Dense and Diverse Goal Coverage in Multi Goal Reinforcement Learning

Sagalpreet Singh
Google Deepmind

Rishi Saket
Google Deepmind

Aravindan Raghuveer
Google Deepmind

Abstract

Reinforcement Learning algorithms are primarily focused on learning a policy that maximizes expected return. As a result, the learned policy can exploit one or few reward sources. However, in many natural situations, it is desirable to learn a policy that induces a dispersed marginal state distribution over rewarding states, while maximizing the expected return which is typically tied to reaching a goal state. This aspect remains relatively unexplored. Existing techniques based on entropy regularization and intrinsic rewards use stochasticity for encouraging exploration to find an optimal policy which may not necessarily lead to dispersed marginal state distribution over rewarding states. Other RL algorithms which match a target distribution assume the latter to be available apriori. This may be infeasible in large scale systems where enumeration of all states is not possible and a state is determined to be a goal state only upon reaching it. We formalize the problem of maximizing the expected return while uniformly visiting the goal states as Multi Goal RL in which an oracle classifier over the state space determines the goal states. We propose a novel algorithm that learns a high-return policy mixture with marginal state distribution dispersed over the set of goal states. Our algorithm is based on optimizing a custom RL reward which is computed - based on the current policy mixture - at each iteration for a set of sampled trajectories. The latter are used via an offline RL algorithm to update the policy mixture. We prove performance guarantees for our algorithm, showing efficient convergence bounds for optimizing a natural objective which captures the expected return as well as the dispersion of the marginal state distribution over the goal states. We design and perform experiments on synthetic MDPs and standard RL environments to evaluate the effectiveness of our algorithm.

1 Introduction

Reinforcement Learning (RL) has been effectively applied to various domains like game playing (Mnih et al., 2013; Silver et al., 2016), robotic control (Gu et al., 2017; Schulman et al., 2015) and aligning pretrained models (Ouyang et al., 2022). The goal is to learn a policy maximizing the expected reward. Many real-world problems fall in the multi-goal settings where the primary objective is to reach an outcome from the subset of desirable outcomes. In such a setting, the reward is a signal of the outcome and not how that specific outcome was arrived at, like the correct placement of a tool by a robotic arm (Andrychowicz et al., 2017), or the discovery of a stable molecular structure (Ghugare et al., 2024), or reinforcement learning on verifiable rewards (Lambert et al., 2024). For each of these examples, there exist multiple desirable outcomes (or goal states), often characterized by a sparse positive scalar reward.

In real world settings, a policy that utilizes only one goal state (or a subset thereof) may discover only a single stable molecular structure (or a subset thereof). Similarly, the navigation policy of a robot is susceptible to failure if a few goal states become unavailable. Thus the desirable strategy is to learn a policy (or a policy mixture) that maximizes the return by exploring a large number of goal states rather than exploiting a small subset of them. Further, in large-scale RL environments the goal states (even if finite) cannot be efficiently enumerated. In particular, the set of goal states is not available apriori, and can only be accessed by sampling trajectories from the environment.

The standard RL objective is defined via reward functions. RL algorithms learn a policy π that maximizes the expected return $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi\right]$ (Sutton

and Barto, 2018). As a result of such formulation, the agent can learn to exploit a single (or a small subset of) reward source. As a result this standard objective is unsuitable for learning a policy which, while maximizing the reward, explores many goal states.

We abstract out the problem of learning a policy mixture that *frequently* visits a *diverse* set of goal states as follows. Consider for ease of exposition, an MDP with finite states $\mathcal{S}$, a subset $\mathcal{S}^+ \subseteq \mathcal{S}$ of goal states, and a state-only dependent reward function $R : \mathcal{S} \to \mathbb{R}$ given by $R(s) := \mathbb{I}\{s \in \mathcal{S}^+\}$. With a discount factor of $\gamma \in (0, 1)$ for a given policy define the discounted marginal state distribution $d[\pi]$ as $d[\pi](s) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \Pr[s_t = s \mid \pi]$ where $s_0, s_1, \ldots$ is the sequence of states visited in a trajectory sampled from π. Thus, the expected return $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$ equals $(1-\gamma)^{-1}\sum_{s\in\mathcal{S}^+} d[\pi](s)$. Note that these quantities can be generalized to a mixture $\overline{\pi}$ of policies by taking $d[\overline{\pi}](s) := \mathbb{E}_{\pi\sim\overline{\pi}}[d[\pi](s)]$, and $J(\overline{\pi}) := \mathbb{E}_{\pi\sim\overline{\pi}}[J(\pi)]$. A policy mixture is a probability distribution over policy class. Following a policy mixture means sampling a policy from the policy mixture and following it for an entire episode.

While $J(\overline{\pi})$ would be the traditional return maximization objective, we need to also capture a measure of the diversity of goal states visited. A natural **objective** is $\mathcal{Z}(\overline{\pi}) := (1-\gamma)J(\overline{\pi}) + \mathcal{I}^{\mathcal{S}^+}(d[\overline{\pi}]) = \sum_{s\in\mathcal{S}^+}(d[\overline{\pi}](s) - d[\overline{\pi}](s)^2/2)$ where $\mathcal{I}^X(p) = -(1/2)\sum_{x\in X} p(x)^2$ is a measure of diversity based on Gini criterion (see Breiman et al. (2017)). In other words, the maximization objective is the sum of the traditional reward and the diversity of the marginal distribution on the goal states. It is easy to see that a policy (assuming it exists) which visits only the goal states and whose discounted marginal state distribution is supported uniformly (and only) over $\mathcal{S}^+$ maximizes $\mathcal{Z}(\overline{\pi})$ with a value of $1 - 1/(2|\mathcal{S}^+|)$. Further, uniformly increasing the marginal state distribution probabilities on $\mathcal{S}^+$, as long as their sum is at most 1, increases the objective (see Section 2.1 for details).

Our Contributions. We provide an iterative algorithm with offline RL as a subroutine to construct a policy mixture with provable performance guarantees. At each iteration, the algorithm samples N_T trajectories of horizon H as per the existing policy mixture, computes rewards for the observed states using the current policy mixture, and uses N_{FQI} iterations of Fitted-Q Iteration (FQI) on the sampled data to obtain a new policy which is added to the policy mixture. We take $\mathcal{F}$ to be the class of action-value or Q-value predictors over which FQI optimizes, and the obtained policy is the greedy one w.r.t. the computed Q-value predictor. Let $\overline{\pi}^K$ be the policy mixture obtained after K iterations of the algorithm for a finite state MDP. Then, $|\mathcal{Z}(\overline{\pi}^K) - \text{argmax}_{\overline{\pi}}\mathcal{Z}(\overline{\pi})|$ is, with high probability, bounded by $O(1/K) + O(\gamma^H + \gamma^{N_{\text{FQI}}}) + O(\epsilon_{\text{approx}}) + O\left(\sqrt{(HN_T)^{-1}\left[\log(K\dim_{\mathcal{F}}/\delta) + \dim_{\mathcal{F}}\log(HN_T)\right]}\right) + O\left(\sqrt{(1/N_T)\log(K/\delta_d)}\right)$ where ϵ_{approx} is the inherent Bellman error and $\texttt{dim}_{\mathcal{F}}$ the *pseudo-dimension* of the class of action-value predictors, assuming a lower bounded *concentrability factor*. Section 3 contains the algorithm and Theorem 3.1 formally states the associated performance guarantees. We also extend the algorithm to continuous state-action spaces by using continuous counterpart of FQI (Antos et al., 2007) along with a set of well motivated techniques detailed in Section 3.1 to bridge the gap between theory and practice.

To evaluate the performance of algorithm empirically, we perform 2 sets of experiments, 1) Discrete MDPs to gain clear insights into the benefits of algorithm by comparing the discounted marginal state distribution induced by the learnt policy mixture with those of the existing algorithms, and 2) Experiments on reacher, pusher, ant and half cheetah environments from JaxGCRL with SAC, Pseudo Counts and SMM as the baseline methods.

1.1 Prior Works

We present a short survey of related works categorized into distinct topics. For each topic, the corresponding problem setting is described and distinguished from the one addressed in this study.

Entropy-Regularized Reinforcement Learning. In widely used RL algorithms like Soft Actor-Critic (Haarnoja et al., 2018), the actor aims to maximize expected return while also maximizing the entropy of learnt policy by introducing a temperature parameter which determines the relative importance of entropy vs. the return thereby controlling the stochasticity of the optimal policy. Similarly, Ahmed et al. (2019), Han and Sung (2021) and Abdolmaleki et al. (2018) propose methods that use maximum entropy objectives to encourage exploration. While such formulations do encourage exploration, no direct conclusion can be drawn about the diversity in set the of goal states visited by the policy. Further, recent works like Zhang et al. (2025) have highlighted how maximum entropy can mislead policy optimization by enforcing high entropy in states where one action is vastly superior compared to other actions. As demonstrated by examples in figures 1c and 1d, learning a highly stochastic high return policy may not align with our objective.

Goal Conditioned Reinforcement Learning

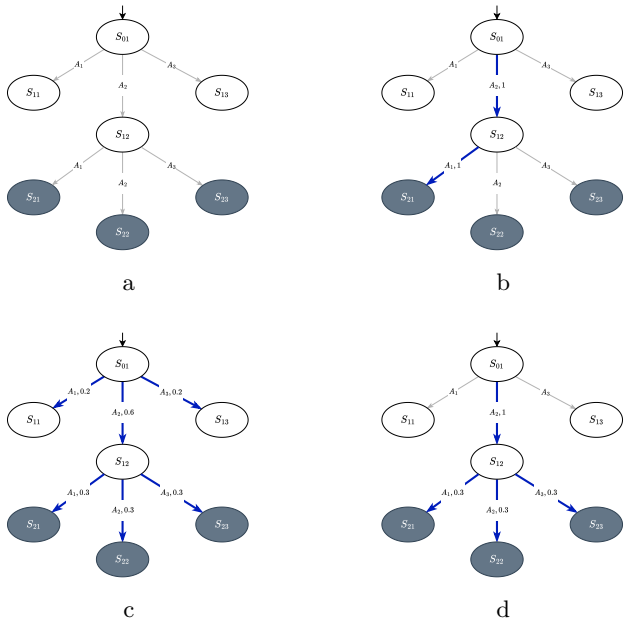


Figure 1: (a) A continuing deterministic MDP with a start state and 3 goal states (highlighted). Assume that all states with no outgoing edges are sink states i.e. any action in these states leads back to the same state. State S_{01} is critical where enforcing high stochasticity can lead to low returns. However, it is perfectly fine to enforce highest entropy on action selection at state S_{12}. (b) An algorithm that encourages exploration may still end up learning a high return policy with low stochasticity which only reaches a subset of goal states. (c) A policy that encourages exploration by promoting stochastic policies will enforce high stochasticity at all states (including critical states) which may lead to policies with suboptimal expected returns. (d) A desirable policy is the one which reaches a diverse set of goal states without compromising on the expected return.

