BRIDGING HIERARCHICAL REINFORCEMENT LEARNING VIA SUBGOAL ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical reinforcement learning (HRL) decomposes complex tasks into high-level subgoal planning and low-level execution, enabling improved performance in long-horizon settings. However, HRL does not consistently outperform flat policies because imperfect subgoal design can cause misalignment between planning and control, leading to information loss and suboptimal behavior, particularly when the task structure is not well aligned with hierarchical decomposition. To address this issue, we propose BRIDGE, a bidirectional consistency regularization framework to achieve subgoal alignment in HRL, enhancing coordination between hierarchical levels. Our method introduces Reachability Regularization to constrain high-level subgoals based on low-level reachability, and Mutual Information Regularization to enforce dependence between subgoals and state–action trajectories, strengthening both top-down feasibility and bottom-up consistency. Extensive experiments across offline goal-conditioned manipulation and locomotion tasks show that our approach improves HRL performance over existing baselines, particularly in scenarios where conventional hierarchical methods suffer from task–hierarchy mismatch.

1 INTRODUCTION

Many real-world decision-making problems (Mnih et al., 2015; Schulman et al., 2015; Gu et al., 2017; Mirowski et al., 2016; Zhu et al., 2017; Chi et al., 2025; Thrun & Schwartz, 1994; Vezhnevets et al., 2016) require long-horizon reasoning and coordination across multiple levels of abstraction. Hierarchical reinforcement learning (HRL) (Dayan & Hinton, 1992; Schmidhuber & Wahnsiedler, 1992; Nachum et al., 2018; Pateria et al., 2021) provides a natural framework by decomposing complex tasks into high-level planning and low-level execution. With temporal abstraction and subgoal-conditioned control, HRL has shown strong performance in long-horizon planning (Nair & Finn, 2019), sparse-reward exploration (Huang et al., 2024; Shan et al., 2026b; Riedmiller et al., 2018), and complex task decomposition (Diuk et al., 2013; Marzari et al., 2021). However, HRL does not consistently outperform flat policies. When hierarchical structure is unnecessary or poorly aligned with the task dynamics, it can introduce additional optimization difficulties, allowing single-level policies to perform better.

One key reason for this underperformance is that HRL's structural complexity can be unnecessary or even harmful when tasks lack a natural hierarchical structure (Nachum et al., 2019). HRL relies on high-level subgoals as intermediate abstractions for low-level control, but in tasks with simple dynamics or direct state–action mappings, these abstractions may provide limited benefit while introducing modeling errors or information loss. As a result, imperfect subgoals can disturb the original optimal policy and lead to worse performance than flat methods. This indicates that the effectiveness of HRL critically depends on the alignment between high-level subgoal generation and low-level execution (Wang et al., 2024; 2023).

Building on this insight, we propose **B**idirectional **R**egular**I**zation for Couple**D** Sub**G**oal-align**E**d HRL (**BRIDGE**) by coordinating high-level subgoal planning with low-level execution. Our method introduces two complementary terms: *Reachability Regularization* and *Mutual Information Regularization*. The former provides a top-down constraint by encouraging the high-level policy to generate low-level reachable subgoals, thereby reducing infeasible planning. The latter imposes a bottom-up inductive bias by maximizing the dependency between subgoals and state–action trajectories, en-

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

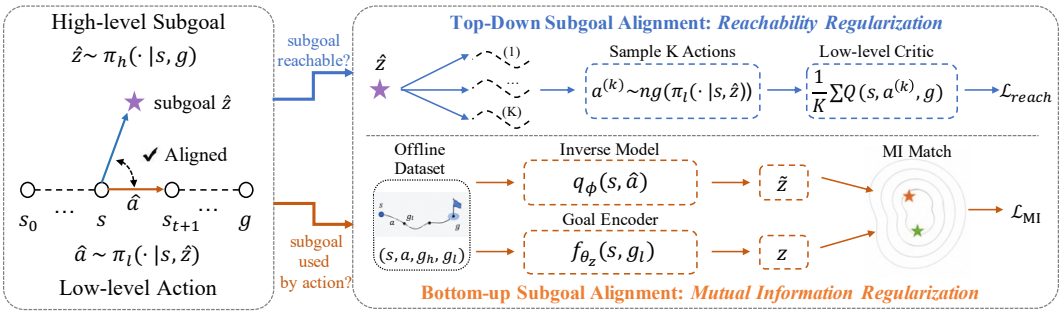

Figure 1: Overview of BRIDGE. Reachability regularization aligns high-level subgoals with low-level feasibility, while mutual information regularization ensures that low-level actions remain conditioned on the assigned subgoals.

suring that low-level behaviors remain subgoal-conditioned and avoiding subgoal-agnostic collapse. Together, they strengthen hierarchical coupling and improve policy consistency.

We conduct a comprehensive offline RL analysis of our method and existing approaches from multiple perspectives, including critic design, scalability, inference latency, and qualitative behaviors. More importantly, this analysis offers a new perspective for evaluating robot simulation methods, showing that our method mitigates high–low policy misalignment by explicitly strengthening hierarchical coupling. Experiments on offline goal-conditioned tasks demonstrate improvements in HRL performance, especially when conventional hierarchical policies suffer from task–hierarchy mismatch. Our main contributions are as follows:

- We conduct a systematic and comprehensive analysis of HRL across diverse environments to explore when and why hierarchical methods succeed or fail, and revealing key factors that affect their performance.
- We propose BRIDGE, a framework that revisits subgoal alignment by strengthening the coupling between high-level and low-level policies, thereby mitigating cross-level misalignment in hierarchical decision-making.
- We provide extensive empirical evaluations across diverse offline goal-conditioned locomotion and manipulation tasks, demonstrating that our method improves HRL performance over existing approaches, especially under task–hierarchy mismatch.

