Dense and Diverse Goal Coverage in Multi-Goal RL

Sagalpreet Singh, Rishi Saket, Aravindan Raghuveer

Google DeepMind

Introduction

RL typically optimizes for expected return, often exploiting a few reward sources. However, real-world applications (e.g., molecular discovery, robot manipulation, design verification) require a policy (or mixture thereof) inducing a dispersed state distribution over all rewarding states.

Trade-off. Exploration RL (e.g., MaxEnt, intrinsic rewards like RND) and State Marginal Matching (SMM) visit under-explored or non-goal states, wasting steps and reducing expected return. We formalize this trade-off and propose a solution.

Our key **contributions** are:

1. **Formulation:** We formalize frequently and uniformly visiting goal states as Multi-Goal RL in general MDPs.
2. **Objective:** We propose a concave objective balancing expected return and goal-state diversity using the Gini criterion.
3. **Algorithm:** We design an iterative Frank-Wolfe-based algorithm (DDGC) using Fitted Actor-Critic (FAC) for offline RL.
4. **Empirical Validation:** We show DDGC achieves near-optimal goal coverage and return, outperforming baselines on 4 MuJoCo benchmarks.

Multi-Goal RL Framework

Consider an infinite-horizon MDP $\mathcal{M} = \{S, A, R, \gamma, P, \rho_0\}$ where state space S is partitioned into goal states S^+ and non-goal states S^-. The reward function is binary: $R(s) = \mathbb{I}(s \in S^+)$.

For a single policy π:

- **Discounted Marginal Distribution:**

$$d[\pi](s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr[s_t = s \mid \pi] \tag{1}$$

- **Expected Return:** $J_\gamma(\pi) := \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t) \mid \pi\right] = \frac{1}{1-\gamma}\sum_{s\in S^+} d[\pi](s)$

For a policy mixture $\overline{\pi} \in \overline{\Pi} := \Delta(\Pi)$ (simplex of distributions over policy space Π):

- **Induced Distribution:** $d[\overline{\pi}](s) := \mathbb{E}_{\pi\sim\overline{\pi}}[d[\pi](s)]$
- **Expected Return:** $J_\gamma(\overline{\pi}) := \mathbb{E}_{\pi\sim\overline{\pi}}[J_\gamma(\pi)] = \frac{1}{1-\gamma}\sum_{s\in S^+} d[\overline{\pi}](s)$

Following a policy mixture means sampling $\pi \sim \overline{\pi}$ at the start of each episode.

Optimization Objective

To achieve both high expected return and diverse goal coverage, we define the diversity of the discounted marginal distribution over the goal manifold S^+ using the Gini criterion: $\mathcal{I}^{S^+}(\overline{\pi}) := -\frac{1}{2}\sum_{s\in S^+} d[\overline{\pi}](s)^2$. Our optimization objective is to maximize the sum of expected return and diversity:

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

Lemma 2.1:

1. The policy mixture class $\overline{\Pi}$ is a convex set.
2. $Z(\overline{\pi})$ is a concave function over $\overline{\Pi}$.
3. Z has a bounded curvature constant $C_Z \leq 1$.

The concavity ensures that local updates can efficiently converge to the globally optimal policy mixture.

Fitted Actor-Critic (FAC) Subroutine

For continuous control domains (e.g., MuJoCo / Brax), the offline batch RL subroutine is implemented as Fitted Actor-Critic (Algorithm 2). At iteration k, FAC trains the update policy μ_k on the accumulated dataset $\mathcal{D}'_k$ under updated custom rewards $\hat{r}_k$ by alternating between:

- **Q-Value Update:** Minimizing the Bellman residual over action-value class $\mathcal{F}$.
- **Policy Update:** Extracting the greedy policy with respect to the action-value function.

Under standard concentrability assumptions, this batch subroutine enables efficient data reuse as new policies are added to the mixture.

Algorithm 2 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 i in $1 \cdots$ N_{FQI} **do**
  $f_i \leftarrow \text{argmin}_{f\in\mathcal{F}} \sum_{(s,a,r,s')\in\Gamma} \left(f(s,a) - [r + \gamma f_{i-1}(s', \pi(s'))]\right)^2$ ▷ Update Q-value
  $\pi_i \leftarrow \text{argmax}_{\pi\in\Pi} \sum_{(s,a,r,s')\in\Gamma} f_i(s, \pi(s))$ ▷ Update Policy
end for
return $\pi_{N_{FQI}}$

Frank-Wolfe Optimization & DDGC

To optimize $Z(\overline{\pi})$, we propose the Dense & Diverse Goal Coverage (DDGC) algorithm based on the Frank-Wolfe (FW) framework:

- **Direction-Finding Step:** The FW step solves a linear maximization, equivalent to maximizing expected return in the MDP under a custom reward r_k:

$$r_k(s) = \begin{cases} 1 - d[\overline{\pi}_{k-1}](s) & s \in S^+ \\ 0 & s \in S^- \end{cases} \tag{3}$$

  which penalizes visiting frequently visited goal states.
- **Offline RL:** We train the update policy μ_k using a batch offline RL solver (e.g., Fitted Actor-Critic) on the accumulated dataset $\mathcal{D}_k$ with the custom reward.
- **Convergence:** Under standard assumptions, the sub-optimality gap is bounded by:

$$O(1/K) + O(\gamma^H + \gamma^{N_{FQI}}) + O(\epsilon_{\text{approx}}) + O(1/\sqrt{N_T}) \tag{4}$$

  where K is outer iterations, H is trajectory horizon, N_{FQI} is FQI update steps, N_T is the number of trajectories, and ϵ_{approx} is the FQI function approximation error. This guarantees PAC convergence to the near-optimal policy mixture.