(GCRL). In GCRL (Liu et al., 2022; Eysenbach et al., 2022; Schaul et al., 2015; Andrychowicz et al., 2017), an agent learns to achieve different goals by making decisions depending on the goal. In such settings, observations are augmented with the goal in an episode for the agent to make decisions to achieve that goal. In other words, the agent's actions are conditioned not only on the current observation but also the goal state. GCRL assumes the knowledge of the specific goal state during learning as well as inference, which differentiates it from our multi-goal setup. Some works in GCRL, for e.g. by Zhao et al. (2019), study the problem of maximizing diversity in the set of goal states achieved. However, we omit an extensive survey of GCRL as it is fundamentally different from the problem studied in this work.

State Marginal Matching in Reinforcement Learning. Hazan et al. (2019) study and propose algorithms for learning a policy mixture that optimizes for an objective defined solely as a function of state-visitation frequencies. Lee et al. (2019) propose an algorithm for learning a policy mixture for which the marginal state density matches a target density function. These methods are useful in learning exploratory policies or scenarios where the goal state distribution is known apriori. However, absence of access to the target marginal state distribution is the key differentiating factor due to which such techniques are not applicable to our problem setting.

Intrinsic Reward based Reinforcement Learning. Intrinsic rewards, exploration bonus, curiosity enabled exploration, and diversity in visited states and actions taken have been explored quite extensively in existing literature. Chentanez et al. (2004) proposed learning a hierarchy of reusable skills by providing intrinsic rewards for achieving novel events, rather than for reaching a diverse set of states. Achiam and Sastry (2017) formulated an intrinsic reward based on surprise, defined as the agent's model prediction error, to drive exploration towards regions of the state space understood poorly by the agent. Pathak et al. (2017) generate an intrinsic reward from the error in predicting the next state's features given the current state and action, which incentivizes the agent to explore novel interactions. Eysenbach et al. (2019) learn a set of diverse skills by rewarding policies for visiting states that are different from the states visited by other skills, thereby maximizing state space coverage rather than explicitly learning to reach varied goals. The primary focus of these techniques is to encourage exploration in learning a policy without any specific consideration of induced marginal state distribution.

2 Preliminaries

MDP. Consider a continuing (infinite horizon) MDP, $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, R, \gamma, P, \rho_0\}$ where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, R denotes the reward function, $0 \leq \gamma < 1$ denotes the discount factor, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the environment dynamics, and $\rho_0 \in \Delta(\mathcal{S})$ denotes the start state distribution.

State & Action Spaces. We assume finite state and action spaces for discrete MDPs with cardinalities $|\mathcal{S}|, |\mathcal{A}| \in \mathbb{N}$ respectively. For continuous MDPs, we assume bounded state and action spaces such that $\mathcal{S} = [0,1]^{d_\mathcal{S}}$ and $\mathcal{A} = [0,1]^{d_\mathcal{A}}$ where $d_\mathcal{S}, d_\mathcal{A} \in \mathbb{N}$.

Reward Function. As we study the multi goal setting, the reward is a function of only the state. We refer to desirable outcomes/states as *goal* states and the rest as *non-goal* states. Formally, the state space

is partitioned into sets of goal ($\mathcal{S}^+$) and non-goal ($\mathcal{S}^-$) states such that $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$ and $\mathcal{S}^+ \cap \mathcal{S}^- = \phi$. The reward function can be defined as $R(s) = \begin{cases} 1, & \text{if } s \in \mathcal{S}^+ \\ 0, & \text{if } s \in \mathcal{S}^- \end{cases}$. R is essentially a binary classifier over state space.

Policy & Policy Mixture. A *policy* π function specifies the probability of taking an action from a state. For any $a \in \mathcal{A}$ and $s \in \mathcal{S}$, $\pi(a|s)$ is the probability with which action a is selected in state s when following policy π. We denote the policy class by Π. A policy mixture (denoted by $\overline{\pi}$) is a probability distribution over the policy class. The class of all valid policy mixtures is denoted by $\overline{\Pi} = \Delta(\Pi)$. Following a policy mixture means sampling a policy from the mixture and then following it for an entire episode.

Action-Value predictors. The action-value or Q-value function corresponding to a policy π is the expected sum of discounted reward starting from a state and a specific action i.e., $q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{t} \gamma^t R_t \mid s_0 = s, a_0 = a\right]$. On the other hand, given an action-value function predictor $f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, the corresponding greedy policy π' given by $\pi'(s) \in \text{argmax}_a f(s,a)$ yields at least as much return (starting from any state) as π. We take $\mathcal{F} : \mathcal{S}\times\mathcal{A} \to [0, V_{\max}]$ to be the class of action-value predictors and use them to derive the greedy policy. Note that $V_{\max} = 1/(1-\gamma)$.

To state our results, we shall also use the *pseudo-dimension* of $\mathcal{F}$, denoted by $\texttt{dim}_\mathcal{F}$. For any class of real-valued functions, $\mathcal{H} : \mathcal{X} \to \mathbb{R}$, the pseudo-dimension $\texttt{dim}_\mathcal{H}$ is defined to be the VC-dimension of the class of binary-valued functions $\{t_h : \mathcal{X}\times\mathbb{R} \to \mathbb{R} \mid h \in \mathcal{H}\}$ where $t_h(x,a) = \text{sign}(h(x) - a)$.

Discounted Marginal State Distribution. Denote the probability of agent being in state s at timestep t when following a fixed policy π for a fixed MDP $\mathcal{M}(\mathcal{S}, \mathcal{A}, R, \gamma, \rho_0)$ as $d^t_{\pi,\mathcal{M}}(s)$ such that $d^0_{\pi,\mathcal{M}}(s) = \rho_0$. Note that $d^t_{\pi,\mathcal{M}}$ is a probability mass function (PMF) for discrete state spaces and probability density function (PDF) for continuous state spaces. In this exposition, for simplicity we consider only finite state spaces (see Section 3.1 for MDPs with continuous state and action spaces). Discounted marginal state distribution is defined as $d_{\pi,\mathcal{M}}(s) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t d^t_{\pi,\mathcal{M}}(s)$. For brevity, we use $d[\pi](s)$ instead of $d_{\pi,\mathcal{M}}(s)$. For a policy mixture $\overline{\pi}$, the induced marginal state distribution is denoted by $d[\overline{\pi}]$. Depending upon whether the policy class is continuous or discrete, $d[\overline{\pi}]$ is defined as either $d[\overline{\pi}](s) = \int_\Pi d[\pi](s)\overline{\pi}(\pi)d\pi$ or $d[\overline{\pi}](s) = \sum_{\pi\in\Pi} d[\pi](s)\overline{\pi}(\pi)$. Inspired from Gini criterion (Breiman et al., 2017), we define the diversity of discounted marginal state distribution over set of positive states as $\mathcal{I}^{\mathcal{S}^+}(\overline{\pi}) = -1/2\sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s)^2$. In this paper, we use terms discounted marginal state distribution and marginal state distribution interchangeably to mean the former unless explicitly mentioned.

Expected Return. Expected return is the discounted sum of rewards received by the agent when following policy π for a fixed MDP $\mathcal{M}(\mathcal{S}, \mathcal{A}, R, \gamma, \rho_0)$. Denote the expected return by $J^\pi_\mathcal{M}$. For brevity, we use $J(\pi)$ instead of $J^\pi_\mathcal{M}$. For a policy mixture $\overline{\pi}$, the expected return is denoted by $J(\overline{\pi})$ and defined as $J(\overline{\pi}) = \mathbb{E}_{\pi\sim\overline{\pi}}\mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\pi}[\sum_{i=0}^{\infty}\gamma^t r_t] = 1/(1-\gamma)\sum_{s\in\mathcal{S}} R(s)d[\overline{\pi}](s) = 1/(1-\gamma)\sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s)$. We also define $J_\gamma(\overline{\pi}) = (1-\gamma)J(\overline{\pi}) = \sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s)$.

2.1 Optimization Objective

The objective is to learn a policy mixture that maximizes return and induces a marginal state distribution that is well dispersed over goal states (i.e. not concentrated on a few goal states). Due to environment dynamics, maximizing return and inducing well dispersed marginal state distribution over goal states can turn out to be conflicting objectives as demonstrated via examples in Appendix A.

It is worth noting that the objectives optimized by entropic regularization techniques could benefit by increasing policy entropy even if it ends up assigning more weight to non-goal states. To avoid such undesirable outcomes, we adopt a balanced approach and propose a maximization objective $\max_{\overline{\pi}\in\overline{\Pi}} \mathcal{Z}(\overline{\pi})$, where $\mathcal{Z}(\overline{\pi}) = J_\gamma(\overline{\pi}) + \mathcal{I}^{\mathcal{S}^+}(\overline{\pi})$ i.e. the sum of return and diversity of induced marginal state distribution restricted to goal states. Thus, the objective is:

$$\mathcal{Z}(\overline{\pi}) = \sum_{s\in\mathcal{S}^+}\left(d[\overline{\pi}](s) - \frac{1}{2}d[\overline{\pi}](s)^2\right) \tag{1}$$

The above optimization objective captures both our goals: maximizing the return and dispersion of the state distribution over $\mathcal{S}^+$. To see this, let us define $\mu := \sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s)/|\mathcal{S}^+|$ as the average state probability, and $\epsilon(s) := d[\overline{\pi}](s) - \mu$, as the deviation from the average. Then,

$$\begin{aligned}\mathcal{Z}(\overline{\pi}) &= |\mathcal{S}^+|\mu - (1/2)\sum_{s\in\mathcal{S}^+}(\epsilon(s)+\mu)^2 \\ &= |\mathcal{S}^+|\mu - (1/2)|\mathcal{S}^+|\mu^2 - (1/2)\sum_{s\in\mathcal{S}^+}\epsilon(s)^2 \end{aligned} \tag{2}$$

where the second equality follows from $\sum_{s\in\mathcal{S}^+}\epsilon(s) = 0$. Therefore, for a fixed μ, the objective increases when the vector of deviations becomes smaller in its Euclidean norm, and is maximized when the deviations are all zero i.e., for all $s \in \mathcal{S}^+$, $d[\overline{\pi}](s) = \mu$.

