A Broader Impact

Designing appropriate rewards is a challenging task in real-world applications of reinforcement learning (RL). Often, researchers or algorithm engineers possess prior knowledge, such as rules and constraints, about the problem they are trying to solve, but are unable to represent this knowledge precisely with numeric values. Improper reward settings can lead to unexpected behaviors learned by an RL algorithm, resulting in exploitation of the rewards for higher returns. In this paper, we propose a potential-based exploration bonus as shaping reward that helps avoid repetitive and tedious tuning of rewards in RL applications, such as autonomous driving, robot learning control, video games and online recommendation/ad optimization. Our approach liberates researchers and algorithm engineers from the laborious task of reward tuning, while also providing insight into the quality of the designed rewards through the shaping weights learned by our methods.

B Bisimulation Relation and Wasserstein Distances

To delve deeply into the formulation of bisimulation metric, we present a detailed definition of both bisimulation relation and Wasserstein distances.

B.1 Definition of Bisimulation Relation

Definition 4. *(Bisimulation Relations Givan et al. [2003]) Given an MDP $\mathcal{M}$, an equivalence relation B between states is a bisimulation relation if, for all states $s_i, s_j \in \mathcal{S}$ that are equivalent under B (denoted $s_i \equiv_B s_i$) the following conditions hold:*

$$\mathcal{R}\left(s_i, a\right) = \mathcal{R}\left(s_j, a\right) \quad \forall a \in \mathcal{A}, \tag{11}$$

$$\mathcal{P}\left(G \mid s_i, a\right) = \mathcal{P}\left(G \mid s_j, a\right) \quad \forall a \in \mathcal{A}, \quad \forall G \in \mathcal{S}_B \tag{12}$$

where $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups G of equivalent states), and $\mathcal{P}(G \mid s, a) = \sum_{s' \in G} \mathcal{P}\left(s' \mid s, a\right)$

B.2 Wasserstein Distances

Definition 5. *(Wasserstein metric Villani et al. [2009]) Let $d : X \times X \to [0, \infty)$ be a distance function and Ω the set of all joint distributions with marginals μ and λ over the space X;*

$$W_p(d)(\mu, \lambda) = \left(\inf_{\omega \in \Omega} \mathbb{E}_{(x_1, x_2) \sim \omega}\left[d\left(x_1, x_2\right)^p\right]\right)^{\frac{1}{p}} \tag{13}$$

Definition 6. *(Dual formulation of the Wasserstein metric Villani et al. [2009]) Let $d : X \times X \to [0, \infty)$ be a distance function and μ and λ marginals over the space X;*

$$W_p(d)(\mu, \lambda) = \left(\sup_{\phi \oplus \psi \leq d^p} \mathbb{E}_{x_1 \sim \mu}\left[\phi\left(x_1\right)\right] + \mathbb{E}_{x_2 \sim \lambda}\left[\psi\left(x_2\right)\right]\right)^{\frac{1}{p}} \tag{14}$$

where $\phi \oplus \psi \leq d^p \Longleftrightarrow \phi(x) + \psi(y) \leq d(x, y)^p, \forall (x, y) \in X \times X$

This dual formulation takes a simple form for $p = 1$:

$$W_1(d)(\mu, \lambda) = \sup_{f \in \mathrm{Lip}_{1,d}(X)} \mathbb{E}_{x_1 \sim \mu}\left[f\left(x_1\right)\right] - \mathbb{E}_{x_2 \sim \lambda}\left[f\left(x_2\right)\right] \tag{15}$$

where $\mathrm{Lip}_{1,d}(X)$ denotes 1-Lipschitz functions $f : X \to \mathbb{R}$ such that $\left|f\left(x_1\right) - f\left(x_2\right)\right| \leq d\left(x_1, x_2\right)$.

It is notable that the 2-Wasserstein metric $W_2\left(\|\cdot\|_2\right)$ (or simply W_2) has a closed-form for Gaussians:

$$W_2\left(\mathcal{N}\left(\mu_i, \Sigma_i\right), \mathcal{N}\left(\mu_j, \Sigma_j\right)\right)^2 = \left\|\mu_i - \mu_j\right\|_2^2 + \left\|\Sigma_i - \Sigma_j\right\|_{\mathcal{F}}^2 \tag{16}$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm. We can observe that for point masses (i.e., $\Sigma_i, \Sigma_j \to 0$), the 2-Wasserstein metric is equivalent to the Euclidean distance between the two points.

Lemma 1. *(p-Wasserstein Inequality Villani et al. [2009]) For any two distributions μ, λ, if $p \leq q$:*

$$W_p(\mu, \lambda) \leq W_q(\mu, \lambda) \tag{17}$$

Lemma 1 shows the inequality between p-Wasserstein.

C Theorems and Proofs

Theorem 1. *Let* $\mathfrak{met}$ *be the space of bounded pseudometrics on* $\mathcal{S}$, $\gamma \in [0,1)$ *and* π *a policy that is continuously improving. Define* $\mathcal{H} : \mathfrak{met} \mapsto \mathfrak{met}$ *by:*

$$\begin{aligned}\mathcal{H}(d,\pi)(s_i,s_j) = |r^{\pi}_{s_i} - r^{\pi}_{s_j}| + \gamma W(d)(\mathcal{P}^{\pi}_{s_i}, \mathcal{P}^{\pi}_{s_j}) \\ + \|I^{\pi}(\cdot \mid s_i, s_{i+1}) - I^{\pi}(\cdot \mid s_j, s_{j+1})\|_1\end{aligned} \tag{18}$$

Then $\mathcal{H}$ *has a least fixed point* $\tilde{d}$ *which is an inverse dynamic bisimulation metric.*

Proof. Let $(X, \preceq)$ be a partial order. An ω-chain of this partial order is an increasing sequence $\{x_n\}$. The partial order is said to be an ω-complete partial order (ω-cpo) if it contains least upper bounds of all ω-chains. It is called an ω-cpo with bottom if it additionally contains a least element, $\perp$, called bottom. A function $f : X \to Y$ between ω-cpos is said to be monotonic if $x \preceq x' \implies f(x) \preceq f(x')$. It is continuous if for every ω-chain $\{x_n\}$, $f\left(\sqcup_{n\in\mathbb{N}} \{x_n\}\right) = \sqcup_{n\in\mathbb{N}} \{f(x_n)\}$. A point $x \in X$ is said to be a prefixed-point of f if $f(x) \preceq x$. It is a fixed-point if $x = f(x)$.