2 Related Work

Goal-conditioned Hierarchical Reinforcement Learning. Goal-conditioned hierarchical reinforcement learning (HRL) extends goal-conditioned policy learning to long-horizon decision-making by decomposing a target goal into intermediate subgoals and executable low-level behaviors (Dietterich, 2000; Daniel et al., 2016; Kulkarni et al., 2016; Peng et al., 2017; Srinivas et al., 2016). Instead of directly selecting primitive actions, the high-level policy proposes subgoals, while the low-level policy executes actions to reach them over shorter horizons. Rooted in temporal abstraction, such as the options framework (Sutton et al., 1999; Precup, 2000; Dietterich et al., 1998; Bagaria et al., 2023) and feudal reinforcement learning (Dayan & Hinton, 1992; Vezhnevets et al., 2017), goal-conditioned HRL has been widely applied to sparse-reward robotic control (Nachum et al., 2018; Levy et al., 2017), navigation (Wöhlke et al., 2021; Eysenbach et al., 2019), and manipulation (Gupta et al., 2019; Lynch et al., 2020). By introducing subgoals as intermediate representations, HRL can facilitate exploration and alleviate long-horizon credit assignment. However, its effectiveness largely depends on the quality of learned subgoals, since unreachable or task-misaligned subgoals may introduce infeasible planning and suboptimal behavior.

Subgoal Reliability and Cross-Level Alignment. Recent studies have increasingly focused on improving the reliability of hierarchical subgoals and the coordination between different policy levels. Guider (Shin & Kim, 2023) constrains offline subgoal generation with learned priors to reduce distribution shift and improve low-level reachability, while BrHPO (Luo et al., 2024) explicitly models bidirectional reachability and encourages mutually responsive high- and low-level policies. Other recent methods further improve subgoal reliability by restricting subgoal selection to

reachable regions or mitigating unreliable value-based decisions (Shan et al., 2026a; Hwang et al., 2026). Beyond reachability, temporal consistency has also been studied through timed subgoals and temporal-distance-aware subgoal generation (Gürtler et al., 2021; Park et al., 2025c). In offline settings, trajectory-stitching approaches such as GAS (Baek et al., 2025) and DiffStitch (Li et al., 2024) improve long-horizon composition by connecting useful segments across different trajectories. BRIDGE complements these directions by directly targeting *cross-level alignment* in hierarchical policy learning. Unlike BrHPO, BRIDGE's reachability regularization uses task-conditioned low-level critic feedback to discourage poorly executable subgoals, while mutual-information regularization encourages low-level actions to remain responsive to assigned subgoals. Together, they improve both high-level subgoal executability and low-level subgoal adherence.

3 PRELIMINARIES

Goal-conditioned HRL. We formulate the task as a goal-conditioned Markov Decision Process (MDP) (Sutton et al., 1998) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, P, r, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{G}$ are the state, action, and goal spaces, $P(s'|s,a)$ is the dynamics, $r(s,a,g)$ is the reward, and γ is the discount factor (Kaelbling, 1993). Given a goal g, the objective is to learn a policy $\pi(a|s,g)$ that maximizes $\mathcal{J}(\pi) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g)]$.

In a two-level hierarchy, the high-level policy $\pi_h(z|s,g)$ predicts a subgoal $z \in \mathcal{Z}$, while the low-level policy $\pi_l(a|s,z)$ executes primitive actions conditioned on z. At the k-th high-level step, $z_k \sim \pi_h(\cdot|s_{kc}, g)$ is sampled and then followed by the low-level policy for c steps, i.e., $a_t \sim \pi_l(\cdot|s_t, z_k)$ for $t \in [kc, (k+1)c - 1]$. Thus, HRL decomposes goal reaching into high-level subgoal planning and low-level subgoal-conditioned control.

Conditional Flow Policies. We parameterize both the high-level and low-level actors using conditional flow matching (Lipman et al., 2023; Liu et al., 2023). Let x_1 denote a data subgoal for the high-level policy or a data action for the low-level policy, with conditioning input $u = (s, g)$ or $u = (s, z)$, respectively. Given $x_0 \sim \mathcal{N}(0, I)$ and $t \sim \mathcal{U}[0, 1]$, we construct

$$x_t = (1 - t)x_0 + tx_1, \tag{1}$$

and train a conditional velocity field $v_\theta(x_t, u, t)$ with the flow-matching objective

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\|v_\theta(x_t, u, t) - (x_1 - x_0)\|_2^2\right]. \tag{2}$$

Following flow-based policy learning in offline RL (Park et al., 2025b), we additionally weight training samples by their offline advantage estimates. At inference, a subgoal or action is generated by integrating

$$\dot{x}_t = v_\theta(x_t, u, t) \tag{3}$$

from x_0 to x_1.

Goal-Conditioned Value Functions. The offline goal-conditioned backbone learns a value function $V(s,g)$ and a low-level critic $Q(s,a,g)$. Their corresponding advantage estimates are used to weight the actor objectives, while the target critic provides the task-conditioned value signal for reachability regularization (Sec. 4.1). Beyond their conventional role in policy evaluation, Q-functions can thus provide a value-based signal for alignment regularization, encouraging high-level subgoals to remain consistent with what the low-level policy can effectively execute.

4 METHODS

In this section, we present BRIDGE, a bidirectional subgoal alignment framework for enhancing HRL performance. Our key idea is that HRL's performance depends on the consistency between high-level planning and low-level execution. Misaligned subgoals may be unreachable for the low-level policy or ignored during execution, leading to infeasible planning or weakened hierarchy. To address this, BRIDGE introduces two complementary regularizers: *reachability regularization* for low-level feasibility and *mutual information regularization* for subgoal-dependent behavior.

Given a hierarchical policy with a high-level policy $\pi_h(z \mid s, g)$ and a low-level policy $\pi_l(a \mid s, z)$, the high level generates subgoals at a slower temporal scale, while the low level executes subgoal-conditioned primitive actions. We improve HRL with two alignment terms:

$$\mathcal{J}_{\mathrm{BRIDGE}} = \mathcal{J} - \lambda_r \mathcal{L}\mathrm{reach} - \lambda_{\mathrm{mi}} \mathcal{L}_{\mathrm{MI}}, \tag{4}$$

where $\mathcal{L}_{\text{reach}}$ discourages low-level-unreachable subgoals, and $\mathcal{L}_{\text{MI}}$ promotes subgoal-dependent low-level behaviors. The weights λ_r and λ_{mi} control their relative strengths.

4.1 Reachability Regularization for Top-Down Subgoal Alignment

A key challenge in HRL is that the high-level policy may propose subgoals that are poorly matched to the current execution capability of the low-level controller, resulting in ineffective coordination. To address this issue, we introduce *Reachability Regularization*, a top-down alignment mechanism that biases high-level subgoal generation toward subgoals whose induced low-level behavior is both executable and useful for the final task.

