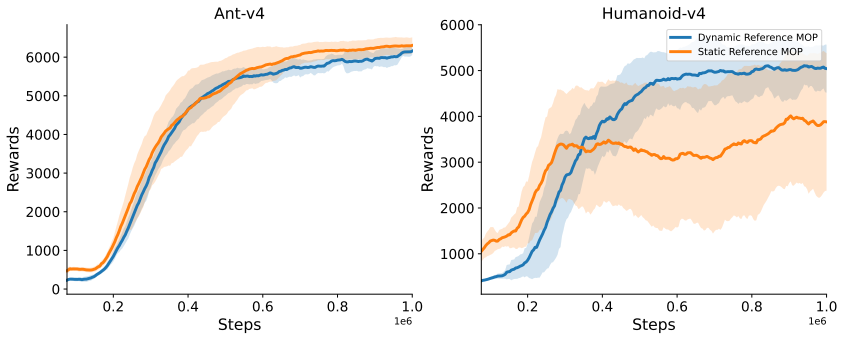

Figure 3: Comparison of dynamic and static reference policies. The graphs display the performance of dynamic (blue) and static (orange) reference policies across two benchmarks: Ant-v4 (left) and Humanoid-v4 (right). The shaded areas represent the confidence intervals for each policy. The results demonstrate the superior adaptability and effectiveness of the dynamic reference policy, particularly in the more complex Humanoid-v4 environment where the static policy's limitations are pronounced.

with a T4 GPU, 16 CPU cores, and 4GB of memory per CPU. The CPUs varied and included types such as Intel Xeon and Broadwell processors. The training time for each agent is documented in the provided data, with each run for 5 agents taking between 20 and 30 hours on average, while baseline training required less time. Our experiments were conducted within a Docker environment based on the `nvidia/cuda:12.0.0-base-ubuntu20.04` image. MuJoCo 2.1.0 was employed for the simulations, with setup details specified in the Dockerfile. The GPU memory usage during training was approximately 8 GB to 10 GB. This setup ensures that all necessary software and dependencies are correctly installed and configured, promoting accurate reproducibility of the experiments. Certain aspects of our algorithm implementation were inspired by the implementation in [1].

9.4.1 Dynamic and Static Reference Policies in Complex Environments

In our study, we conducted experiments to compare dynamic and static reference policies in the Ant-v4 and Humanoid-v4 Mujoco environments. The results, illustrated in (Fig. 3), show that while both types of policies enhance learning in the Ant-v4 environment, the dynamic reference policy demonstrates a consistent advantage by adapting its strategy according to the evolving state of the environment and the main policy's performance. In the more complex Humanoid-v4 environment, the static reference policy plateaued early, underperforming significantly compared to the dynamic policy, which continued to adapt and guide the main policy towards more optimal behaviors. This adaptability of the dynamic reference was important in complex environments, highlighting its utility in preventing premature convergence to suboptimal policies and promoting better exploration and performance outcomes.

9.4.2 Escape room

In our experiments, we employed a simulated environment created using the MuJoCo physics engine, specifically designed to evaluate the exploration capabilities of RL algorithms. The environment features a quadrupedal robotic agent, modeled as an "ant," which is tasked with navigating a bounded arena characterized by a room with opened wall and additional wall forward. The main objective in this scenario is to assess how effectively the agent can explore its surroundings, with the reward mechanism based on the distance between the agent and the center of the room, promoting central exploration. Additionally, the agent has access to its positional information, which is crucial for making informed decisions about movements and pathfinding. This experimental design is intended to challenge the agent's exploratory behaviors and decision-making processes, leveraging the detailed and realistic simulation capabilities of MuJoCo to derive meaningful conclusions about the performance of the learning algorithms. You can find the XML file description of this environment with the provided code.

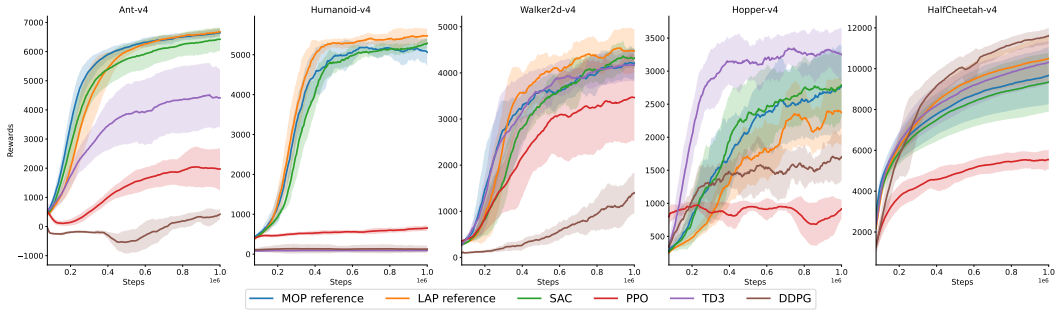

Figure 4: Training curves on continuous control benchmarks indicate that the target policy, when paired with an Exploration-Driven off-reward dynamic reference one, outperforms across several tasks.

9.4.3 Mujoco

For all our MuJoCo benchmark experiments, we followed standard literature regarding the action and state spaces. You can find a detailed information about our set of hyperparameters in Table 3. As in [17], we constrain the actions within a finite interval by applying the hyperbolic tangent (tanh) squashing function to Gaussian-distributed samples. We observed that DDPG and PPO, implemented from RL-Zoo3 [37], exhibited poor results in some Mujoco environments, aligning with trends documented in [17]. Our independently developed SAC implementation also mirrored the performance results reported by the original authors [17].

Figure 4 shows the total average return of evaluation rollouts during training for SAC, DDPG, PPO, and TD3. We train ten separate instances of each algorithm with distinct random seeds, with each carrying out one evaluation rollout every 4000 environment steps. The solid lines represent the mean, while the shaded areas represent the standard deviation of returns over the ten trials. The results indicate that, in general, Lap reference outperforms the baseline approaches on challenging tasks, with poor outcomes in only one out of five tasks. Dynamic Reference methods demonstrate a clear advantage in learning speed, stability and final performance.

Table 3: Hyperparameters

Parameter	Value
Optimizer	Adam [21]
Learning rate	3×10^{-4}
Discount (γ)	0.99
Replay buffer size	10^7
Number of hidden layers (all networks)	2
Number of hidden units per layer	256
Number of samples per minibatch	512
Steps per epoch	4000
Initial random steps	10000
Number of test episodes	10
Nonlinearity	ReLU
Similarity hyper-parameters $\mathcal{C}$	8
Update interval of target networks	500
Lap	
Update interval of target networks	1000
Batch Size	1024
Similarity hyper-parameters $\mathcal{C}$	16
ϵ	10^{-3}
Initial λ	30
SAC α	0.2
MOP	
Update interval of target networks	1000
Batch Size	1024
Similarity hyper-parameters $\mathcal{C}$	8
β	0
SAC α	1.0
SAC entropy target	$-\dim(A)$