Lemma 2. *Given a fixed policy* $\hat{\pi}$. *Define* $\mathcal{H}^{\hat{\pi}} : \mathfrak{met} \mapsto \mathfrak{met}$ *by* $\mathcal{H}(d,\hat{\pi})(s_i,s_j) = |r^{\hat{\pi}}_{s_i} - r^{\hat{\pi}}_{s_j}| + \gamma W(d)(\mathcal{P}^{\hat{\pi}}_{s_i}, \mathcal{P}^{\hat{\pi}}_{s_j}) + \|I^{\hat{\pi}}(\cdot \mid s_i, s_{i+1}) - I^{\hat{\pi}}(\cdot \mid s_j, s_{j+1})\|_1$, *then* $\mathcal{H}^{\hat{\pi}}$ *has a least fixed point* $d^{\hat{\pi}}$.

Proof. This proof mimics the proof of Theorem 4.5 from Ferns et al. [2004]. We make use of the same pointwise ordering on $\mathfrak{met}$: $d \leq d'$ iff $d(s,t) \leq d'(s,t)$ for all $s,t \in \mathcal{S}$, which gives us an ω-cpo with bottom $\perp$, which is the everywhere-zero metric. Since Lemma 4.4 from Ferns et al. [2004] (wasserstein metric W is continuous) also applies in our definition, it only remains to show that $\mathcal{H}(d,\hat{\pi})$ is continuous:

$$\begin{aligned}\mathcal{H}^{\hat{\pi}}\left(\bigsqcup_{n\in\mathbb{N}} \{x_n\}\right)(s_i,s_j) &= |r^{\hat{\pi}}_{s_i} - r^{\hat{\pi}}_{s_j}| + \gamma W\left(\bigsqcup_{n\in\mathbb{N}} \{x_n\}\right)(\mathcal{P}^{\hat{\pi}}_{s_i}, \mathcal{P}^{\hat{\pi}}_{s_j}) \\ &\quad + \|I^{\hat{\pi}}(\cdot \mid s_i, s_{i+1}) - I^{\hat{\pi}}(\cdot \mid s_j, s_{j+1})\|_1 \\ &= |r^{\hat{\pi}}_{s_i} - r^{\hat{\pi}}_{s_j}| + \gamma \sup_{n\in\mathbb{N}} W(x_n)(\mathcal{P}^{\hat{\pi}}_{s_i}, \mathcal{P}^{\hat{\pi}}_{s_j}) \\ &\quad + \|I^{\hat{\pi}}(\cdot \mid s_i, s_{i+1}) - I^{\hat{\pi}}(\cdot \mid s_j, s_{j+1})\|_1 \\ &\quad \text{by continuity of } W \\ &= \sup_{n\in\mathbb{N}}(|r^{\hat{\pi}}_{s_i} - r^{\hat{\pi}}_{s_j}| + \gamma W(x_n)(\mathcal{P}^{\hat{\pi}}_{s_i}, \mathcal{P}^{\hat{\pi}}_{s_j}) \\ &\quad + \|I^{\hat{\pi}}(\cdot \mid s_i, s_{i+1}) - I^{\hat{\pi}}(\cdot \mid s_j, s_{j+1})\|_1) \\ &= \sup_{n\in\mathbb{N}} \left\{\mathcal{H}^{\hat{\pi}}(x_n)(s_i,s_j)\right\} \\ &= \left(\bigsqcup_{n\in\mathbb{N}} \left\{\mathcal{H}^{\hat{\pi}}(x_n)\right\}\right)(s_i,s_j)\end{aligned} \tag{19}$$

The rest of the proof follows in the same way as in Ferns et al. [2004]. □

Ideally, to prove this theorem we show that $\mathcal{H}$ is monotonically increasing and continuous, and apply Fixed Point Theorem to show the existence of a fixed point that $\mathcal{H}$ converges to. Unfortunately, we can show that $\mathcal{H}$ under π as π monotonically converges to π^* is not also monotonic, unlike the original bisimulation metric setting Ferns et al. [2004] and the policy evaluation setting Castro [2020]. We start the iterates $\mathcal{H}^n$ from bottom $\perp$, denoted as $\mathcal{H}^n(\perp)$. In Ferns et al. [2004] the $\max_{a\in\mathcal{A}}$ can be thought of as learning a policy between every two pairs of states to maximize their distance, and therefore this distance can only stay the same or grow over iterations of $\mathcal{H}$. In Castro [2020], π is fixed, and under a deterministic MDP it can also be shown that distance between states $d_n(s_i,s_j)$ will only expand, not contract as n increases. In the policy iteration setting, however, with π starting from initialization π_0 and getting updated:

$$\pi_k(s) = \arg\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} [r^a_{ss'} + \gamma V^{\pi_{k-1}}(s')] \tag{20}$$

there is no guarantee that the distance between two states $d_{n-1}^{\pi_{k-1}}(s_i, s_j) < d_n^{\pi_k}(s_i, s_j)$ under policy iterations π_{k-1}, π_k and distance metric iterations d_{n-1}, d_n for $k, n \in \mathbb{N}$, which is required for monotonicity.

Instead, we show that using the policy improvement theorem which gives us:

$$V^{\pi_k}(s) \geq V^{\pi_{k-1}}(s), \forall s \in \mathcal{S} \tag{21}$$

π will converge to a fixed point using the Fixed Point Theorem, and taking the result of lemma 2, that $\mathcal{H}^\pi$ has a fixed point for every $\pi \in \Pi$, we can show that a fixed point inverse dynamic bisimulation metric will be found with policy iteration. □

Theorem 2. *(Value difference bound) Given any two states $s_i, s_j \in \mathcal{S}$ in an MDP $\mathcal{M}$, let $V^\pi(s)$ be the value function of policy π, we can get:*

$$|V^\pi(s_i) - V^\pi(s_j)| \leq d_{inv}(s_i, s_j) \tag{22}$$

where d_{inv} is a inverse dynamic bisimulation metric.