Given the current state s and task goal g, the high-level policy generates a subgoal $\hat{z} \sim \pi_h(\cdot \mid s, g)$. Although π_h is parameterized by conditional flow matching, the proposed regularization is agnostic to the specific actor parameterization and directly operates on the generated subgoal.

We use the pretrained low-level actor as a differentiable executor. Given $\hat{z}$, we sample K actions $a^{(k)} \sim \pi_l(\cdot \mid s, \hat{z})$ and evaluate them with the target critic $Q(s, a^{(k)}, g)$. Here, Q measures the task-conditioned value induced by $\hat{z}$ toward the final goal g, serving as a *task-aware executability proxy*. The objective is

$$\mathcal{L}_{\text{reach}} = -\mathbb{E}_{s,g,\hat{z}}\left[\frac{1}{K}\sum_{k=1}^{K} Q(s, a^{(k)}, g)\right], \quad \hat{z} \sim \pi_h(\cdot \mid s, g), ; a^{(k)} \sim \pi_l(\cdot \mid s, \hat{z}). \tag{5}$$

The low-level actor and critic are frozen, while gradients flow through $\hat{z} \to a^{(k)} \to Q(s, a^{(k)}, g)$ to update only the high-level actor. Minimizing $\mathcal{L}_{\text{reach}}$ therefore favors subgoals that induce higher-value low-level actions, aligning high-level planning with the executor's current capability.

4.2 Mutual Information Regularization for Bottom-Up Subgoal Alignment

While reachability regularization aligns high-level subgoals with low-level executability, the reverse mismatch can also arise: the low-level policy may respond weakly to its assigned subgoal. We therefore introduce *Mutual Information Regularization* to strengthen the dependence of low-level actions on subgoals.

Given s, z, and $a \sim \pi_l(\cdot \mid s, z)$, we maximize the conditional mutual information $I(z; a \mid s)$. If the controller effectively uses z, the subgoal should be recoverable from (s, a). With a variational inverse model $q_\phi(z \mid s, a)$,

$$I(z; a \mid s) = H(z \mid s) - H(z \mid s, a) \geq \mathbb{E}_{s,z,a}\left[\log q_\phi(z \mid s, a)\right] + H(z \mid s). \tag{6}$$

We construct

$$z = \text{sg}(f_{\theta_z}(s, g_l)), \qquad (s, g_l) \sim \mathcal{D}, \tag{7}$$

and sample a from the *current* low-level policy rather than using offline actions, so the regularizer directly measures its current dependence on z. Assuming a fixed-variance isotropic Gaussian posterior, maximizing the variational bound reduces to

$$\mathcal{L}_{\text{MI}} = \mathbb{E}_{(s,g_l)\sim\mathcal{D},\, z=\text{sg}(f_{\theta_z}(s,g_l)),\, a\sim\pi_l(\cdot|s,z)}\left[\|q_\phi(s, a) - z\|_2^2\right]. \tag{8}$$

Gradients update both the inverse model and low-level actor through a, while stop-gradient on z prevents trivial collapse. Minimizing $\mathcal{L}_{\text{MI}}$ therefore encourages distinct subgoals to induce distinguishable actions, strengthening bottom-up subgoal–action alignment.

5 Experiments

Our experiments evaluate whether our method generalizes across tasks with varying compatibility with hierarchical decomposition. Specifically, we study: 1) its overall performance on diverse tasks (Sec. 5.2); 2) its performance when standard HRL suffers from poor decomposition (Sec. 5.3); and 3) its effects on hierarchical learning through ablations, efficiency and scalability analysis, and behavior visualization (Sec. 5.4).

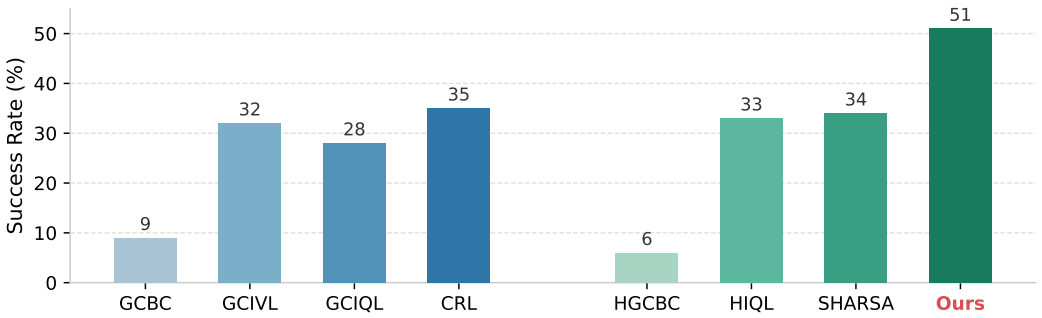

Figure 2: Average success rates over 55 tasks from 11 environments. Blue bars denote flat methods, and green bars denote hierarchical methods. Our method achieves the best overall performance.

5.1 Experimental Setup

Tasks and datasets. We evaluate the performance of different RL algorithms on OGBench (Park et al., 2025a), which includes long-horizon goal-conditioned robot navigation and manipulation tasks. Our evaluation covers 11 environments and 55 tasks, spanning tasks with less explicit hierarchical structure, such as `pointmaze-{medium,large}-navigate` and `scene-play`, and hierarchy-compatible tasks, including `antmaze-{medium,large}-navigate`, `humanoidmaze-{medium,large}-navigate`, `cube-{single,double}-play`, and `puzzle-{4×4,4×5}`. In addition, we further evaluate our method on the D4RL (Fu et al., 2020) manipulation benchmarks, including `pen`, `door`, `relocate`, and `hammer`, to examine its effectiveness beyond OGBench. Details of the environments are shown in Appendix G.

Baselines. We compare with flat goal-conditioned methods, including GCBC (Lynch et al., 2020; Ghosh et al., 2019), GCIQL (Kostrikov et al., 2021), GCIVL (Kostrikov et al., 2021), and CRL (Eysenbach et al., 2022), and hierarchical methods, including HGCBC (Le et al., 2018; Lynch et al., 2020), HIQL (Park et al., 2023), and SHARSA (Park et al., 2026). These baselines cover imitation learning, Q-learning, and contrastive representation learning, allowing us to evaluate whether hierarchical decomposition improves over direct goal-conditioned tasks and whether subgoal alignment further enhances coordination between high-level planning and low-level execution. In particular, SHARSA serves as a strong recent HRL baseline for assessing hierarchical performance. Detailed descriptions of these methods are provided in Appendix H, with comparisons to additional hierarchical baselines reported in Appendix C.