Further, define for a fixed set of probabilities, $Z(c) := \sum_{s\in\mathcal{S}^+} \left(cd[\overline{\pi}](s) - \frac{1}{2}c^2 d[\overline{\pi}](s)^2\right)$. Then, $Z'(c) > 0$ for some $c > 1$, when $\sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s) > \sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s)^2$. This is always true when $\sum_{s\in\mathcal{S}^+} d[\overline{\pi}](s) \in (0,1)$ as $d[\overline{\pi}](s) < d[\overline{\pi}](s)^2$ for each non-zero $d[\overline{\pi}](s)$. Thus, uniformly increasing the probabilities in $\mathcal{S}^+$ by some $c > 1$ (as long as the total probability is at most 1) increases the objective.

We denote the optimal policy mixture for this objective by $\overline{\pi}^* \in \text{argmax}_{\overline{\pi}\in\overline{\Pi}}\mathcal{Z}(\overline{\pi})$

Next, we outline key properties of the policy mixture class as well as the objective function. These properties are instrumental in proving convergence guarantees for the proposed algorithm.

Lemma 2.1. *(proved in Appendix B) The following properties hold for $\overline{\Pi}$ and $\mathcal{Z}$:*

1. *$\overline{\Pi}$ is a convex set.*
2. *$\mathcal{Z}$ is a concave function over $\overline{\Pi}$.*
3. *$C_{\mathcal{Z}} < \infty$, where*

$$C_{\mathcal{Z}} = \sup_{\substack{\overline{\pi}_1,\overline{\pi}\in\overline{\Pi} \\ \lambda\in(0,1) \\ \overline{\pi}_2=\overline{\pi}_1+\lambda(\overline{\pi}-\overline{\pi}_1)}} \frac{2}{\lambda^2}\Big[\mathcal{Z}(\overline{\pi}_1) + \langle\overline{\pi}_2-\overline{\pi}_1, \nabla_\pi\mathcal{Z}(\overline{\pi}_1)\rangle - \mathcal{Z}(\overline{\pi}_2)\Big]$$

3 Algorithm

To optimize the objective defined in Equation 1, we use Algorithm 1 which is based on Frank–Wolfe algorithm (Jaggi, 2013). A policy mixture is randomly initialized and updated at each iteration.

The algorithm iteratively updates the policy mixture by adding a new policy and decaying the weights of existing policies in the mixture. The new policy is obtained by optimizing for a reward function that is dependent on the existing policy mixture. The core idea is to encourage visitation of goal states that are less frequently visited over more frequently visited goal states as per the existing policy mixture. This is achieved by assigning a reward on goal states which decreases monotonically with state visitation frequency but is always non-negative. It is also important to note that 0 reward is given for visiting non-goal states. Thus, reward on any goal state is always greater than any other non-goal state. This is a clear distinction from algorithms that provide exploration bonus or bonus aimed at encouraging stochasticity in the policies. Note that the reward function is fixed in each iteration and is derived from the estimate of goal state visitation frequency of the policy mixture obtained from the previous iteration. The particular choice of batch RL is useful as it requires state visitation frequency estimation for only the goal states encountered in the trajectories sampled from the policy mixture and not for all the states.

We analyze the algorithm and provide performance guarantees on policy mixture obtained at Kth iteration. The following lemma bounds the sub-optimality gap $h_k = \mathcal{Z}(\overline{\pi}^*) - \mathcal{Z}(\overline{\pi}_k)$.

Lemma 3.1. *(proved in Appendix C) If performance of the policy $\overline{\mu}_k$ output by the RL algorithm using empirical estimate $\hat{d}$ in k^{th} iteration of Algorithm 1 is ϵ_k sub-optimal with probability $\geq 1-\delta_k$ over the choice of Γ_k, then $h(\overline{\pi}_K) \leq \frac{2C_{\mathcal{Z}}}{K+1} + \frac{2}{K(K+1)}\sum_{k=1}^{K} k\epsilon_k$ holds with probability $\geq 1 - \sum_{k=1}^{K}\delta_k$.*

We choose FQI for Batch RL step in the proposed algorithm. Under assumptions 3.1 and 3.2, we bound the performance of policy which is obtained using FQI to optimize on estimated reward in lemma 3.2. These are standard assumptions used for FQI analysis as in Agarwal et al. (2019) and Le et al. (2019).

Assumption 3.1. (Concentrability) There exists a constant C such that for all policy mixtures $\overline{\pi} \in \overline{\Pi}$ and for policy mixture $\overline{\pi}_k$ at every step k, we have for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$

$$\frac{d[\overline{\pi}(s)]\cdot p_{\overline{\pi}}(a|s)}{d[\overline{\pi}_k(s)]\cdot p_{\overline{\pi}_k}(a|s)} \leq C$$

where $p_{\overline{\pi}}(a|s)$ is the probability of taking action a in state s while following the policy mixture $\overline{\pi}$.

Assumption 3.2. (Inherent Bellman Error) The following error bound holds for all k,

$$\epsilon_{\text{approx}} := \max_{f\in\mathcal{Z}}\min_{f'\in\mathcal{Z}} \|f' - \mathcal{T}f\|^2_{2,\overline{\pi}_k}$$

where $\mathcal{T}$ is the Bellman operator defined as $\mathcal{T}f(s,a) := r(s,a) + \gamma\mathbb{E}_{s'\in P(.|s,a)}\max_{a'\in\mathcal{A}} f(s',a')$.

Lemma 3.2. *(proved in Appendix D) Performance of the policy obtained using FQI with estimated reward is ϵ_d sub-optimal with probability at least $1-(\delta_d+\delta)$ where,*

$$\epsilon_d \leq \frac{\sqrt{C}}{(1-\gamma)^2}\sqrt{4\epsilon^2_{\text{approx}} + \frac{48\cdot 214\cdot V^4_{\max}}{HN_T}\cdot\Psi(\delta)} + \frac{\gamma^{N_{\text{FQI}}}V_{\max}}{1-\gamma} + \frac{1}{(1-\gamma)^2}\left(\gamma^H + \sqrt{\frac{\log(2/\delta_d)}{2N_T}}\right)$$

Algorithm 1 Dense & Diverse Goal Coverage (DDGC)

Input: $\mathcal{M}$ (MDP)
Initialize: $\overline{\pi}_0$ (policy mixture)
Hyper-parameters: N_T (Number of Trajectories), H (Horizon), K (Mixture Size)

for k in $1 \cdots$ K **do**
  $\Gamma_k \sim \mathcal{M}(\overline{\pi}_{k-1})$, $|\Gamma_k| = HN_T$ ▷ Sample data by following $\overline{\pi}$, $\Gamma_k^{(i)} \in \mathcal{S} \times \mathcal{A} \times \{0,1\} \times \mathcal{S} \times [H]$, $[H] := \{1, 2, \cdots, H\}$
  $\hat{d}_k(s) = \frac{(1-\gamma)}{N_T(1-\gamma^H)} \sum_{(s,a,r,s',t) \in \Gamma_k} \gamma^{t-1} \mathbb{I}(s' = s)$ ▷ Normalized discounted state visitation frequency
  $\hat{r}_k(s) = \begin{cases} 1 - \hat{d}_k(s), & s \in \mathcal{S}^+ \\ 0, & s \in \mathcal{S}^- \end{cases}$ ▷ Custom reward
  $\Gamma'_k \leftarrow \{(s, a, \hat{r}_k(s'), s') | (s, a, r, s', t) \in \Gamma_k\}$ ▷ Update batch with custom reward
  $\mu_k \leftarrow \text{RL}(\mathcal{M}, \Gamma'_k)$ ▷ Batch RL
  $\overline{\mu}_k(\pi) = \begin{cases} 1, & \pi = \mu_k \\ 0, & \pi \neq \mu_k \end{cases}$ ▷ Policy mixture with all weight on μ_k
  $\overline{\pi}_k \leftarrow (1 - \lambda_k)\overline{\pi}_{k-1} + (\lambda_k)\overline{\mu}_k$, $\lambda_k = \frac{2}{k+1}$ ▷ Mixture update
end for
return $\overline{\pi}_K$

and,

$$\Psi(\delta) = \log \frac{28e(dim_{\mathcal{F}} + 1)}{\delta} + dim_{\mathcal{F}} \log(640 H N_T V_{\max}^2)$$

Theorem 3.1. *Define the optimality gap as* $h_k = \mathcal{Z}(\overline{\pi}^*) - \mathcal{Z}(\overline{\pi}_k)$ *where* $\overline{\pi}^* \in \text{argmax}_{\overline{\pi} \in \overline{\Pi}} \mathcal{Z}(\overline{\pi})$. *Then with probability greater than or equal to* $1 - (\delta_d + \delta)$,

$$h(\overline{\pi}_K) \leq \frac{\sqrt{C}}{(1-\gamma)^2} \sqrt{4\epsilon_{\text{approx}}^2 + \frac{48 \cdot 214 \cdot V_{\max}^4}{HN_T} \cdot \Psi(\delta/K)} + \frac{\gamma^{N_{\text{FQI}}} V_{\max}}{1-\gamma} + \frac{2C_{\mathcal{Z}}}{K+1} + \frac{1}{(1-\gamma)^2} \left(\gamma^H + \sqrt{\frac{\log(2K/\delta_d)}{2N_T}} \right)$$

Proof. Follows directly from lemmas 3.1 and 3.2. □

3.1 Algorithmic extensions

To bridge the gap between theory and practice, we augment Algorithm 1 with the following set of well-motivated techniques:

1. **Exploratory Sampling.** The underlying theory for Algorithm 1 (FQI in particular) assumes concentrability of behavior policy mixture at each step k i.e. the policy mixture $\overline{\pi}_{k-1}$. As the algorithm begins with a random policy in the mixture, this assumption is justified theoretically. However, the concentrability constant could decay in later iterations. To overcome this, an exploration strategy must be used to sample trajectories at each step k. Note, however that these trajectories are not used for $\hat{d}_k(s)$ computation. These trajectories are only used as batch data in the Batch RL step. This modification in the algorithm, essentially leads to an efficient discovery of new goal states.