Proof. We will use the standard value function update: $V_n^\pi(s) = \mathcal{R}_s^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_s^\pi(s') V_{n-1}^\pi(s')$ with $V_0^\pi \equiv 0$ and our update operator from Theorem 1: $d_n^\pi(s,t) = \mathcal{H}^\pi\left(d_{n-1}^\pi\right)(s,t)$ with $d_0^\pi \equiv 0$, and prove this by induction, showing that for all $n \in \mathbb{N}$ and $s, t \in \mathcal{S}$, $|V_n^\pi(s) - V_n^\pi(t)| \leq d_n^\pi(s,t)$.

The base case holds vacuously: $0 = V_0^\pi(s,t) \leq d_0^\pi(s,t) = 0$, so assume true up to n.

$$\begin{aligned}
\left|V_{n+1}^\pi(s) - V_{n+1}^\pi(t)\right| &= \left|\mathcal{R}_s^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_s^\pi(s') V_n^\pi(s') - \left(\mathcal{R}_t^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_t^\pi(s') V_n^\pi(s')\right)\right| \\
&\leq |\mathcal{R}_s^\pi - \mathcal{R}_t^\pi| + \left|\gamma \sum_{s' \in \mathcal{S}} V_n^\pi(s')\left(\mathcal{P}_s^\pi(s') - \mathcal{P}_t^\pi(s')\right)\right| \\
&\leq |\mathcal{R}_s^\pi - \mathcal{R}_t^\pi| + |\gamma W(d_n^\pi)(\mathcal{P}_s^\pi, \mathcal{P}_t^\pi)| \\
&= |\mathcal{R}_s^\pi - \mathcal{R}_t^\pi| + \gamma W(d_n^\pi)(\mathcal{P}_s^\pi, \mathcal{P}_t^\pi) \text{ since } W(d_n^\pi) \text{ is a metric} \\
&\leq |\mathcal{R}_s^\pi - \mathcal{R}_t^\pi| + \gamma W(d_n^\pi)(\mathcal{P}_s^\pi, \mathcal{P}_t^\pi) + \|I^\pi(\cdot \mid s, s') - I^\pi(\cdot \mid t, t')\|_1 \\
&= \mathcal{H}^\pi(d_n^\pi)(s,t) \\
&= d_{n+1}^\pi(s,t)
\end{aligned} \tag{23}$$

where the second inequality follows from noticing that, by induction, for all $s, t \in \mathcal{S}, V_n^\pi(s) - V_n^\pi(t) \leq d_n(s,t)$, and the third inequality shows that the property as value difference bound holds for both bisimulation metric and inverse dynamic bisimulation metric. □

Theorem 3. *The potential function $d_{inv}(s, s_0)$ is an approximation of absolute value of optimal value function $V^*(s)$.*

Proof. Based on theorem 2, we can get:

$$\begin{aligned}
d_{inv}(s, s_0) &\geqslant |V^\pi(s) - V^\pi(s_0)| \\
&\geqslant |V^\pi(s)| - |V^\pi(s_0)| \\
&= |V^\pi(s)| - C_1
\end{aligned} \tag{24}$$

where C_1 is a constant. Assume that the potential function $d_{inv}(s, s_0)$ is defined under policy π, we use the closed form induction for the 2-Wasserstein metric $W_2(\|\cdot\|_2)$ as Equation (16):

$$\begin{aligned}
\lim_{t \to \infty} d_{inv}(s_t, s_0) - |V^*(s_t)| = \lim_{t \to \infty} [&|r_t^\pi - r_0^\pi| + \gamma \|\mu_t^\pi - \mu_0\|_2^2 \\
&+ \gamma \|\Sigma_t^{(1/2)} - \Sigma_0^{(1/2)}\|_F^2 \\
&+ \gamma \|a_t^\pi - a_0\|_1 - V^*(s_t)]
\end{aligned} \tag{25}$$

The action a belongs to the set of bounded action space $\mathcal{A}$, so the fourth term on the right of Equation (25) is bounded, as the timestep increases, the distribution $\mathcal{P}^{\pi}(\cdot \mid s)$ converges to the transition function $\mathcal{N}(\mu_c, \Sigma_c)$, thus the second term and third term are bounded. So we can get:

$$\begin{aligned}\lim_{t\to\infty} d_{inv}(s_t, s_0) - |V^*(s_t)| &= \lim_{t\to\infty} [|r_t^{\pi}| - |V^*(s_t)| + C_2 \\ &= \lim_{t\to\infty} [|r_t^{\pi}| - |\max_a r_t(s,a)| \\ &\quad - \gamma|\sum_{s_{t+1}\in\mathcal{S}} \mathcal{P}^a_{s_t s_{t+1}} V^*(s_{t+1})|] + C_2 \\ &\leqslant C_2\end{aligned} \tag{26}$$

where C_2 is a constant. Based on equation 24 and equation 26, we can get:

$$\lim_{t\to\infty} |V^*(s_t)| - C_1 \leqslant \lim_{t\to\infty} d_{inv}(s_t, s_0) \leqslant \lim_{t\to\infty} |V^*(s_t)| + C_2 \tag{27}$$

so the potential function $d_{inv}(s, s_0)$ is an approximation of absolute value of optimal value function $V^*(s)$ □

Theorem 4. *Suppose that the shaping reward function $\mathcal{F}$ takes the form of Equantion (1), the optimal value function of the modified MDP $\mathcal{M}'$, the potential function $\Phi(s)$ and the optimal value function of original MDP $\mathcal{M}$ holds the condition that:*

$$V^*_{\mathcal{M}'}(s,a) = V^*_{\mathcal{M}}(s,a) - \Phi(s) \tag{28}$$

Proof. According to the Bellman equation, the optimal value function of the original MDP $\mathcal{M}$ has the following form:

$$V^*_{\mathcal{M}}(s) = \max_a \mathrm{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[\mathcal{R}(s,a,s') + \gamma V^*_{\mathcal{M}}(s')] \tag{29}$$