5.2 Main Results

Figure 2 reports the average success rate over 55 tasks from 11 environments, enabling a broad evaluation of performance across diverse environments and task goals. Our method BRIDGE achieves the best performance with a success rate of 51%, demonstrating that subgoal alignment improves HRL performance across diverse goal-conditioned tasks. It outperforms the strongest flat baseline, CRL, by 16 percentage points (51% vs. 35%), and improves over the strongest HRL baseline, SHARSA, by 17 percentage points (51% vs. 34%). These results suggest that bidirectional subgoal alignment improves coordination between high-level subgoal generation and low-level execution, leading to better performance across diverse goal-conditioned tasks.

5.3 Task-Specific Results

To further analyze task-level behavior, Table 1 reports success rates on tasks with different compatibility to hierarchical decomposition. `pointmaze-large-navigate` and `scene-play` have less explicit hierarchical structure, where existing HRL baselines often struggle against flat methods, while `puzzle-4×4` requires multi-step reasoning and sequential execution, making it more suitable for hierarchy. Our method performs better across these settings, achieving the best overall success rate on `pointmaze-large-navigate` (63%), matching the strongest flat baseline on `scene-play` (54%) while outperforming HRL baselines, and reaching 85% on `puzzle-4×4` with stable task-level performance. On D4RL manipulation tasks, it achieves the best `pen` result (89%) and overall success rate (86%), while remaining competitive on other manipulation tasks.

Table 1: Task-level success rates across tasks with varying compatibility to hierarchical decomposition. Our method performs better across both less-structured and naturally hierarchical tasks.

Success Rate (%)		Flat				Hierarchy			
		GCBC	GCIVL	GGIQL	CRL	HGCBC	HIQL	SHARSA	**Ours**
Pointmaze-Large -Navigate	Task1	62±12	**98**±4	70±9	98±6	4±3	66±8	0±0	**99**±2
	Task2	0±0	0±0	0±0	56±15	0±0	0±0	0±0	0±0
	Task3	12±5	**100**±0	20±7	34±7	0±0	82±5	0±0	**100**±0
	Task4	0±0	0±0	0±0	16±2	0±0	8±2	0±0	**90**±3
	Task5	34±10	**100**±0	62±3	52±11	0±0	0±0	0±0	26±8
	Overall	22±3	60±1	30±3	51±4	1±1	31±3	0±1	**63**±2
Scene-play	Task1	20±7	82±6	**96**±3	44±7	8±4	46±12	82±6	86±5
	Task2	0±0	60±13	**96**±3	14±4	28±9	30±10	6±3	66±12
	Task3	2±2	60±14	78±8	23±9	0±0	12±5	40±12	**80**±7
	Task4	4±3	8±4	0±0	8±3	0±1	**16**±6	0±0	**16**±6
	Task5	0±0	0±0	0±0	1±1	0±0	10±4	0±0	**20**±7
	Overall	5±3	42±6	**54**±2	18±2	7±4	23±8	26±3	**54**±4
Puzzle-4*4	Task1	2±2	10±4	20±7	0±0	6±3	4±3	**88**±5	**88**±5
	Task2	0±0	10±4	0±0	0±0	8±4	2±2	74±8	**84**±6
	Task3	0±0	4±3	52±13	0±0	2±2	4±3	**90**±4	82±6
	Task4	0±0	6±3	30±10	0±0	0±0	0±0	80±7	**88**±2
	Task5	0±0	8±4	6±3	0±0	2±2	0±0	**94**±3	82±6
	Overall	0±0	8±4	22±4	0±0	4±3	2±2	**85**±2	**85**±4
D4RL	Pen	48±7	50±5	39±6	6±4	54±6	80±5	58±2	**89**±1
	Relocate	84±6	74±4	8±2	1±4	86±4	86±6	**94**±4	86±4
	Door	**93**±2	92±3	85±5	86±4	82±3	87±3	87±7	87±3
	Hammer	73±7	81±4	39±6	30±5	**96**±4	81±4	86±4	81±4
	Overall	75±3	74±1	43±4	31±4	80±2	84±5	81±3	**86**±2

These results show that subgoal alignment preserves hierarchical benefits in naturally decomposable tasks and improves HRL robustness when task structure is less compatible with hierarchy.

5.4 Ablation Studies and Analysis

Q1: Is value function enough for hierarchical alignment?

As described in Sec. 3, Q-learning estimates expected returns to provide value feedback. Table 2 separates the effects of Q critics and subgoal alignment. Here, *Flat* is a single-level variant, while *HRL* removes our alignment regularization. Q critics improve Flat from 9% to 28% and HRL from 6% to 34%/39%, confirming the benefit of value feedback. However, standard HRL still underperforms BRIDGE, showing that Q-learning alone cannot resolve hierarchical misalignment. With a low-level Q critic, BRIDGE achieves the best success rate of 51%, suggesting that subgoal alignment complements Q learning by promoting executable subgoals. Bi-level Q critics help on `puzzle-4×4` but hurt `pointmaze-navigation`, indicating instability when hierarchy is less natural.

Table 2: Ablation of Q-learning feedback and subgoal alignment. Results are success rates (%).

Env.	Flat		HRL			BRIDGE	
	w/o Q	w/ Q	w/o Q	Low-Q	Bi-Q	Low-Q	Bi-Q
Scene-play	5	54	7	23	50	54	**56**
Puzzle-4 × 4	0	22	4	2	91	85	**96**
Pointmaze	22	30	1	31	19	**63**	23
Overall 55 Tasks	9	28	6	34	39	**51**	42

Table 3: Component ablation. Results are success rates (%).

Variant	Scene	Puzzle	PointMaze-L	Average
Base	48	80	50	59.3
+ Reachability	50	80	50	60.0
+ MI	48	84	52	61.3
BRIDGE	**54**	**85**	**63**	**67.3**

Q2: How does the alignment mechanism affect BRIDGE?