2. **Goal Buffer.** Whenever a trajectory leading to a goal state is observed, it is added to the global goal buffer. This prevents forgetting of already discovered goal states. Similar to exploratory sampling, data from goal buffer is only used in the Batch RL step and not $\hat{d}_k(s)$ computation. Note that if a goal state isn't observed when sampling from $\overline{\pi}_{k-1}$ at step k, it likely means that the discounted marginal state distribution on that goal state is low. On such a state s, high reward of 1 is assigned because the estimate $\hat{d}_k(s)$ would be 0 for that state.

3. **Continuous State-Action Spaces.** We also adapt DDGC for continuous state-action spaces by using continuous version of FQI (also called Fitted Actor Critic) proposed by Antos et al. (2007) and using state discretization for $\hat{d}_k$ estimation. We provide detailed description of DDGC for continuous state-action spaces in Appendix G.2.1. We also provide the pseudo codes for continuous version of DDGC and continuous version of FQI as Algorithms 2 and 3 in the appendix, respectively. A formal analysis of the con-

tinuous algorithm can be constructed in a manner that closely parallels the one for the discrete setting. A full exposition is omitted to avoid redundancy.

It is worth noting that the theoretical analysis holds just as well with modifications 1 and 2 above, as both of these modifications just augment the batch data for Batch RL step thereby effectively changing the behavior policy used to collect the batch data.

4 Experiments

To evaluate the performance of algorithm empirically, we perform 2 sets of experiments. Experiments on controlled synthetic environments are provided in Section 4.1. These are discrete MDPs designed to carefully study different aspects of the proposed algorithm and comparison with existing algorithms. Section 4.2 contains experimental results on standard robotics environments. Specifically, we perform experiments on 4 different Brax (Freeman et al., 2021) environments adapted for multi-goal settings by Bortkiewicz et al. (2025).

Figure 2: For the MDP in Figure 1a, we compare the discounted marginal state distribution induced by different algorithms over the 7 states in the MDP. Each bar indicates the induced probability mass over a state. Green bars indicate discounted marginal state probability of different goal states and red bars indicate those of the non-goal states.

4.1 Controlled Synthetic Environments

To empirically validate the importance of objective function, we conduct experiments for tabular settings on MDP defined in Figure 1a. The discounted marginal state distribution (probability mass function) induced by policies learnt via different algorithms can be visualized in Figure 2.

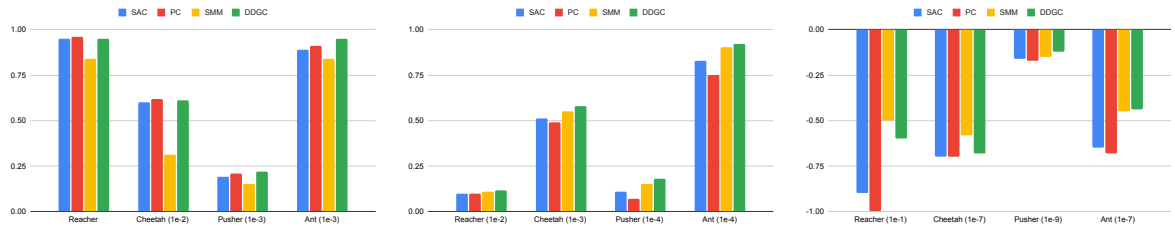
DDGC achieves optimal performance by spreading uniform mass over goal states without compromising on the return (which is proportional to sum of probability mass over goal states). With count based exploration in Q-Learning, once a high return path to goal is found, the same is exploited with no consideration of visiting diverse goal states. A random policy on the other hand puts no consideration on finding a high return policy. The SMM objective encourages the agent to visit all states almost uniformly. More precisely, we use the target density function $d(s) = \exp(r(s))$ as proposed in Lee et al. (2019). Clearly, SMM assigns a positive discounted marginal state distribution to each state, including the non-goal states which is not an optimal choice. While the performance difference between SMM and DDGC might not appear very significant in this small scale setup, once can imagine how this difference could become more prominent for complex environments. We defer the experimental details to Appendix F.

4.2 RL Benchmarks

We also evaluate our algorithm and benchmark against SAC, Pseudo Counts and SMM baselines on reacher, pusher, ant and half cheetah environments. Pseudo Counts and SMM use SAC as the underlying RL algorithm similar to Lee et al. (2019). We use the multi goal setup as proposed in JaxGCRL (Bortkiewicz et al., 2025) with sparse rewards and without goal conditioning. In other words, rewards are binary and the observation doesn't contain any information about the goal. The MDP is multi-goal as the set of goal states comprises all states that satisfy a certain criteria (for example, distance from some point is less than a certain threshold). We defer rest of the experimental details related to the environments and algorithm implementations to Appendix G.

Due to large state space, we cannot visualize the probability distribution induced over all the states unlike the synthetic MDP in Section 4.1. Thus, we empirically compute the return and diversity in visited goal states for each policy. We rollout trajectories using learnt policies (or policy mixtures) with fixed horizon length. To estimate the discounted marginal state distribution, we discretize the state space similar to Lee et al. (2019). We use the sampled trajectories to empirically compute the following statistics:

- **Return.** This is the empirical estimation of discounted return by following a policy (or a mixture thereof).
- **Partial Entropy.** Entropy for a distribution $d(s)$ is defined as $\sum_{s\in\mathcal{S}} -d(s)\log d(s)$. We, however are interested only in the diversity over



a Return b Partial Entropy c Modified Partial Gini Criterion

Figure 3: (a) Plot of discounted return by learnt policies; (b) Plot of partial entropy of discounted marginal state distribution over discretized state space of learnt policies; (c) Plot of modified partial gini criterion of discounted marginal state distribution over discretized state space of learnt policies. In each of the plots, values are averaged over 5 runs and normalized for each environment. Higher value is better for all 3 metrics visualized in the plots above.

goal states. So, the quantity of interest is $\sum_{s\in\mathcal{S}^+} -d(s)\log d(s)$ which we call the partial entropy. We compute partial entropy of discounted marginal state distribution over discretized state space.

- **Modified Partial Gini Criterion.** Gini Criterion for a distribution $d(s)$ is usually defined as $1-\sum_{s\in\mathcal{S}} d(s)^2$. We, however are interested only in the diversity over goal states. So, the quantity of interest is $1-\sum_{s\in\mathcal{S}^+} d(s)^2$ which we call partial gini criterion. For better visualization, we get rid of the constant and compute $-\sum_{s\in\mathcal{S}^+} d(s)^2$ instead and call it modified partial gini criterion.

Return is a direct indicator of how densely the goals are covered by a policy. For diversity, different metrics can be used. We use two such metrics, *Partial Entropy* and *Modified Partial Gini Criterion*. The experimental results and comparison with baselines is visualized in Figure 3.

DDGC consistently matches the best performing algorithm (Pseudo Counts based exploration) in terms of return. At the same time, DDGC also achieves a policy with high diversity in goal states visited as is indicated by Partial Entropy as well as Modified Partial Gini Criterion.

5 Conclusion

Our work studies the problem of learning a policy mixture for multi-goal RL to frequently visit the goal states but also visit a diverse set of goal states. With this motivation, we design an objective function that maximizes the goal state visitation frequency as well the diversity of goal states visited. We optimize this objective function using the Frank-Wolfe method, developing novel algorithm which solves an offline RL subproblem at each step. We provide theoretical guarantees on convergence of the algorithm and we adapt the algorithm for continuous state-action spaces. We also evaluate the algorithm on a small synthetic MDP to highlight the importance of the proposed objective function. We also perform experiments on Mujoco environments which clearly demonstrate the improvement in goal coverage without losing much on the frequency of visitation. As our algorithm relies on multiple calls to RL subroutine (as is also the case with existing works on distribution matching), this leads to an increased computational requirement for training. Future works in this direction can further advance this area with advent of more efficient algorithms for reaching diverse goals.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pages 151–160. PMLR, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proceed-*

ings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 5055–5065, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

András Antos, Csaba Szepesvári, and Rémi Munos. Fitted q-iteration in continuous action-space mdps. *Advances in neural information processing systems*, 20, 2007.

Michał Bortkiewicz, Władek Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating Goal-Conditioned RL Algorithms and Research. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2408.11052.

Leo Breiman, Jerome Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Chapman and Hall/CRC, 2017.

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

Benjamin Eysenbach, Julian Ibarz, Abhishek Gupta, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Raj Ghugare, Santiago Miret, Adriana Hugessen, Mariano Phielipp, and Glen Berseth. Searching for high-value molecules using reinforcement learning and transformers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nqlymMx42E.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

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

Seungyul Han and Youngchul Sung. A max-min entropy framework for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:25732–25745, 2021.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 427–435, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/jaggi13.html.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. TÜlu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124, 2024. URL https://doi.org/10.48550/arXiv.2411.15124.

Hoang Minh Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *Proc. ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3703–3712. PMLR, 2019. URL https://arxiv.org/pdf/1903.08738.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5502–5511. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/770. URL https://doi.org/10.24963/ijcai.2022/770. Survey Track.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Tom Schaul, Dan Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1312–1320. JMLR.org, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Ruipeng Zhang, Ya-Chien Chang, and Sicun Gao. When maximum entropy misleads policy optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=FRFuvBRueA`.