We can obtain the following by subtracting the potential function from both sides:

$$\begin{aligned}V^*_{\mathcal{M}}(s) - \Phi(s) = \max_a \mathrm{E}_{s'\sim\mathcal{P}(\cdot|s,a)}[&\mathcal{R}(s,a,s') + \gamma\Phi(s') - \Phi(s) \\ &+ \gamma(V^*_{\mathcal{M}}(s') - \Phi(s'))]\end{aligned} \tag{30}$$

Based on Equantion (1), we can get:

$$\begin{aligned}V^*_{\mathcal{M}'}(s) &= \max_a \mathrm{E}_{s'\sim\mathcal{P}(\cdot|s,a)} \left[\mathcal{R}\left(s,a,s'\right) + \mathcal{F}\left(s,a,s'\right) + \gamma V^*_{\mathcal{M}'}\left(s'\right)\right] \\ &= \max_a \mathrm{E}_{s'\sim\mathcal{P}(\cdot|s,a)} \left[\mathcal{R}'\left(s,a,s'\right) + \gamma\hat{V}_{\mathcal{M}'}\left(s'\right)\right]\end{aligned} \tag{31}$$

So the value function of the modified MDP $\mathcal{M}'$ is:

$$V^*_{\mathcal{M}'}(s) \triangleq V^*_{\mathcal{M}}(s) - \Phi(s) \tag{32}$$

□

D Additional Experiments

D.1 Delayed Reward Setting

We provide the full result for the delayed reward setting, where the accumulated reward is given every 10, 20, 30 or 40 steps. Table 2 demonstrates that LIBERTY outperforms other methods in 19 out of 24 delayed reward tasks. This suggests that even with sparse rewards, LIBERTY can achieve effective exploration while maintaining high performance. Since DPBA only receives delayed rewards, its performance decreases as the delay period increases. In contrast, RIDE and RND can generate more exploration for the agent at each step, resulting in better performance than DPBA. Moreover, curiosity-driven methods perform similarly to delayed reward-based methods, despite the lack of explicit rewards. The result provides further evidence that LIBERTY's potential-based exploration bonus encourages efficient exploration in the environment, even in scenarios with only delayed rewards.

Table 2: Full Table of quantitative results comparison between LIBERTY and other baseline methods in different environments of Mujoco with the delayed reward setting.The best and the runner-up results are (**bold**) and (underline)

Methods	Delay = 10					
	HalfCheetah	Hopper	Walker2d	Ant	Humanoid	Swimmer
ICM	1374 ± 368	1258 ± 325	1127 ± 225	-105 ± 43	462 ± 54	27 ± 11
RND	1694 ± 495	1976 ± 458	1405 ± 262	143 ± 17	532 ± 29	32 ± 15
NGU	1180 ± 513	989 ± 262	1275 ± 480	-164 ± 35	413 ± 78	24 ± 12
RIDE	2467 ± 456	1876 ± 431	1651 ± 325	92 ± 31	570 ± 45	65 ± 16
DPBA	1514 ± 365	2103 ± 129	1997 ± 115	**592 ± 67**	518 ± 23	43 ± 17
LIBERTY	**2973 ± 437**	**2479 ± 315**	**2766 ± 487**	292 ± 68	**681 ± 73**	**73 ± 21**
LIBERTY w/o I.D.	1783 ± 412	1676 ± 275	1732 ± 392	131 ± 22	505 ± 37	46 ± 11
Methods	Delay = 20					
	HalfCheetah	Hopper	Walker2d	Ant	Humanoid	Swimmer
ICM	1185 ± 287	1097 ± 275	995 ± 201	-175 ± 23	434 ± 48	23 ± 10
RND	1595 ± 415	1925 ± 401	1379 ± 193	$127 + 12$	519 ± 25	29 ± 12
NGU	1198 ± 492	978 ± 177	1158 ± 375	-177 ± 31	416 ± 57	21 ± 11
RIDE	2285 ± 402	1621 ± 382	1724 ± 307	105 ± 27	509 ± 21	59 ± 13
DPBA	1337 ± 315	1783 ± 89	1543 ± 137	**381 ± 45**	501 ± 18	32 ± 15
LIBERTY	**2619 ± 354**	**2112 ± 208**	**2345 ± 414**	263 ± 55	**617 ± 53**	**67 ± 18**
LIBERTY w/o I.D.	1554 ± 256	1527 ± 312	1447 ± 363	103 ± 19	497 ± 36	37 ± 15
Methods	Delay = 30					
	HalfCheetah	Hopper	Walker2d	Ant	Humanoid	Swimmer
ICM	1017 ± 276	965 ± 213	798 ± 199	-198 ± 25	417 ± 45	19 ± 11
RND	1483 ± 393	1773 ± 391	1038 ± 191	$99 + 13$	501 ± 27	24 ± 11
NGU	1113 ± 435	901 ± 172	1013 ± 366	-181 ± 29	399 ± 45	20 ± 13
RIDE	1973 ± 369	1405 ± 315	1345 ± 305	87 ± 21	487 ± 25	41 ± 15
DPBA	1021 ± 297	1461 ± 91	1179 ± 135	**241 ± 37**	453 ± 18	21 ± 11
LIBERTY	**2273 ± 317**	**1873 ± 228**	**2077 ± 398**	215 ± 48	**587 ± 63**	**52 ± 15**
LIBERTY w/o I.D.	1305 ± 243	1297 ± 238	1197 ± 367	64 ± 21	468 ± 31	25 ± 11
Methods	Delay = 40					
	HalfCheetah	Hopper	Walker2d	Ant	Humanoid	Swimmer
ICM	919 ± 199	857 ± 175	697 ± 172	-213 ± 27	403 ± 34	13 ± 7
RND	1276 ± 387	**1683 ± 338**	968 ± 168	$71 + 15$	483 ± 25	17 ± 11
NGU	1028 ± 405	879 ± 155	997 ± 280	-198 ± 27	387 ± 27	11 ± 6
RIDE	1798 ± 355	1235 ± 269	1025 ± 282	63 ± 18	468 ± 23	**32 ± 11**
DPBA	883 ± 275	1382 ± 85	1016 ± 129	105 ± 31	405 ± 15	9 ± 3
LIBERTY	**2039 ± 315**	1612 ± 215	**1921 ± 372**	**142 ± 45**	**566 ± 35**	31 ± 13
LIBERTY w/o I.D.	1231 ± 253	1213 ± 207	1012 ± 358	58 ± 13	455 ± 27	17 ± 8