We first isolate the contribution of the two alignment regularizers. As shown in Table 3, either reachability or mutual information (MI) regularization improves the base HRL policy, while combining them yields the best overall performance. This supports their complementary roles: reachability aligns high-level subgoals with low-level executability, while MI keeps low-level actions responsive to the selected subgoals.

We further examine sensitivity to λ_r and λ_{mi} in Figure 3, varying one weight at a time while fixing the other to 0.1. Both regularizers perform best at moderate strengths: overly large λ_r can restrict high-level planning, while excessive λ_{mi} can over-constrain low-level learning. Performance peaks around $\lambda_r = 1.7$ and $\lambda_{\text{mi}} = 0.5$, further supporting the benefit of balanced bidirectional alignment.

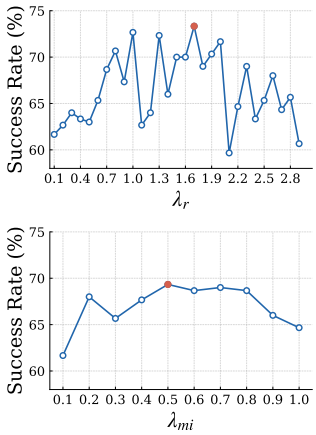

Figure 3: Sensitivity analysis of alignment weights.

Q3: Does BRIDGE run efficiently?

Table 4 compares the training and inference latency of different methods over 100 random run steps. Although our alignment regularization introduces additional training overhead, the cost remains practical for HRL. Moreover, BRIDGE has nearly identical inference latency to its unregularized counterpart (93.75 ms vs. 93.95 ms), indicating that the proposed regularizers improve policy learning without introducing additional inference-time overhead.

Table 4: Training and inference latency of different methods.

Latency (ms)	Flat		Hierarchy				
	GCBC	GCIQL	HGCBC	HIQL	SHARSA	Ours w/o Regulation	Ours
Training Latency	11.69	69.09	28.16	102.45	197.16	97.58	194.29
Inference Latency	4.09	4.09	8.26	8.12	1761.11	93.95	93.75

Q4: Can BRIDGE scale with larger data?

To evaluate scalability, we further test our method on the more challenging `puzzle-4×6` task in OGBench, which requires longer-horizon reasoning and more complex sequential execution. As shown in Figure 4, our method achieves steady performance gains as the dataset size increases, showing stronger scalability than other methods in this longer-horizon setting. These results suggest that subgoal alignment helps HRL better exploit large-scale data and extend to more complex sequential decision-making tasks.

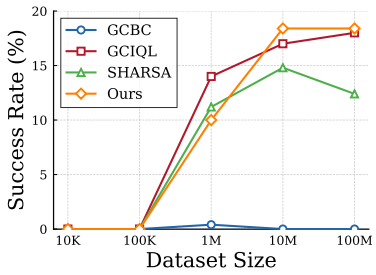

Figure 4: Scalability results on `puzzle-4×6` with varying dataset sizes.

Q5: Is BRIDGE robust under perturbations?

We evaluate BRIDGE under observation noise, actuation noise, and gravity shifts across Scene-play, Puzzle-4×4, and PointMaze-Large. As shown in Table 5, BRIDGE remains stable under moderate perturbations. Observation noise causes a larger degradation in Scene-play and Puzzle-4×4 at $\sigma = 0.1$, while actuation noise and gravity shifts have relatively limited impact. PointMaze-Large is particularly robust, with performance remaining close to the nominal setting across all perturbations. These results indicate that BRIDGE tolerates moderate execution and dynamics uncertainty without requiring explicit robustness training.

Table 5: Evaluation under perturbations: success rate (%).

Environment	Base	Obs. Noise		Act. Noise		Gravity	
		$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.05$	$\sigma = 0.1$	$0.75 \times G$	$1.5 \times G$
Scene-play	53.6	51.6	43.6	53.2	52.8	50.0	49.2
Puzzle-4×4	85.2	83.1	76.9	82.4	76.0	82.0	86.0
PointMaze-Large	63.4	63.8	65.4	63.0	63.8	63.8	65.8

Q6: Do trajectories show better alignment?

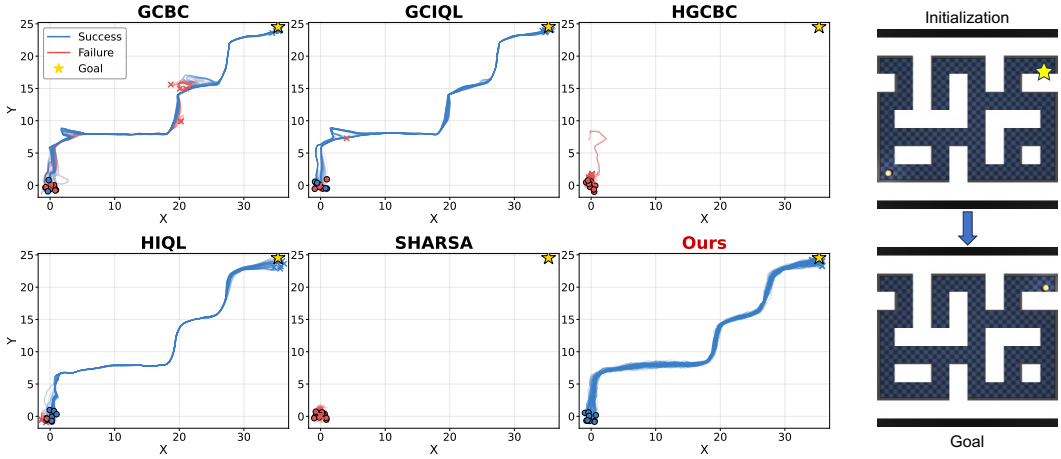

Figure 5: Trajectory visualization on `pointmaze-navigation` task. Blue and red curves denote successful and failed trajectories, respectively, and yellow stars indicate goals. Our method produces smoother and more direct successful trajectories with fewer detours and oscillations, while baselines often get stuck near the start or fail around key turning points.