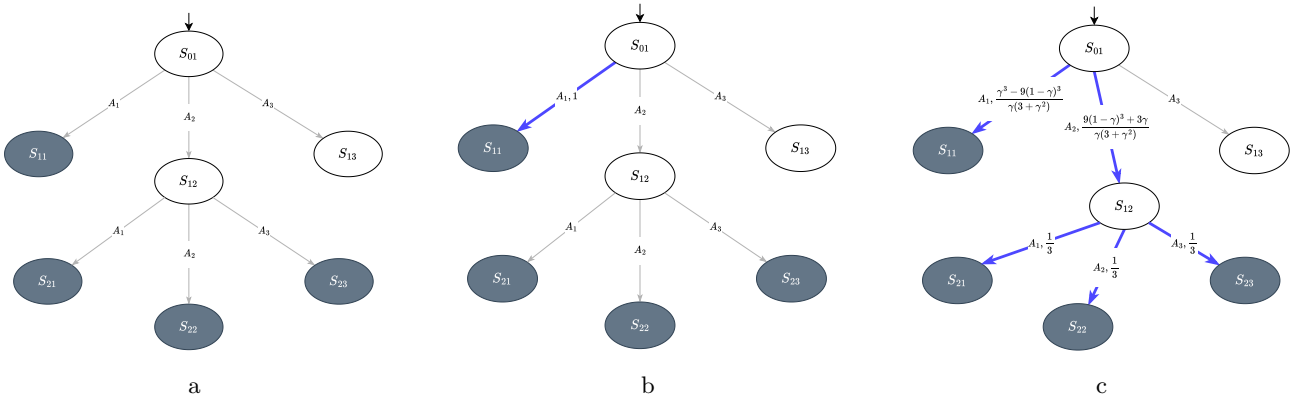
Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7553–7562. PMLR, 2019.

A Balancing Return and Goal State Dispersion

The return maximization objective and goal state dispersion objective may not necessarily align. Particularly, we highlight the following two underlying reasons that may lead to this misalignment:

1. **Discounting.** Because of discounting, the return maximization objective encourages visitation of goal states that are closer to the start states over the ones that are farther. This is demonstrated via a simple example in Figure 4. For a discount factor value < 1, the highest expected return would be given by a policy that takes action A_1 in state S_{01}. Maximizing the proposed objective, however, leads to a policy (or a mixture) that visits all the goal states with non-zero probability. Further, for $\gamma = 1$, it can be observed that the objective is maximized when all goal states are visited equally likely as the objectives align in this case.

2. **Environment Dynamics.** The return maximization and diverse goal visitation objectives may also conflict because of the environment dynamics. Figure 5 illustrates this with an example. In this example, even with $\gamma = 1$, the two objectives do not align. While the highest return is obtained by exploiting smaller loop comprising only the goal states, a more diverse goal coverage can be obtained by also visiting goal states in the loop which contains a non-goal state.



a b c

Figure 4: (a) A continuing deterministic MDP with a start state, and highlighted states as the goal states. The terminal states act as sink states i.e. any action taken in these states leads back to the same state. (b) Marginal state distribution with the highest expected return as it is concentrated on the closest goal state to avoid higher discounting penalty. (c) Marginal state distribution that is well dispersed across goal states but compromises on the expected return by incurring larger discounting penalty.

B Proof of Lemma 2.1

1. $\overline{\Pi}$ is a convex set.

Proof. $\overline{\Pi}$ is convex as a convex combination of two valid probability distributions over Π is another valid probability distribution over Π and thus belongs to $\overline{\Pi}$. □

2. $\mathcal{Z}$ is a concave function over $\overline{\Pi}$.

Proof. Define $d(s) = d[\overline{\pi}](s)$. Let $\mathcal{G}(d) = \sum_{s \in \mathcal{S}^+} \left[d(s) - \frac{1}{2} d(s)^2 \right]$.

We first show that $\mathcal{G}$ is concave over the space of all valid probability mass functions $d \in \Delta\mathcal{S}$ by showing that the Hessian of $\mathcal{G}$ is negative semi-definite.

$$\frac{\delta G}{\delta d(s_i)} = R(s_i) \left[1 - d(s_i) \right]$$

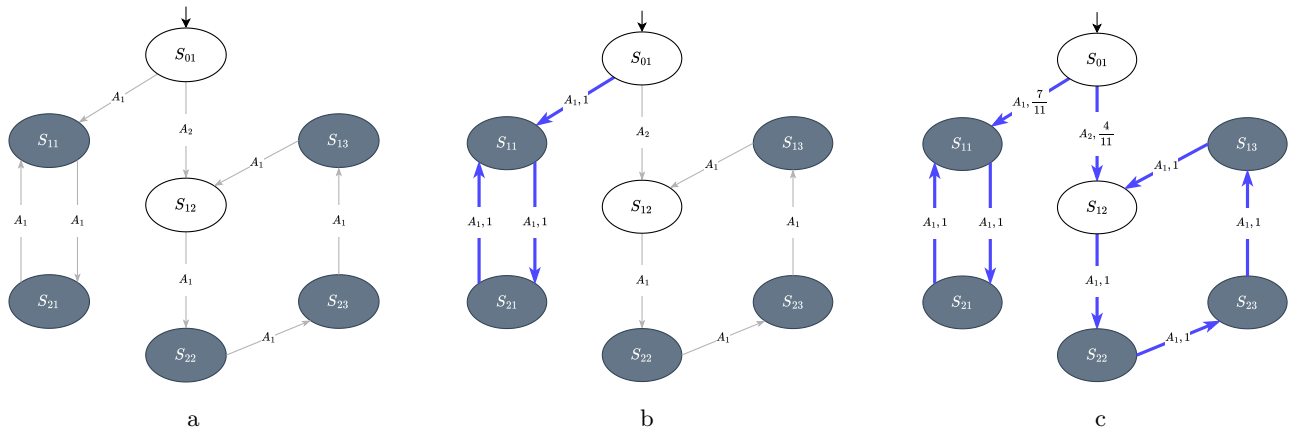


Figure 5: (a) A continuing deterministic MDP with a start state, and highlighted states as the goal states. (b) Marginal state distribution with the highest expected return as it is concentrated densely on the goal state by avoiding visitation on non-goal state. (c) Marginal state distribution that is obtained for $\gamma = 1$ by optimizing for the DDGC objective is well dispersed across goal states but compromises on expected return by visiting one non-goal state.

$$H_{ij} = \frac{\delta^2 G}{\delta d(s_i)\delta d(s_j)} = \begin{cases} -R(s_i) & \text{if } s_i = s_j \\ 0 & \text{if } s_i \neq s_j \end{cases}$$

For any non-zero $v \in \mathbb{R}^{|\mathcal{S}|}$,

$$v^T H_{ij} v = \sum_i \sum_j v_i H_{ij} v_j = \sum_i v_i^2 H_{ii} + \sum_{i \neq j} v_i H_{ij} v_j = -\sum_i v_i^2 R(s_i) \leq 0$$

Thus, $v^T H_{ij} v \leq 0$. So, H is negative semi-definite.

$$\begin{aligned} \mathcal{Z}(\overline{\pi}_1) &= \mathcal{G}(d[\overline{\pi}_1]) = \mathcal{G}(d_1) \\ \mathcal{Z}(\overline{\pi}_2) &= \mathcal{G}(d[\overline{\pi}_2]) = \mathcal{G}(d_2) \\ \mathcal{Z}(\alpha\overline{\pi}_1 + (1-\alpha)\overline{\pi}_2) &= \mathcal{G}(\alpha d[\overline{\pi}_1] + (1-\alpha) d[\overline{\pi}_2]) = \mathcal{G}(\alpha d_1 + (1-\alpha) d_2) \end{aligned}$$

Since, we have shown $\mathcal{G}$ to be concave, from the above argument it follows that $\mathcal{Z}$ is concave as well.

□

3. $C_{\mathcal{Z}} = \sup_{\substack{\overline{\pi}_1, \overline{\pi} \in \overline{\Pi} \\ \lambda \in (0,1) \\ \overline{\pi}_2 = \overline{\pi}_1 + \lambda(\overline{\pi} - \overline{\pi}_1)}} \frac{2}{\lambda^2} \left[\mathcal{Z}(\overline{\pi}_1) + \langle \overline{\pi}_2 - \overline{\pi}_1, \nabla_\pi \mathcal{Z}(\overline{\pi}_1) \rangle - \mathcal{Z}(\overline{\pi}_2) \right] < \infty$

$$\frac{\delta \mathcal{Z}}{\delta \overline{\pi}(\pi)} = \sum_{s \in \mathcal{S}^+} d[\pi](s) \left(1 - d[\overline{\pi}](s)\right)$$

Using the expression for functional derivative to find the required dot product,

$$\langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi}) \rangle = \int_{\Pi} \sum_{s \in \mathcal{S}^+} d[\pi](s)(1 - d[\overline{\pi}](s))\overline{\mu}(\pi) d\pi = \sum_{s \in \mathcal{S}^+} (1 - d[\overline{\pi}](s)) \int_{\Pi} d[\pi](s)\overline{\mu}(\pi) d\pi$$
$$= \sum_{s \in \mathcal{S}^+} d[\overline{\mu}](s)(1 - d[\overline{\pi}](s)) = \sum_{s \in \mathcal{S}^+} d[\overline{\mu}](s) - \sum_{s \in \mathcal{S}^+} d[\overline{\mu}](s) d[\overline{\pi}](s)$$

$$\langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi}) \rangle = \sum_{s \in \mathcal{S}^+} d[\overline{\mu}](s) - \sum_{s \in \mathcal{S}^+} d[\overline{\mu}](s) d[\overline{\pi}](s) \tag{3}$$

Using expression for dot product and definition of $\mathcal{Z}$,

$$\begin{aligned}\mathcal{Z}(\overline{\pi}_1) + \langle \overline{\pi}_2 - \overline{\pi}_1, \nabla_\pi \mathcal{Z}(\overline{\pi}_1) \rangle - \mathcal{Z}(\overline{\pi}_2) = &\sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_1](s) - d[\overline{\pi}_2](s)) \\ &+ \frac{1}{2} \sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_2](s)^2 - d[\overline{\pi}_1](s)^2) \\ &+ \sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_2](s) - d[\overline{\pi}_1](s)) \\ &+ \sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_1](s)^2 - d[\overline{\pi}_1](s) d[\overline{\pi}_2](s)) \\ = &\frac{1}{2} \sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_2](s) - d[\overline{\pi}_1](s))^2\end{aligned}$$

As $\overline{\pi}_2 - \overline{\pi}_1 = \lambda(\overline{\pi} - \overline{\pi}_1)$, we have,

$$d[\overline{\pi}_2](s) - d[\overline{\pi}_1](s) = \lambda(d[\overline{\pi}](s) - d[\overline{\pi}_1](s))$$

Squaring and summing over all positive states, we have

$$\sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_2](s) - d[\overline{\pi}_1](s))^2 = \lambda^2 \sum_{s \in \mathcal{S}^+} (d[\overline{\pi}](s) - d[\overline{\pi}_1](s))^2 \leq 2\lambda^2$$

Re-arranging we have,

$$\frac{1}{\lambda^2} \leq \frac{2}{\sum_{s \in \mathcal{S}^+} (d[\overline{\pi}_2](s) - d[\overline{\pi}_1](s))^2}$$

Combining above results, we have

$$\frac{2}{\lambda^2} \left[\mathcal{Z}(\overline{\pi}_1) + \langle \overline{\pi}_2 - \overline{\pi}_1, \nabla_\pi \mathcal{Z}(\overline{\pi}_1) \rangle - \mathcal{Z}(\overline{\pi}_2) \right] \leq 1$$

Thus, $C_{\mathcal{Z}} \leq 1$.