Algorithms: DDGC

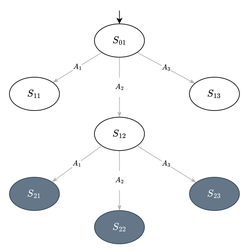
Algorithm 1 Dense & Diverse Goal Coverage (DDGC)

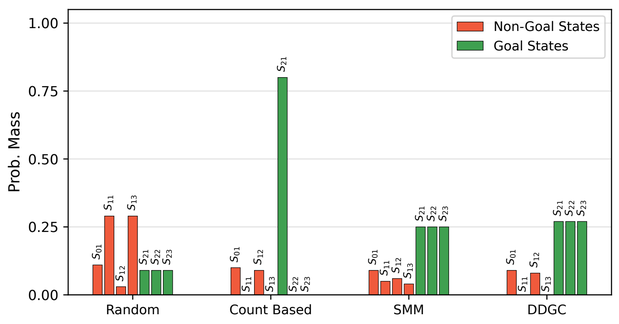
Input: $\mathcal{M}$ (MDP), h_d (State space discretization function, identity for discrete state spaces)
Initialize: $\overline{\pi}_0$ (policy mixture)
Hyper-parameters: N_T (Number of Trajectories), H (Horizon), K (Mixture Size)

$\Gamma_e \sim \text{RND}(\mathcal{M})$ ▷ Sample data using exploratory strategy
for k in $1 \cdots$ K **do**
  $\Gamma_k \sim \mathcal{M}(\overline{\pi}_{k-1})$, $|\Gamma_k| = HN_T$ ▷ Sample data following $\overline{\pi}$
  $\hat{d}_k(s) = \frac{(1-\gamma)}{N_T(1-\gamma^H)} \sum_{(s,a,r,s',t)\in\Gamma_k} \gamma^{t-1}\mathbb{I}(h_d(s') = h_d(s))$ ▷ Normalized discounted 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
  $\mathcal{D}_k \leftarrow \Gamma_e \cup \left(\bigcup_{i=1}^{k}\Gamma_i\right)$ ▷ Combine all data for batch RL
  $\mathcal{D}'_k \leftarrow \{(s, a, \hat{r}_k(s'), s') | (s, a, r, s', t) \in \mathcal{D}_k\}$ ▷ Update custom reward
  $\mu_k \leftarrow \text{RL}(\mathcal{M}, \mathcal{D}'_k)$ ▷ 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 $\overline{\pi}_K$

Controlled Synthetic MDP Experiments

We analyze the discounted marginal state distribution induced on a 6-state tree MDP to test diversity under exploration:



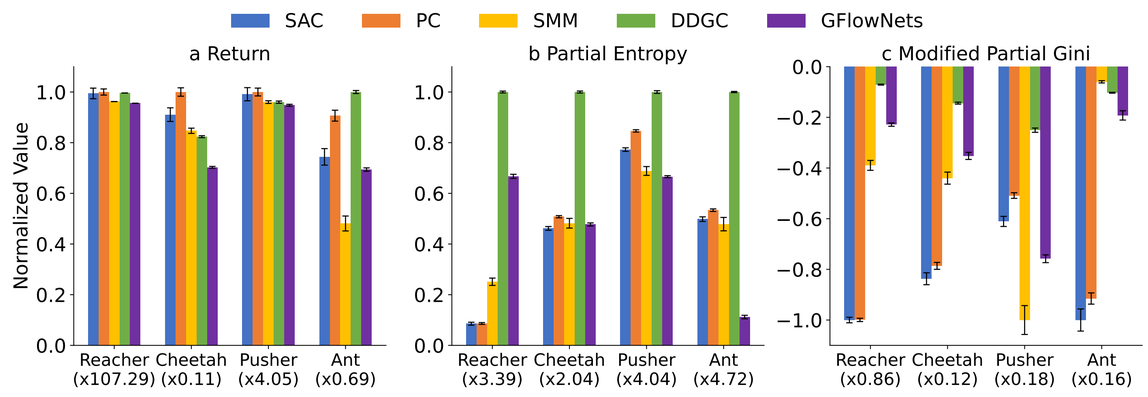


Analysis: Count-based RL exploits one path (S_{21}). SMM matches target but wastes mass on non-goals. DDGC achieves uniform mass on goals and 0 on non-goals.

Continuous Control Benchmarks (Brax)

We evaluate return and diversity metrics on Reacher, HalfCheetah, Pusher, and Ant environments with sparse rewards:

- **Discounted Return:** $(1-\gamma)J(\overline{\pi}) = \sum_{s\in S^+} d[\overline{\pi}](s)$ (dense goal coverage)
- **Partial Entropy (PE):** $-\sum_{s\in S^+} d[\overline{\pi}](s)\log d[\overline{\pi}](s)$ (diversity of coverage)
- **Modified Partial Gini (MPG):** $-\sum_{s\in S^+} d[\overline{\pi}](s)^2$ (uniformity of coverage)



[1] E. Hazan, S. Kakade, K. Singh, and A. Van Soest, "Provably efficient maximum entropy exploration," in *International Conference on Machine Learning*, pp. 2681–2691, 2019.

[2] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov, "Efficient exploration via state marginal matching," in *arXiv preprint arXiv:1906.05274*, 2019.

[3] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *International Conference on Machine Learning*, pp. 427–435, 2013.

[4] M. Andrychowicz, F. Wolski, A. Ray, S. Jonas, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, pp. 5055–5065, 2017.

github.com/google-deepmind/ddgc

{sagalpreet, rishisaket, araghuveer}@google.com