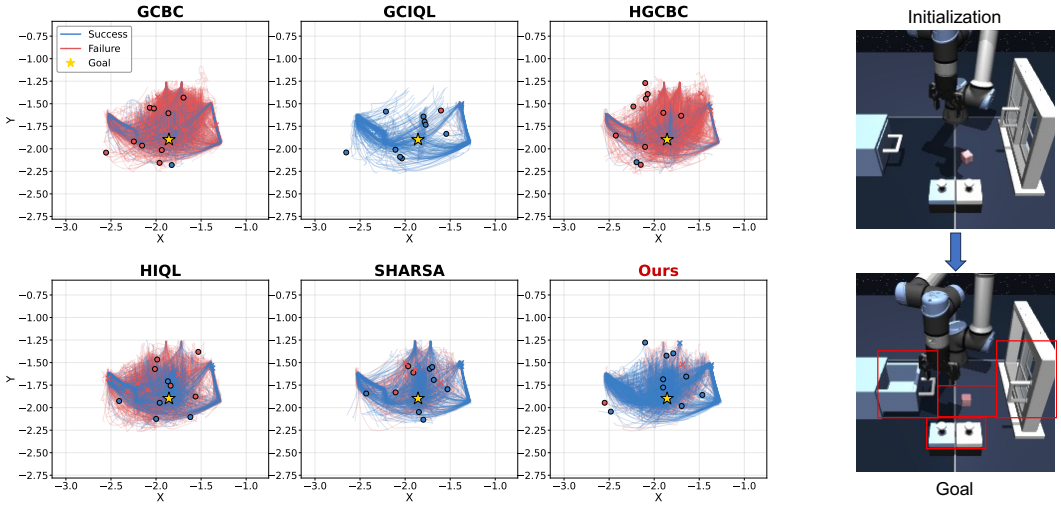

Figure 6: Trajectory visualization on `scene-play` task. Blue and red curves denote successful and failed trajectories, respectively, and yellow stars indicate goals. Compared with HRL baselines, our method produces more concentrated and goal-directed trajectories around key objects, with fewer scattered or oscillatory failures.

Figures 5 and 6 compare trajectory behaviors on `pointmaze-navigation` and `scene-play` tasks. Flat methods sometimes reach the goal but exhibit unstable behaviors near key decisions, while several HRL baselines either stay near the initial region or produce scattered failed trajectories around intermediate objects. In contrast, our method shows smoother and more goal-directed trajectories across both tasks. This indicates that subgoal alignment improves cross-level coordination, leading to more reliable hierarchical behaviors in long-horizon tasks.

Q7: Does BRIDGE improve execution efficiency?

Figure 7 reports the cumulative proportion of successful episodes completed within different step budgets, measuring how efficiently each method finishes tasks. On `pointmaze-navigation`, our method reaches near-complete success within about 250–270 steps, whereas flat methods require longer horizons and several HRL baselines obtain lower final completion rates. On `puzzle-4×4`, our method achieves the highest cumulative completion proportion and finishes more episodes within shorter step budgets than most baselines. Results show that our method improves execution

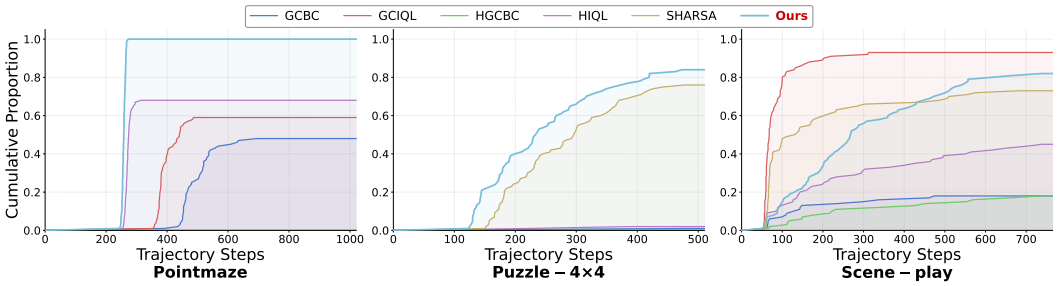

Figure 7: Cumulative success over trajectory steps. Curves show the cumulative proportion of successful episodes completed under different step budgets.

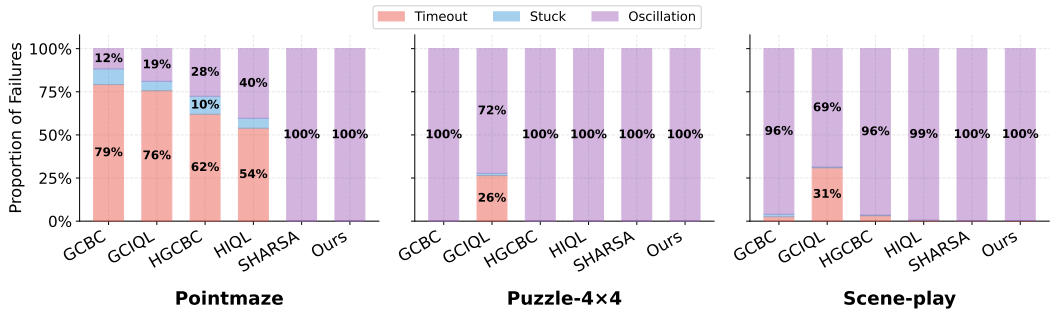

Figure 8: Failure-mode distribution across tasks. Timeout dominates `pointmaze-navigation` failures, whereas oscillation is the major failure type in `puzzle-4×4` and `scene-play`, revealing different bottlenecks across task structures.

efficiency by enabling agents to complete more tasks in fewer steps, which is important for practical robotic systems where shorter trajectories reduce runtime, energy cost, and failure exposure.

Q8: Where do robot simulation RL methods fail?

To better understand where current robot simulation RL methods fail, we analyze their failure modes in Figure 8. Failure modes are *timeout*, *stuck*, and *oscillation*; detailed definitions are provided in Appendix F. In `pointmaze-navigation`, failures are mostly timeout, suggesting that agents make progress but cannot finish within the horizon. In `puzzle-4×4` and `scene-play`, oscillation becomes dominant, indicating unstable sequential decisions in long-horizon reasoning and manipulation tasks. Stuck failures are less frequent but still reflect local trapping in navigation. Overall, this analysis shows that failure patterns vary by task structure: navigation is more prone to timeout, while compositional and manipulation tasks suffer more from oscillatory behaviors. Our method mitigates these issues by improving subgoal alignment, reducing infeasible planning and unstable oscillatory behaviors.