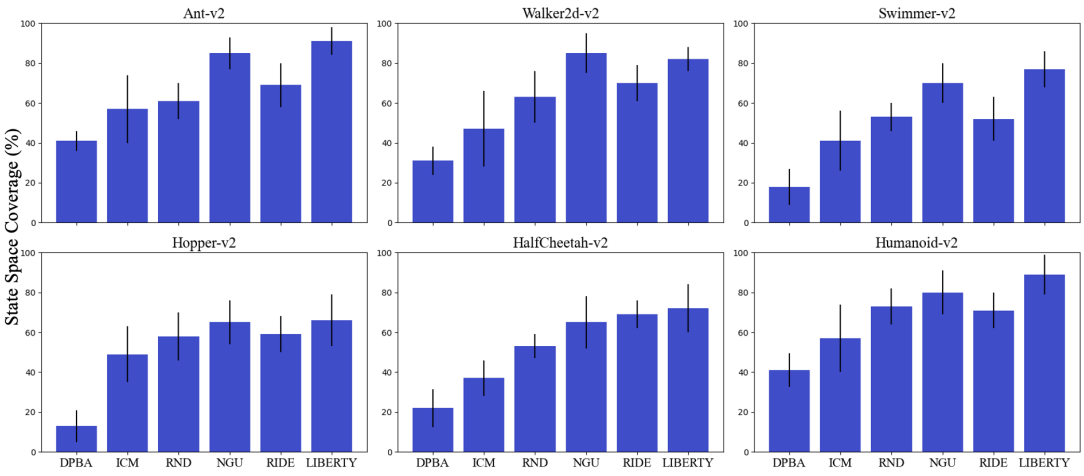


Figure 6: Reward-free exploration results on Ant, Walker2d, Swimmer, Hopper, HalfCheetah and Humanoid. Error bars represent std, deviations over 10 seeds.

D.2 Reward-free Exploration

We discretize the state-space of all six environments in MuJoCo into bins and compare the number of bins explored, in terms of coverage percentage. As Figure 6 shows, LIBERTY demonstrates its robustness across different tasks by achieving the highest number of bins in 5 out of 6 environments. RIDE and NGU also perform well, closely following LIBERTY. On the other hand, RND outperforms ICM in all six cases, while DPBA performs the worst among the evaluated methods.

D.3 Episodic Counts for Continuous Control

For the continuous control experiments, we present an additional comparison against the NGU Badia et al. [2019] and RIDE Raileanu and Rocktäschel [2020], as shown in Table 3, which also employ episodic counts for their intrinsic reward signals. Exact state counts are poorly informative in continuous state spaces, as identical states are rare. Therefore, we adopt episodic counts as pseudo-counts following the procedure presented in NGU for both methods. For NGU, we use inverse-dynamics features to compute pseudo-counts, while for RIDE, we directly use the states. Figure 7 presents the results, including ablations with respect to episodic count modulation. The analysis shows a significant drop in the performance of RIDE in all six environments, while NGU without count consistently underperforms NGU itself. Overall, the episodic count component improves performance for both NGU and RIDE across all environments. It is worth noting that our method eliminates the need for episodic count, which greatly improves scalability.

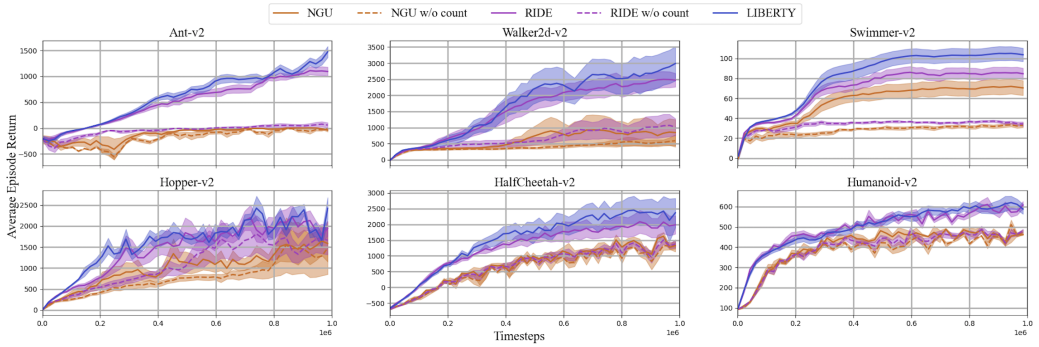


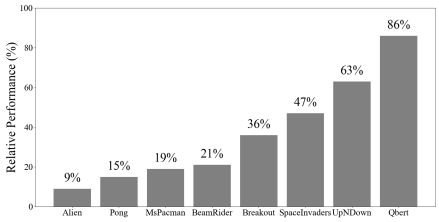
Figure 7: Episodic Counts for Continuous Control. Comparison between LIBERTY, NGU and RIDE, with the latter two being tested both with (original methods) and without (ablations) the episodic-count modulation component. The x-axis represents the number of steps (1e6) in training. The y-axis represents the average episode return over the last 100 training episodes (standard deviations in shade). All of the experiments were run using 10 different seeds.

D.4 Improvements Over Benchmarks

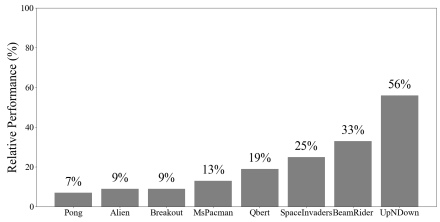
To compare the improvements of LIBERTY over different methods, we select three types of agents: PPO as the baseline agent, DPBA as the PBRS agent, and RIDE as the exploration (via state difference) agent. We ran each agent for 5 separate runs each for 50 million frames on each game for both the baseline agents and LIBERTY-augmented agents. We plotted the best results in Figure 8. The results show that the LIBERTY exploration bonus is effective across different game settings and significantly improves the performance of different types of agents.