C Proof of Lemma 3.1

Proof. Define $d_k = d[\overline{\pi}_k]$ and $\mathcal{G}(d) = \sum_{s\in\mathcal{S}^+}\left[d(s) - (1/2)d(s)^2\right]$. Note that $\mathcal{G}(d[\overline{\pi}]) = \mathcal{Z}(\overline{\pi})$.

The update step in algorithm is $\overline{\pi}_k = (1-\lambda_k)\overline{\pi}_{k-1} + \lambda_k\overline{\mu}_k = \overline{\pi}_{k-1} + \lambda_k(\overline{\mu}_k - \overline{\pi}_{k-1})$. This gives us $\overline{\pi}_k - \overline{\pi}_{k-1} = \lambda_k(\overline{\mu}_k - \overline{\pi}_{k-1})$.

From Lemma 2.1, we have $\mathcal{Z}(\overline{\pi}_k) - \mathcal{Z}(\overline{\pi}_{k-1}) \geq \langle \overline{\pi}_k - \overline{\pi}_{k-1}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle - \frac{C_\mathcal{Z}}{2}\lambda_k^2$.

Following the proof structure in Jaggi (2013), adding and subtracting $\mathcal{Z}(\overline{\pi}^*)$ on LHS and multiplying both sides by -1, we have,

$$h(\overline{\pi}_k) - h(\overline{\pi}_{k-1}) \leq \frac{C_\mathcal{Z}}{2}\lambda_k^2 - \lambda_k\langle \overline{\mu}_k - \overline{\pi}_{k-1}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle \tag{4}$$

From Equation 3,

$$\langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi})\rangle = \sum_{s\in\mathcal{S}^+} d[\overline{\mu}](s) - \sum_{s\in\mathcal{S}^+} d[\overline{\mu}](s)d[\overline{\pi}](s)$$

Define $r_k(s)$ as follows:

$$r_k(s) = \begin{cases} 1 - d[\overline{\pi}_{k-1}](s), & s \in \mathcal{S}^+ \\ 0, & s \in \mathcal{S}^- \end{cases} \tag{5}$$

So we have,

$$\langle \overline{\mu}_k, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle = \sum_{s\in\mathcal{S}} d[\overline{\mu}_k](s) r_k(s) \tag{6}$$

Since $\overline{\Pi}$ is a convex set as established in 2.1, $\exists \mu \in \Pi$ such that the policy mixture containing only μ maximizes $\langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle$ because policy mixtures containing a single policy from set Π form the extreme points of the convex set $\overline{\Pi}$.

Note that for a policy mixture $\overline{\mu}_k = \{(\mu_k, 1)\}$, $\forall s \in \mathcal{S}$, $d[\overline{\mu}_k](s) = d[\mu_k](s)$. And maximizing $\langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle$ boils down to solving the following problem

$$\underset{\mu_k\in\Pi}{\operatorname{argmax}} \sum_{s\in\mathcal{S}} d[\mu_k](s) r_k(s)$$

Note that this is equivalent to solving an RL problem where the MDP is the same as original MDP but with r_k as the reward function instead. This emergent structure has been previously exploited in Hazan et al. (2019). $\langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle$ is the expected return of a policy mixture $\overline{\mu}$ w.r.t. reward function r_k. Let $\overline{\mu}_k^*$ be the optimal policy mixture for the reward function r_k. Assuming that the solution given by RL algorithm is at most ϵ_k sub-optimal with probability $1-\delta_k$, we have

$$\begin{aligned}
&\langle \overline{\mu}_k, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle + \epsilon_k \geq \langle \overline{\mu}_k^*, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle \\
&\langle \overline{\mu}_k, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle + \epsilon_k \geq \langle \overline{\mu}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle \quad \forall \overline{\mu} \in \overline{\Pi} \\
&\langle \overline{\mu}_k, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle + \epsilon_k \geq \langle \overline{\pi}^*, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle
\end{aligned}$$

From Equation 4 and 6, we have

$$h(\overline{\pi}_k) - h(\overline{\pi}_{k-1}) \leq \frac{C_\mathcal{Z}}{2}\lambda_k^2 - \lambda_k\langle \overline{\mu}_k - \overline{\pi}_{k-1}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1})\rangle$$

With probability $1-\delta_k$,

$$h(\overline{\pi}_k) - h(\overline{\pi}_{k-1}) \leq \frac{C_{\mathcal{Z}}}{2}\lambda_k^2 - \lambda_k \langle \overline{\pi}^* - \overline{\pi}_{k-1}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1}) \rangle + \lambda_k \epsilon_k$$

Since $\mathcal{Z}$ is concave, $\mathcal{Z}(\overline{\pi}^*) - \mathcal{Z}(\overline{\pi}_{k-1}) \leq \langle \overline{\pi}^* - \overline{\pi}_{k-1}, \nabla_\pi \mathcal{Z}(\overline{\pi}_{k-1}) \rangle$. Hence,

$$\begin{aligned} h(\overline{\pi}_k) - h(\overline{\pi}_{k-1}) &\leq \frac{C_{\mathcal{Z}}}{2}\lambda_k^2 - \lambda_k(\mathcal{Z}(\overline{\pi}^*) - \mathcal{Z}(\overline{\pi}_{k-1})) + \lambda_k \epsilon_k \\ h(\overline{\pi}_k) - h(\overline{\pi}_{k-1}) &\leq \frac{C_{\mathcal{Z}}}{2}\lambda_k^2 - \lambda_k h(\overline{\pi}_{k-1}) + \lambda_k \epsilon_k \\ h(\overline{\pi}_k) &\leq (1-\lambda_k) h(\overline{\pi}_{k-1}) + \frac{C_{\mathcal{Z}}}{2}\lambda_k^2 + \lambda_k \epsilon_k \end{aligned}$$

For $\lambda_k = \frac{2}{k+1}$,

$$h(\overline{\pi}_k) - \frac{k-1}{k+1} h(\overline{\pi}_{k-1}) \leq \frac{2C_{\mathcal{Z}}}{(k+1)^2} + \frac{2\epsilon_k}{k+1}$$

Multiplying $k(k+1)$ on both sides,

$$k(k+1)h(\overline{\pi}_k) - (k-1)kh(\overline{\pi}_{k-1}) \leq \frac{2kC_{\mathcal{Z}}}{k+1} + 2k\epsilon_k$$

Summing for $k = 1 \cdots K$ by union bound and Boole's inequality, with probability $\geq 1 - \sum_{k=1}^{K} \delta_k$,

$$\begin{aligned} K(K+1)h(\overline{\pi}_K) &\leq 2\sum_{k=1}^{K} \left(\frac{kC_{\mathcal{Z}}}{k+1} + k\epsilon_k \right) \\ &\leq 2\sum_{k=1}^{K} (C_{\mathcal{Z}} + k\epsilon_k) \\ h(\overline{\pi}_K) &\leq \frac{2C_{\mathcal{Z}}}{K+1} + \frac{2}{K(K+1)} \sum_{k=1}^{K} k\epsilon_k \end{aligned}$$

□

D Proof of Lemma 3.2

Proof. Let μ_k be the policy obtained using FQI with rewards estimated with $\hat{d}_k$. Since, FQI is an iterative algorithm, let the policy obtained at i^{th} iteration be denoted by μ_k^i and the corresponding Q-value function by f_i. Also, denote the optimal policy for MDP with true reward function as defined in Equation 5 by μ_k^* and the corresponding Q-value function by Q^*. We use $V(\pi)$ to denote the performance of a policy π for MDP with true reward as defined in Equation 5. Note that there are two sources of error here, 1) our algorithm optimizes for a pseudo-reward which is essentially an estimate of the true reward, and 2) inherent error because of FQI. Both these factors contribute to the sub-optimality of μ_k. We denote the discounted marginal state distribution induced by the policy π under aforementioned MDP as d^π.

Using performance difference lemma from Agarwal et al. (2019),

$$\begin{aligned}(1-\gamma)(V(\mu_k^*)-V(\mu_k^i)) &= \mathbb{E}_{s\sim d^{\mu_k^i}}\left[Q^*(s,\mu^*(s))-Q^*(s,\mu_k^i(s))\right]\\ &\le \mathbb{E}_{s\sim d^{\mu_k^i}}\left[Q^*(s,\mu^*(s))-f_i(s,\mu^*(s))+f_i(s,\mu_k^i(s))-Q^*(s,\mu_k^i(s))\right]\\ &\le \|Q^*-f_i\|_{1,d^{\mu_k^i}\odot\mu^*}+\|Q^*-f_i\|_{1,d^{\mu_k^i}\odot\mu_k^i}\\ &\le \|Q^*-f_i\|_{2,d^{\mu_k^i}\odot\mu^*}+\|Q^*-f_i\|_{2,d^{\mu_k^i}\odot\mu_k^i}\end{aligned}$$

Consider state-action distribution β induced by some policy.