6 Conclusion

In this paper, we propose BRIDGE, a generalizable HRL framework that aligns high-level subgoal planning with low-level execution by encouraging executable subgoals through reachability regularization and preserving subgoal-responsive behaviors through mutual information regularization. Beyond performance improvements, our study provides a general analysis and evaluation perspective for robot simulation algorithms by examining not only final task success but also the internal decision structures, cross-level coordination, scalability, inference efficiency, and qualitative behaviors of learned policies. Extensive experiments on diverse goal-conditioned locomotion and manipulation tasks show that BRIDGE improves HRL performance, especially when standard hierarchical methods suffer from poor hierarchy–task compatibility.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, and improving the readability and organization of the manuscript.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of BRIDGE and its training objectives in the main paper, including the hierarchical policy formulation and the reachability and mutual-information regularizers. The appendix further documents the implementation details, training configurations, hyperparameters, benchmark and task settings, evaluation protocols, additional ablations, robustness experiments, quantitative analyses of subgoal alignment, and failure-mode definitions. We report results across multiple environments and tasks and include controlled ablations to isolate the contribution of each component.

REFERENCES

Hongjoon Ahn, Heewoong Choi, Jisu Han, and Taesup Moon. Option-aware temporally abstracted value for offline goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. Graph-assisted stitching for offline hierarchical reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2391–2408. PMLR, 2025.

Akhil Bagaria, Ben Abbatematteo, Omer Gottesman, Matt Corsaro, Sreehari Rammohan, and George Konidaris. Effectively learning initiation sets in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 36:73441–73463, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 17(93):1–50, 2016.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Thomas G Dietterich et al. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pp. 118–126, 1998.

Carlos Diuk, Anna Schapiro, Natalia Córdova, José Ribas-Fernandes, Yael Niv, and Matthew Botvinick. Divide and conquer: hierarchical reinforcement learning and task decomposition in humans. In *Computational and robotic models of the hierarchical organization of behavior*, pp. 271–291. Springer, 2013.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in neural information processing systems*, 32, 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.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

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)*, pp. 3389–3396. IEEE, 2017.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Nico Gürtler, Dieter Büchler, and Georg Martius. Hierarchical reinforcement learning with timed subgoals. In *Advances in Neural Information Processing Systems*, volume 34, pp. 21732–21743, 2021.

Zhigang Huang, Quan Liu, Fei Zhu, Lihua Zhang, and Lan Wu. Hierarchical reinforcement learning with unlimited option scheduling for sparse rewards in continuous spaces. *Expert Systems with Applications*, 237:121467, 2024.

Jaebak Hwang, Sanghyeon Lee, Jeongmo Kim, and Seungyul Han. Strict subgoal execution: Reliable long-horizon planning in hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2026.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8, 1993.

Kaiqiang Ke, Qian Lin, Zongkai Liu, Shenghong He, and Chao Yu. Conservative offline goal-conditioned implicit v-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 29591–29607. PMLR, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pp. 2917–2926. PMLR, 2018.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. DiffStitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28597–28609. PMLR, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Yu Luo, Fuchun Sun, Tianying Ji, and Xianyuan Zhan. Bidirectional-reachable hierarchical reinforcement learning with mutually responsive policies. *Reinforcement Learning Journal*, 2:733–762, 2024.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. Pmlr, 2020.

Luca Marzari, Ameya Pore, Diego Dall'Alba, Gerardo Aragon-Camarasa, Alessandro Farinelli, and Paolo Fiorini. Towards hierarchical task decomposition using deep reinforcement learning for pick and place subtasks. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 640–645. IEEE, 2021.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829*, 2019.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 48104–48127, 2025b.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2026.

Taegeon Park, Seungho Baek, Jongchan Park, Seungjun Oh, and Yusung Kim. Temporal distance-aware subgoal generation for offline hierarchical reinforcement learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 2305–2314. Association for Computing Machinery, 2025c. doi: 10.1145/3746252.3761326.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *Acm transactions on graphics (tog)*, 36(4):1–13, 2017.

Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International conference on machine learning*, pp. 4344–4353. PMLR, 2018.

Jürgen Schmidhuber and Reiner Wahnsiedler. Planning simple trajectories using neural subgoal generators. In *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, pp. 196–202, 1992.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Yixiang Shan, Haipeng Liu, Ting Long, and Yi Chang. RD-HRL: Generating reliable sub-goals for long-horizon sparse-reward tasks. In *International Conference on Learning Representations*, 2026a.

Yixiang Shan, Haipeng Liu, Ting Long, and Yi Chang. Rd-hrl: Generating reliable sub-goals for long-horizon sparse-reward tasks. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Wonchul Shin and Yusung Kim. Guide to control: Offline hierarchical reinforcement learning using subgoal generation for long-horizon and sparse-reward tasks. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 4217–4225, 2023. doi: 10.24963/ijcai.2023/469.

Aravind Srinivas, Ramnandan Krishnamurthy, Peeyush Kumar, and Balaraman Ravindran. Option discovery in hierarchical reinforcement learning using spatio-temporal clustering. *arXiv preprint arXiv:1605.05359*, 2016.

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

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

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. *Advances in neural information processing systems*, 7, 1994.

Franki Nguimatsia Tiofack, Fabian Schramm, Théotime Le Hellard, and Justin Carpentier. Svl: Goal-conditioned reinforcement learning as survival learning. *arXiv preprint arXiv:2604.17551*, 2026.

Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. *Advances in neural information processing systems*, 29, 2016.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Vivienne Huiling Wang, Joni Pajarinen, Tinghuai Wang, and Joni-Kristian Kämäräinen. State-conditioned adversarial subgoal generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 10184–10191, 2023.

Vivienne Huiling Wang, Tinghuai Wang, Wenyan Yang, Joni-Kristian Kämäräinen, and Joni Pajarinen. Probabilistic subgoal representations for hierarchical reinforcement learning. *arXiv preprint arXiv:2406.16707*, 2024.

Jan Wöhlke, Felix Schmitt, and Herke van Hoof. Hierarchies of planning and reinforcement learning for robot navigation. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 10682–10688. IEEE, 2021.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3357–3364. ieee, 2017.