D.5 Scale of Shaping Reward

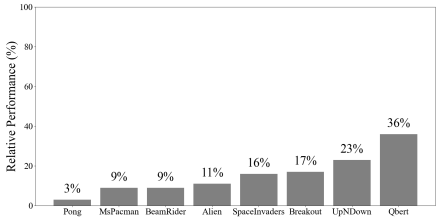
We consider the additive form of reward shaping, which can be defined as $\mathcal{R}' = \mathcal{R} + \mathcal{F}$, where $\mathcal{R}$ is the original reward function, $\mathcal{F}$ is the shaping reward function, and $\mathcal{R}'$ is the modified reward function. The scale of the shaping reward can significantly impact the performance of the trained policy. Therefore, we conducted a study on the scale parameter λ, where the modified reward function can be expressed as $\mathcal{R}' = \mathcal{R} + \lambda\mathcal{F}$. Note that the default setting in our experiments is $\lambda = 1$. To investigate the impact of different scales of shaping rewards on exploration efficiency, we selected $\lambda = 0.01$, $\lambda = 0.1$, $\lambda = 10$ and $\lambda = 100$ as different hyper-parameters to be tested on Atari games.



(a) Improvements of LIBERTY augmented agents over baseline (PPO) agents



(b) Improvements of LIBERTY augmented agents over PBRS (DPBA) agents



(c) Improvements of LIBERTY augmented agents over exploration (RIDE) agents

Figure 8: In all figures, the columns correspond to different games labeled on the x-axes and the y-axes show human score normalized improvements.

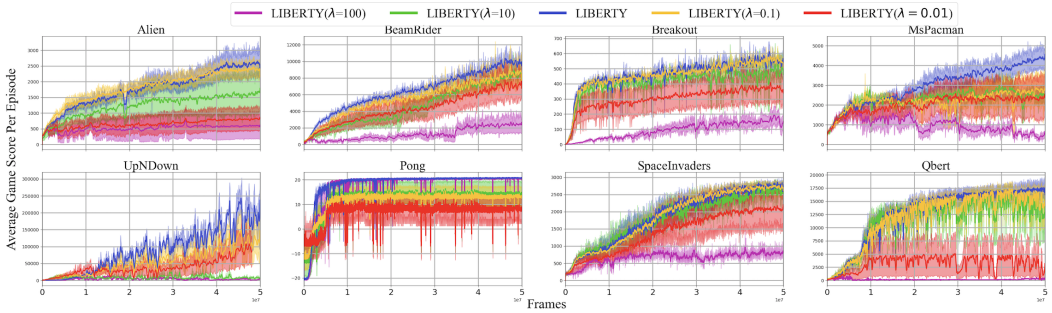


Figure 9: Comparison between LIBERTY with different hyper-parameter λ in the atari games. The x-axis represents the number of frames (1e7) in training. The y-axis represents the average game score per episode over the last 100 training episodes (standard deviations in shade). All of the experiments were run using 10 different seeds.

Figure 9 demonstrates that the performance of LIBERTY with $\lambda = 100$ and $\lambda = 0.01$ significantly drops. On the other hand, the performance of LIBERTY with $\lambda = 0.1$ and $\lambda = 10$ are comparable and almost at the same level as LIBERTY. These results indicate that too much exploration bonus ($\lambda = 100$) or too little exploration bonus ($\lambda = 0.01$) can destroy the algorithm's performance.

D.6 Goal-conditioned Tasks

Goal-conditioned reinforcement learning setting is represented by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, G)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the probabilistic transition model, $G \subset \mathcal{S}$ is the goal space which is a set of assignment of values to states, and $R(s, g) = \mathbb{I}(s = g) \in \mathcal{R}$ is the sparse deterministic reward function that returns 1 only if the state s match the goal g. We evaluate LIBERTY in the robot-arm control environment, including three tasks, `FetchPush`, `FetchPickAndPlace`, `FetchSlide`. As Figure 10 shows, we can see that our method LIBERTY still achieves the best performance in the challenging tasks of goal-conditioned environments compared with other baselines, indicating its scalability and superiority.

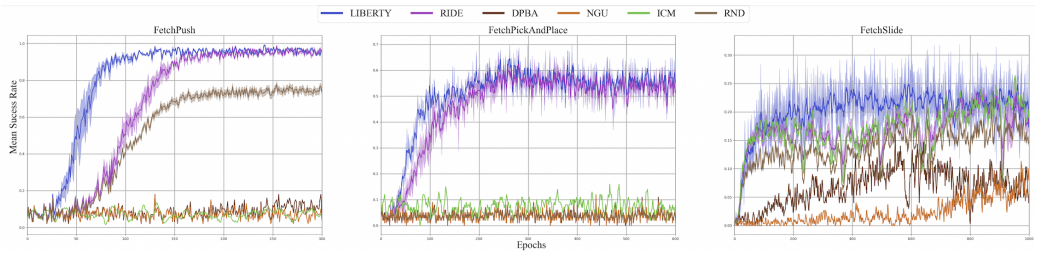


Figure 10: Comparison between LIBERTY with baseline methods in three goal-conditioned tasks, `FetchPush`,`FetchPickAndPlace`,`FetchSlide` . The x-axis represents the number of epochs in training. The y-axis represents the mean success rate (standard deviations in shade). All of the experiments were run using 10 different seeds.

D.7 Discussion and Ablation Study on ℓ_1 and ℓ_2 Norms

We choose ℓ_1 norm to measure the discrepancy between actions output by the inverse dynamic module. The reason behind is that actions are scalar or low dimensional vectors in most RL environments. In discrete-action environments, such as Breakout, the action space is scalar so there is no difference between ℓ_1 and ℓ_2 norm. For the case of continuous-action, such as the 4-dimensional vector action space in the `FetchReach` task where the actions' value range is $[-1, 1]$, since the action's value range is small ,the difference between ℓ_1 and ℓ_2 is almost negligible. And we had carried ablation study on ℓ_1 and ℓ_2 norms on six mujoco continuous control tasks, as Figure 11 shows, the performance between ℓ_1 and ℓ_2 norm is very closed, so we choose the low computational-cost ℓ_1 norm.