Let $\beta^{i-1}(s',a') = \sum_{s,a}\beta^i(s,a)P(s'|s,a)\mathbf{1}\{a' = \text{argmax}_a\left[Q^*(s',a)-f_{i-1}(s',a)\right]^2\}$. Also, let $\beta_P^i(s') = \sum_{s,a}\beta^i(s,a)P(s'|s,a)$. We have

$$\begin{aligned}\|Q^*-f_i\|_{2,\beta} &\le \|Q^*-\mathcal{T}f_{i-1}\|_{2,\beta}+\|f_i-\mathcal{T}f_{i-1}\|_{2,\beta}\\ &\le \sqrt{\mathbb{E}_{\substack{s,a\sim\beta\\ s'\sim P(\cdot|s,a)}}\begin{bmatrix} R(s')\big(d[\overline{\pi}_k](s')-\hat{d}_k(s')\big)\\ +\gamma\big(\max_{a'}Q^*(s',a')-\max_{a'}f_{i-1}(s',a')\big)\end{bmatrix}^2}+\sqrt{C}\|f_i-\mathcal{T}f_{i-1}\|_{2,\nu}\\ &\le \|R\odot(r_k-\hat{r}_k)\|_{2,\beta_P}+\gamma\|Q^*-f_{i-1}\|_{2,\beta'}+\sqrt{C}\|f_i-\mathcal{T}f_{i-1}\|_{2,\nu}\end{aligned}$$

Let $\beta^i=\beta$ and $\beta_P^i=\beta_P$,

$$\begin{aligned}\|Q^*-f_i\|_{2,\beta^i}-\gamma\|Q^*-f_{i-1}\|_{2,\beta^{i-1}} &\le \|R\odot(r_k-\hat{r}_k)\|_{2,\beta_P^i}+\sqrt{C}\|f_i-\mathcal{T}f_{i-1}\|_{2,\nu}\\ &\cdots\\ \|Q^*-f_1\|_{2,\beta^1}-\gamma\|Q^*-f_0\|_{2,\beta^0} &\le \|R\odot(r_k-\hat{r}_k)\|_{2,\beta_P^1}+\sqrt{C}\|f_1-\mathcal{T}f_0\|_{2,\nu}\end{aligned}$$

Multiplying j^{th} inequality by γ^{j-1} and summing up the above inequalities,

$$\|Q^*-f_i\|_{2,\beta}\le\sum_{j=1}^{i}\gamma^{i-j}\|R\odot(r-\hat{r})\|_{2,\beta_P^j}+\sqrt{C}\sum_{j=1}^{i}\gamma^{j-1}\|f_{i-j+1}-\mathcal{T}f_{i-j}\|_{2,\nu}+\gamma^i\|Q^*-f_0\|_{2,\beta^0}$$

We will now show $\|R\odot(r-\hat{r})\|_{2,\alpha}\le\epsilon_d$.

We know that $r(s)-\hat{r}(s)=0$ for $s\notin\mathcal{S}^+$. For simplicity, denote $\hat{d}_k$ by $\hat{d}$ and $d[\overline{\pi}_k]$ by d. For $s\in\mathcal{S}^+$, we have

$$|r(s)-\hat{r}(s)| = |\hat{d}(s)-d(s)|$$

In Appendix E, we show that $|\hat{d}(s)-d(s)|\le\epsilon_d$ with probability $\ge 1-\delta_d$.

So, with probability $\ge 1-\delta_d$,

$$|r(s)-\hat{r}(s)|\le\epsilon_d$$

As the above inequality holds for all $s\in\mathcal{S}^+$,

$$\|r(s)-\hat{r}(s)\|_\infty\le\epsilon_d$$

So, we have that with probability $\ge 1-\delta_d$, $\|R\odot(r-\hat{r})\|_{2,\alpha}\le\|R\|_\infty\cdot\|(r-\hat{r})\|_\infty\le\epsilon_d$ and thus assuming $\|f_{i-t+1}-\mathcal{T}f_{i-t}\|_{2,\nu}\le\omega$ for all $t\in\{0,\ldots,i\}$, we have,

$$\|Q^* - f_i\|_{2,\beta} \leq \frac{\epsilon_d + \sqrt{C}\omega}{1-\gamma} + \gamma^i V_{max} \tag{7}$$

Following the pseudo-dimension based analysis of FQI in Appendix F.2 of Le et al. (2019), we bound ω with high probability. Specifically, we use the result from Equation 35 of Le et al. (2019),

$$P\left\{\|f_j - \mathcal{T}f_{j-1}\|_{2,\nu} > \epsilon + 4\epsilon^2_{\text{approx}}\right\} \leq \xi(\epsilon) = 14 \cdot e \cdot (\dim_{\mathcal{F}} + 1)\left(\frac{640V^2_{\max}}{\epsilon}\right)^{\dim_{\mathcal{F}}} \cdot \exp\left(-\frac{\epsilon H N_T}{48 \cdot 214 V^4_{\max}}\right) + \exp\left(-\frac{3}{416}\frac{\epsilon H N_T}{V^2_{\max}}\right)$$

Since $V_{\max} \geq 1$,

$$-\frac{\epsilon H N_T}{48 \cdot 214 V^4_{\max}} \geq -\frac{3}{416}\frac{\epsilon H N_T}{V^2_{\max}}$$

Hence,

$$P\left\{\|f_j - \mathcal{T}f_{j-1}\|_{2,\nu} - 4\epsilon^2_{\text{approx}} > \epsilon\right\} \leq \xi(\epsilon) = 28 \cdot e \cdot (\dim_{\mathcal{F}} + 1)\left(\frac{640V^2_{\max}}{\epsilon}\right)^{\dim_{\mathcal{F}}} \cdot \exp\left(-\frac{\epsilon H N_T}{48 \cdot 214 V^4_{\max}}\right)$$

Consider δ such that,

$$\delta \geq 28 \cdot e \cdot (\dim_{\mathcal{F}} + 1)\left(\frac{640V^2_{\max}}{\epsilon}\right)^{\dim_{\mathcal{F}}} \cdot \exp\left(-\frac{\epsilon H N_T}{48 \cdot 214 V^4_{\max}}\right)$$

Taking negative log on both sides, we obtain the following

$$\log\left(\frac{1}{\delta}\right) \leq \frac{\epsilon H N_T}{48 \cdot 214 V^4_{\max}} - \log\left[28e(\dim_{\mathcal{F}} + 1)(640V^2_{\max})^{\dim_{\mathcal{F}}}\right] + \dim_{\mathcal{F}} \log \epsilon = \chi(\epsilon)$$

For the following value of ϵ, we have $\chi(\epsilon) \geq \log(1/\delta)$.

$$\epsilon = \frac{48 \cdot 214 V^4_{\max}}{H N_T}\left[\log\frac{28e(\dim_{\mathcal{F}} + 1)}{\delta} + \dim_{\mathcal{F}} \log(640 V^2_{\max} H N_T)\right] \tag{8}$$

Thus, with probability $\geq 1 - \delta$ we have,

$$\|f_j - \mathcal{T}f_{j-1}\|_{2,\nu} < \sqrt{4\epsilon^2_{\text{approx}} + \epsilon} \tag{9}$$

Using Expressions 8, 7 and 9, we can conclude that the following holds with probability $\geq 1 - (\delta + \delta_d)$

$$\|Q^* - f_{N_{\text{FQI}}}\|_{2,\beta} \leq \frac{\sqrt{C}}{(1-\gamma)^2}\sqrt{4\epsilon^2_{\text{approx}} + \frac{48 \cdot 214 \cdot V^4_{\max}}{H N_T}\left[\log\frac{28e(\dim_{\mathcal{F}} + 1)}{\delta} + \dim_{\mathcal{F}} \cdot \log(640 H N_T V^2_{\max})\right]} + \frac{\gamma^{N_{FQI}} V_{\max}}{1-\gamma} + \frac{1}{(1-\gamma)^2}\left(\gamma^H + \sqrt{\frac{\log(2/\delta_d)}{2N_T}}\right)$$

□

E Error in discounted frequency approximation

Consider the following distributions,

$$d(s) = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t P(s_t = s|s_0 \sim \rho_0, \pi)\right]$$
$$\hat{d}(s) = (1-\gamma)\sum_{i=1}^{N_T}\sum_{t=0}^{H-1}\gamma^t I(s_{i,t} = s)$$
$$d_H(s) = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{H-1}\gamma^t P(s_t = s|s_0 \sim \rho_0, \pi)\right]$$

Using triangle inequality,

$$|\hat{d}(s) - d(s)| \leq |\hat{d}(s) - d_H(s)| + |d_H(s) - d(s)|$$

We bound each of the terms on RHS,

$$\begin{aligned}|d_H(s) - d(s)| &= (1-\gamma)\left|\sum_{t=0}^{H-1}\gamma^t P(s_t = s|s_0 \sim \rho_0, \pi) - \sum_{t=0}^{\infty}\gamma^t P(s_t = s)\right| \\ &= (1-\gamma)\left|\sum_{t=H}^{\infty} P(s_t = s)\right| \\ &\leq (1-\gamma)\frac{\gamma^H}{1-\gamma} = \gamma^H\end{aligned}$$

$|\hat{d}(s) - d_H(s)|$ is deviation of sample mean from true expectation of random variable $X_i(s) = (1-\gamma)\sum_{t=0}^{H-1}\gamma^t I(s_{i,t} = s)$. Using Hoeffdings' inequality,

$$P\left(\left|\hat{d}(s) - d_H(s)\right| \geq \sqrt{\frac{\log 2/\delta_d}{2N_T}}\right) \leq \delta_d$$

Thus, with probability $\geq 1-\delta_d$, $\left|\hat{d}(s) - d_H(s)\right| \leq \sqrt{\frac{\log 2/\delta_d}{2N_T}}$.

Combining, the following holds with probability $\geq 1-\delta_d$

$$\left|\hat{d}(s) - d(s)\right| \leq \gamma^H + \sqrt{\frac{\log 2/\delta_d}{2N_T}}$$

F Experimental Details (Tabular)

For the simple MDP, we implement a vectorized step function for faster training. The baseline algorithms are based on tabular Q Learning. We use Algorithm 1 for DDGC implementation. As described in Figure 1a, the MDP is continuing with all the terminal states acting like sink states i.e. any action taken from those states leads back to the same state. Since this is a discrete MDP, the learnt policies are actually deterministic. The policy mixture comprises of multiple deterministic policies with possibly different weights.