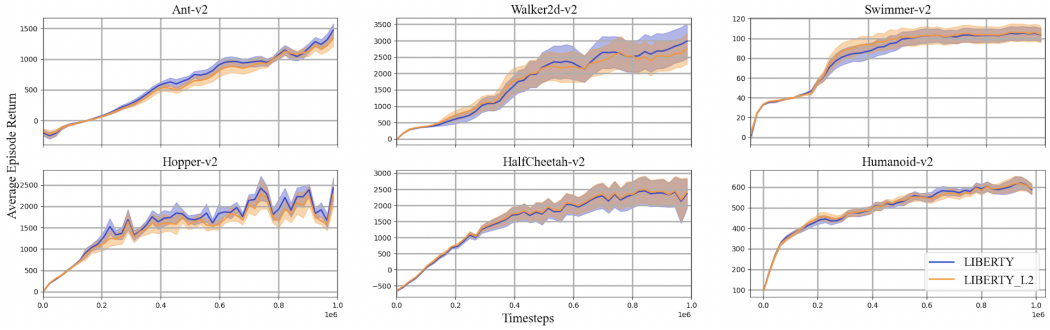


Figure 11: Comparison between LIBERTY and LIBERTY using ℓ_2 norm for inverse dynamic module. The x-axis represents the number of timesteps in training. The y-axis represents the average episode reward(standard deviations in shade). The experiments were run using 20 different seeds.

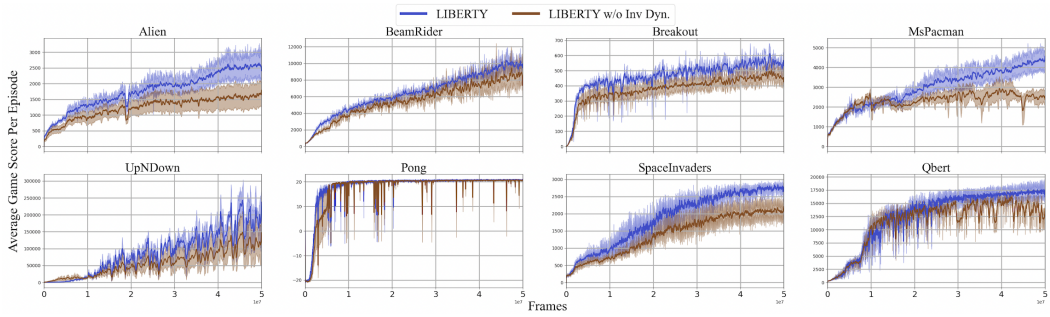


Figure 12: Comparison between LIBERTY and LIBERTY without inverse dynamic module in the atari games. The x-axis represents the number of frames (1e7) in training. The y-axis represents the average game score per episode over the last 100 training episodes (standard deviations in shade). All of the experiments were run using 10 different seeds.

D.8 Ablation study of Inverse Dynamic on Atari Games

To investigate the necessity of inverse dynamic, we denote the variant of LIBERTY as "LIBERTY w/o I.D." which is trained without inverse dynamic. The variant uses the setting of bisimulation metric Castro [2020] only as the potential function. As Figure 12 shows, the performance of LIBERTY on Atari games without the inverse dynamic module had a significant drop in all eight games, which further supports the necessity of inverse dynamic module in promoting efficient exploration.

D.9 Results on SuperMarioBros

We present the results of level 1 in SuperMarioBros in Figure 13. The x-axis represents the number of frames (1e6) in training. The y-axis represents the average episode reward (standard deviations in shade). The experiments were run using 10 different seeds. We use environment wrappers to preprocess each frame to resize the input image, and stack observations to accelerate the training speed. We can see that our method LIBERTY achieves the best performance among all competitive baselines, which demonstrates the superiority and efficiency of our method. The explanation is that the shaping reward based on state discrepancy and value difference can be helpful in the exploration stage of training. Thus, with the assistance of the shaping reward, LIBERTY helps the agent explore novel states at the start of training, and achieves better or comparable performance at the end of training. NGU achieves the second-best result, because it focus on a long-term exploration. The results of RIDE and RND are comparable, followed by ICM. which suggests that the curiosity-driven methods may fail to explore efficiently compared with our method. DPBA perform the worst which means that the shaping reward is not always useful.

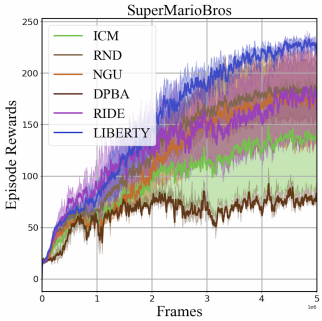


Figure 13: Results of level 1 on SuperMarioBros.

E Implementation Details

E.1 Baseline Method Comparison

We make a comparison between the benchmarked methods across three key aspects. Firstly, we consider whether the method includes an episodic count term in the exploration bonus. Secondly, we evaluate whether it ensures policy invariance from the oringinal MDP. Finally, we examine whether the method relies on prior knowledge. The results are demonstrated in Table 3.

Algorithm	Objective	Episodic Count	Policy Invariance	Prior Knowledge
ICM	$-\log p\left(\phi(s_{t+1}) \mid \phi(s_t), a_t\right)$	✗	✗	✗
RND	$-\log p\left(\phi(s_{t+1}) \mid s_{t+1}\right)$	✗	✗	✗
RIDE	$\|\phi(s_{t+1}) - \phi(s_t)\|_2 / \sqrt{N_{ep}\left(s_{t+1}\right)}$	✓	✗	✗
NGU	$\approx \alpha \cdot b_{\mathrm{RND}}(s) / \sqrt{N_{ep}\left(\phi_{t+1}\right)}$	✓	✗	✗
PBRS	$\gamma\Phi(s_{t+1}) - \Phi(s_t)$	✗	✓	✓
LIBERTY (ours)	$\gamma d_{inv}(s_{t+1}, s_0) - d_{inv}(s_t, s_0)$	✗	✓	✗

Table 3: Summary and comparison of several benchmark methods. $\phi = f(s)$: features encoder; θ : model parameters; $b_{\mathrm{RND}}(s)$: the RND bonus; $N_{ep}(s)$: episodic (pseudo) count of visits to state s; α : normalized coefficient; $\Phi(s)$: potential function defined by prior knowledge; $d_{inv}(s, s_0)$: the inverse dynamic bisimulation metric-based potential function.