G Experimental Details (JaxGCRL Benchmark)

We describe the experimental setup used for experiments on JaxGCRL benchmark. We split the details into 3 parts: environments, algorithm and baselines, and network architectures.

G.1 Environments

JaxGCRL (Bortkiewicz et al., 2025) provides vectorized goal-conditioned environments. This resembles closely to our problem setting making it suitable for experiments on simulated robotic environments to empirically validate the efficacy of our algorithm. We adapt JaxGCRL for our setting with following modifications:

- We fix goal criterion for each environment. This leads to multiple goals but the observation vector has no information about the goal. Thus, on every reset of the environment, the initial state may differ but the set of goal states remains the same.
- We set dense reward flag to `False` for each environment to have outcome based sparse binary rewards.

Table 1 describes the state-action space and goal criteria for each of the environments that we experiment on.

Environment	State Space	Action Space	Goal Criteria
Reacher	$\mathbb{R}^{10}$	$(-1.0, 1.0)^2$	Tip-Target Distance < 0.05
Pusher	$\mathbb{R}^{20}$	$(-1.0, 1.0)^7$	Arm-Goal Distance < 0.1
Ant	$\mathbb{R}^{29}$	$(-1.0, 1.0)^8$	Distance from Goal < 0.5
Half Cheetah	$\mathbb{R}^{18}$	$(-1.0, 1.0)^6$	Distance from Goal < 0.5

Table 1: Description of environments used from JaxGCRL benchmark. Horizon length is restricted to **200** steps for all the environments and the discount factor is set as **0.999** for all the environments. Sparse rewards are used in all the environments. Specifically, the extrinsic reward is **1** when the agent moves into a goal state and **0** otherwise.

G.2 Algorithm & Baselines

We choose one baseline from each class of prior works described in Section 1.1 except GCRL (as algorithms for GCRL require goal information to be part of the observation which is not the case in our problem setting.) Table 2 contains information about different hyperparameter values for all the algorithms. Adam optimizer is used in all experiments. Baseline implementations closely follow the experimental details described in Lee et al. (2019).

G.2.1 Practical DDGC for Continuous State Action Spaces

For continuous state-action space, we use a variant of FQI proposed by Antos et al. (2007) called Fitted Actor Critic. We also incorporate the practical modifications - Exploratory Sampling and Goal Buffer proposed in Section 3 in the algorithm and provide a complete pseudo-code in Algorithm 2. The exploration strategy could be as simple as a random policy or a more efficient strategy like pseudo-counts. For the sake of completion, we also provide the pseudo code of Fitted Actor Critic in Section 3.

G.3 Network Architectures

Policy Network comprises 2 ReLU activated hidden layers of dimensions 256 each followed by separate dense layers for mean and log standard deviation. The output for log standard deviation is clipped for stability. The predicted action is sampled from Tanh-transformed normal distribution with obtained mean and log standard deviation.

Critic Network also comprises 2 ReLU activated hidden layers of dimension 256 each and is followed by a dense layer. State and action vectors are concatenated before passing to the network.

VAE Density Network is a Variational Autoencoder (VAE) (Kingma and Welling, 2013). Encoder comprises 2 ReLU activated hidden layers of dimensions 256 each followed by separated dense layers for mean and log standard deviation with latent dimension equal to 32. The decoder mirrors the encoder with 2 ReLU activated hidden layers of dimensions 256 each followed by a dense layer for reconstruction.

Algorithm 2 Practical Dense & Diverse Goal Coverage for Continuous Spaces (P-DDGC: CS)

Input: $\mathcal{M}$ (MDP), h_d (State Space Discretization Function)
Initialize: $\overline{\pi}_0$ (policy mixture), π_X (exploratory policy), $\mathcal{B}_g^0$ (buffer to store goal reaching trajectories)
Hyper-parameters: N_T (Number of Trajectories), H (Horizon), K (Mixture Size)

for k in $1 \cdots$ K **do**

$\Gamma_k \sim \mathcal{M}(\overline{\pi}_{k-1})$, $|\Gamma_k| = HN_T$ ▷ Sample data by following $\overline{\pi}$, $\Gamma_k^{(i)} \in \mathcal{S} \times \mathcal{A} \times \{0,1\} \times \mathcal{S} \times [H]$, $[H] := \{1, 2, \cdots, H\}$

$\Gamma_X \sim \mathcal{M}(\pi_X)$ ▷ Sample from exploratory policy

$\mathcal{B}_g^k \leftarrow \mathcal{B}_g^{k-1} \cup \{(s,a,r,s',t) \in \tau | \sum_{r \in \tau} r > 0, \tau \in \Gamma_k \cup \Gamma_X\}$ ▷ Add goal reaching trajectories to goal buffer

$\hat{d}_k(s) = \frac{(1-\gamma)}{N_T(1-\gamma^H)} \sum_{(s,a,r,s',t) \in \Gamma_k} \gamma^{t-1} \mathbb{I}(h_d(s') = h_d(s))$ ▷ Normalized discounted discretized state visitation frequency

$\hat{r}_k(s) = \begin{cases} 1 - \hat{d}_k(s), & s \in \mathcal{S}^+ \\ 0, & s \in \mathcal{S}^- \end{cases}$ ▷ Custom reward

$\Gamma'_k \leftarrow \{(s,a,\hat{r}_k(s'),s') | (s,a,r,s',t) \in \Gamma_k\}$ ▷ Update batch with custom reward

$\mu_k \leftarrow \text{RL}(\mathcal{M}, \Gamma'_k \cup \Gamma_X \cup \mathcal{B}_g^{k-1})$ ▷ Batch RL

$\overline{\mu}_k(\pi) = \begin{cases} 1, & \pi = \mu_k \\ 0, & \pi \neq \mu_k \end{cases}$ ▷ Policy mixture with all weight on μ_k

$\overline{\pi}_k \leftarrow (1 - \lambda_k)\overline{\pi}_{k-1} + (\lambda_k)\overline{\mu}_k$, $\lambda_k = \frac{2}{k+1}$ ▷ Mixture update

end for
return $\overline{\pi}_K$

Algorithm 3 Fitted Actor Critic

Input: $\mathcal{M}$ (MDP), $\{s,a,r,s'\} \in \Gamma$ (Batch Data)
Initialize: π_0 (policy), f_0 (Q Value Approximator)
Hyper-parameters: N_{FQI} (Number of Steps)

for k in $1 \cdots$ N_{FQI} **do**

$f_k \leftarrow \text{argmin}_{f \in \mathcal{F}} \sum_{(s,a,r,s') \in \Gamma} (f(s,a) - [r + \gamma f_{k-1}(s', \pi(s'))])^2$ ▷ Updating Q Value Approximator

$\pi_k \leftarrow \text{argmax}_{\pi \in \Pi} \sum_{(s,a,r,s') \in \Gamma} f_k(s, \pi(s))$ ▷ Updating Policy

end for
return $\overline{\pi}_K$

G.4 Computational Requirements

All of the experiments were run on a system with standard 16-core CPU, 256GB of memory with one V100 GPU.

Algorithm	Hyperparameters Selected					Hyperparameters Considered
SAC	Hyperparameter	Reacher	Pusher	Ant	Half Cheetah	`Learning Rate: 1e-2, 1e-3, 1e-4, 1e-5`
	Policy Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Alpha Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Critic Learning Rate	1e-2	1e-2	1e-3	1e-3	
	Entropy Scale	2.0	7.0	8.0	6.0	
	Polyak Coefficient	0.005	0.005	0.005	0.005	
	Warmup Steps	1e5	1e5	1e6	1e6	
	Gradient Steps per Env Step	16	16	16	16	
	Buffer Size	1e5	1e5	1e6	1e6	
	Batch Size	256	256	256	256	
PC (SAC)	Hyperparameter	Reacher	Pusher	Ant	Half Cheetah	`Learning Rate: 1e-2, 1e-3, 1e-4, 1e-5` `Intrinsic Reward Scale: 1e-4, 1e-2, 1.0`
	Policy Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Alpha Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Critic Learning Rate	1e-2	1e-2	1e-3	1e-3	
	Density Learning Rate	1e-3	1e-3	1e-4	1e-3	
	Intrinsic Reward Scale	1e-4	1e-4	1e-2	1e-2	
	Entropy Scale	2.0	7.0	8.0	6.0	
	Polyak Coefficient	0.005	0.005	0.005	0.005	
	Warmup Steps	1e5	1e5	1e6	1e6	
	Gradient Steps per Env Step	16	16	16	16	
	Buffer Size	1e5	1e5	1e6	1e6	
	Batch Size	256	256	256	256	
SMM (SAC)	Hyperparameter	Reacher	Pusher	Ant	Half Cheetah	`Learning Rate: 1e-2, 1e-3, 1e-4, 1e-5`
	Policy Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Alpha Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Critic Learning Rate	1e-2	1e-2	1e-3	1e-3	
	Density Learning Rate	1e-3	1e-3	1e-3	1e-3	
	Entropy Scale	2.0	7.0	8.0	6.0	
	Polyak Coefficient	0.005	0.005	0.005	0.005	
	Warmup Steps	1e5	1e5	1e6	1e6	
	Gradient Steps per Env Step	16	16	16	16	
	Buffer Size	1e5	1e5	1e6	1e6	
	Batch Size	256	256	256	256	
	Mixture Size	8	8	8	8	
DDGC	Hyperparameter	Reacher	Pusher	Ant	Half Cheetah	`Learning Rate: 1e-2, 1e-3, 1e-4, 1e-5` `Discretization Precision: 1, 1e2, 1e4`
	Policy Learning Rate	1e-3	1e-3	1e-4	1e-4	
	Critic Learning Rate	1e-2	1e-2	1e-3	1e-3	
	Discretization Precision	1e4	1e2	1e2	1e2	
	Batch Size	256	256	256	256	
	Gradient Steps per Env Step	16	16	16	16	
	FQI Training Steps	1e2	1e2	1e3	1e3	
	Mixture Size	8	8	8	8	

Table 2: Equal learning rates were used for alpha as well as policy and density networks wherever applicable (from the list above) while the critic's learning rate was chosen to 0.1 times the policy's learning rate for stable training. Total environment steps used for each algorithm is the same, **1e7**.