E.2 Evaluation Setup

We use the encoder architecture in Zhang et al. [2020] with two more convolutional layers to the convnet trunk as the potential function network. The encoder has kernels of size 3×3 with 32 channels for all the convolutional layers and set stride to 1 everywhere, except of the first convolutional layer, which has stride 2, and interpolate with ReLU activations. Finally, we add tanh nonlinearity to the 50 dimensional output of the fully-connected layer. Each deconvolutional layer has kernels of size 3×3

with 32 channels and stride 1 , except of the last layer, where stride is 2 . We use ReLU activations after each layer. The forward dynamic model are MLPs with two hidden layers with 200 neurons each and ReLU activations. The inverse dynamic module has 3 hidden layers with 64 neurons with ReLU activation functions. The output is a 64 dimensional embedding feature. Adam is chosen as the optimizer with learning rate α=0.0001.

E.3 Algorithm Details

Algorithm 1 LIBERTY

while *not converged* **do**
  for $t = 1$ to MAX_STEP_PER_EPISODE **do**
    Sample action $a_t \sim \pi_\theta\left(a_t \mid s_t\right)$
    Step environment $s_{t+1} \sim p\left(s_{t+1} \mid s_t, a_t\right)$
    Record transition in the buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \left\{s_t, a_t, s_{t+1}, r_{t+1}\right\}$
    Sample transitions batch $B \sim \mathcal{D}$ from the buffer
    Permute batch: $\hat{B} = \text{permute}\left(B\right)$.
    Set shaping intrinsic reward $F\left(s_t, a, s_{t+1}\right) = \gamma d_{inv}\left(s_{t+1}, s_0\right) - d_{inv}\left(s_t, s_0\right)$
    Train policy $\pi(\theta)$ via policy gradient: $\mathbb{E}_B[J(\theta)]$
    Train the potential function d_{inv}: $\mathbb{E}_{B,\hat{B}}[J(\phi)]$ ▷Equation (5)
    Train inverse dynamics: $J(\theta_I)$ ▷Equation (3)
    Train forward dynamics: $J(\eta) = \left(\mathcal{P}(\cdot \mid s_t, a_t; \eta) - s_{t+1}\right)^2$
  end for
end while

E.4 Codebase Used

Our codebase was built atop the following codebases:

- The official NGU codebase (including the implementation of ICM and RND): https://github.com/opendilab/DI-engine
- The official RIDE codebase: https://github.com/facebookresearch/impact-driven-exploration

E.4.1 Hyperparameters of Benchmarks

Algorithm	Hyperparameters
Base Learner (PPO)	policy module: three 64 -unit FC hidden layers, relu activation threshold of probability ratio clipping (ϵ) : 0.2 update timesteps: 1e6 number of epoches per update: 50 number of minibatches: 4 batch size: 1024 GAE parameter (λ) : 0.95 optimizer: Adam learning rate: 10^{-4} policy gradient clip norm: 1.0 discount rate (γ) : 0.999
DPBA	potential network: two 64-unit FC layers, tanh activation optimizer: Adam learning rate of potential network: 5×10^{-4} potential network gradient clip norm: 10.0
ICM	forward model loss coefficient: 1.0 inverse model loss coefficient: 0.1 entropy cost: 0.005 intrinsic reward coefficient β: 0.01
RND	proportion of experience used for training predictor: 0.05 predictor Model updates per PPO epoch: 3 entropy cost: 0.005 intrinsic reward coefficient β: 0.01
NGU	episodic memory capacity: 2000 action prediction network filter sizes: (3,3) action prediction network filter strides: (1, 1) RND clipping factor L: 3 intrinsic reward coefficient β: 0.1
RIDE	forward model loss coefficient: 1.0 inverse model loss coefficient: 0.1 entropy cost: 0.0005 intrinsic reward coefficient β: 0.1

Table 4: The hyperparameters of the tested benchmark algorithms in the MuJoCo experiment

Algorithm	Hyperparameters
Base Learner (PPO)	policy network: two CNN with 3 convolutional layers and 1 FC layer with 512 hidden units the 1st convolutional layer has thirty-two 8 × 8 filters with stride 4 the 2nd convolutional layer has sixty-four 4 × 4 filters with stride 2 The 3rd convolutional layer has sixty-four 3 × 3 filters with stride 1 threshold of probability ratio clipping $(\epsilon) : 0.5$ update timesteps: 5e7 number of epoches per update: 50 number of minibatches: 16 batch size: 2048 GAE parameter $(\lambda) : 0.95$ optimizer: Adam learning rate: 7×10^{-4} policy gradient clip norm: 0.9 discount rate $(\gamma) : 0.999$
DPBA	potential network: 32-unit FC layer + 16-unit FC layer tanh activation optimizer: Adam learning rate of potential network: 5×10^{-4} potential network gradient clip norm: 1.0
ICM	forward model loss coefficient: 0.2 inverse model loss coefficient: 0.08 entropy cost: 10^{-4} intrinsic reward coefficient β: 0.005
RND	proportion of experience used for training predictor: 0.25 predictor Model updates per PPO epoch: 6 entropy cost: 0.005 intrinsic reward coefficient β: 0.1
NGU	episodic memory capacity: 5000 action prediction network filter sizes: (3,3) action prediction network filter strides: (1, 1) RND clipping factor L: 5 intrinsic reward coefficient β: 0.01
RIDE	forward model loss coefficient: 0.5 inverse model loss coefficient: 0.8 entropy cost: 0.0005 intrinsic reward coefficient β: 0.1

Table 5: The hyperparameters of the tested benchmark algorithms in the Atari experiment