APPENDIX OVERVIEW

This appendix provides additional methodological details, experimental results, analyses, and implementation information for BRIDGE. It is organized as follows:

1. **Appendix A: BRIDGE Training Procedure.** Complete pseudocode of BRIDGE, including value learning, hierarchical flow policies, reachability regularization, and mutual-information regularization.
2. **Appendix B: Detailed Per-Environment Results.** Fine-grained success rates across individual OGBench environments.
3. **Appendix C: Comparison with Additional Hierarchical Baselines.** Additional comparisons with recent hierarchical goal-conditioned methods across all 55 tasks.
4. **Appendix D: Quantitative Validation of Subgoal Alignment.** Direct measurements of high-level subgoal executability and low-level subgoal-conditioned execution.
5. **Appendix E: Hyperparameter Stability across Environments.** Sensitivity and stability analyses of the reachability and mutual-information regularization weights.
6. **Appendix F: Failure Mode Definitions.** Operational definitions of timeout, stuck, and oscillation used in the failure analysis.
7. **Appendix G: Environment Details.** Detailed descriptions of OGBench and D4RL tasks, including their compatibility with hierarchical decomposition.
8. **Appendix H: Baseline Details.** Formulations and implementation details of the flat and hierarchical baselines considered in our experiments.
9. **Appendix I: Implementation Details.** Network architectures, training configurations, hyperparameters, inference procedures, goal sampling, and baseline settings.

A PSEUDOCODE OF BRIDGE

The pseudocode of BRIDGE is shown in Algorithm 1.

B ADDITIONAL DETAILED PER-ENVIRONMENT RESULTS.

In the main paper, we report experimental results primarily in aggregated form, together with results on selected representative environments. For completeness, Table 6 provides additional per-environment results across the evaluated methods. These results offer a more fine-grained view of performance across environments with different task characteristics and difficulty levels.

Table 6: Detailed per-environment results. We report the success rate (%). The best result in each environment is highlighted in bold.

Environment	GCBC	GCIVL	GCIQL	CRL	HGCBC	HIQL	SHARSA	**BRIDGE**
Cube-Single	7	42	55	0	3	9	64	**72**
Cube-Double	6	7	6	7	**17**	7	3	6
Antmaze-Medium	27	62	65	90	22	90	59	**91**
Antmaze-Large	22	14	33	89	13	**92**	36	62
Humanoidmaze-Medium	6	31	8	40	2	31	**48**	37
Humanoidmaze-Large	1	2	1	14	0	4	7	**16**

Overall, the detailed results are consistent with the aggregate trends reported in the main paper. BRIDGE achieves the highest success rate on Cube-Single, Antmaze-Medium, and Humanoidmaze-Large, while other methods perform best on some individual environments. This variation also illustrates the heterogeneity of the evaluated tasks and motivates reporting both aggregate and per-environment results.

Algorithm 1 Training Procedure of BRIDGE

Require: Offline dataset $\mathcal{D}$, value V, critic Q, target critic $\bar{Q}$, high-level flow v_h, low-level flow v_l, goal encoder f_ϕ, inverse model q_ψ

1: **for** each training iteration **do**

2: Sample a batch $\mathcal{B} = \{(s, a, s', g, g_l, g_h, s_h)\}$ from $\mathcal{D}$.

3: Update V by expectile regression:

$$\mathcal{L}_V = \mathbb{E}\left[|\tau - \mathbb{I}\{Q(s,g,a) - V(s,g) < 0\}| \left(Q(s,g,a) - V(s,g)\right)^2\right].$$

4: Update Q by TD regression:

$$\mathcal{L}_Q = \mathbb{E}\left[\left(Q(s,g,a) - r - \gamma m V(s',g)\right)^2\right].$$

5: Train the low-level flow with goal representation $z_l = f_\phi(s, g_l)$:

$$\mathcal{L}_l = \mathbb{E}\left[w_l \left\|v_l(s, z_l, x_t, t) - (a - x_0)\right\|^2\right], \quad w_l = \exp(\alpha_l(Q(s,g_l,a) - V(s,g_l))).$$

6: Train the high-level flow with target subgoal $z_h^\star = f_\phi(s, s_h)$:

$$\mathcal{L}_h = \mathbb{E}\left[w_h \left\|v_h(s, g_h, z_t, t) - (z_h^\star - z_0)\right\|^2\right], \quad w_h = \exp(\alpha_h(V(s_h,g_h) - V(s,g_h))).$$

7: Apply reachability regularization:

$$\mathcal{L}_r = -\mathbb{E}\left[\bar{Q}(s, g_h, \hat{a})\right], \quad \hat{z}_h = \mathrm{Flow}_h(s, g_h),\ \hat{a} = \mathrm{Flow}_l(s, \hat{z}_h).$$

8: Apply mutual-information regularization:

$$\mathcal{L}_{\mathrm{mi}} = \mathbb{E}\left[\left\|q_\psi(s,a) - z_l\right\|^2\right].$$

9: Update all networks by minimizing:

$$\mathcal{L} = \mathcal{L}_V + \mathcal{L}_Q + \mathcal{L}_l + \mathcal{L}_h + \lambda_r \mathcal{L}_r + \lambda_{\mathrm{mi}} \mathcal{L}_{\mathrm{mi}}.$$

10: Update target critic:

$$\bar{Q} \leftarrow \tau Q + (1-\tau)\bar{Q}.$$

11: **end for**

C Comparison with Additional Hierarchical Baselines

To further broaden the baseline comparison, we evaluate BRIDGE against three additional hierarchical methods, SVL Tiofack et al. (2026), OTA Ahn et al. (2025), CGCIVL Ke et al. (2025), BrHPO Luo et al. (2024), and GAS Baek et al. (2025) across all 55 offline goal-conditioned tasks. As shown in Table 7, BRIDGE achieves an average success rate of 51%, outperforming SVL, OTA, CGCIVL, BrHPO and GAS. These results further demonstrate that the improvements of BRIDGE remain consistent under a broader set of competitive baselines.

Table 7: Comparison with additional baselines across 55 offline goal-conditioned tasks. Results are average success rates (%).

Method	SVL(Tiofack et al. (2026))	OTA(Ahn et al. (2025))	CGCIVL(Ke et al. (2025))	BrHPO(Luo et al. (2024)	GAS(Baek et al. (2025))	BRIDGE
Success Rate (%)	37	47	44	38	47	**51**

D More quantitative validation of subgoal alignment

We further provide quantitative evidence that the two regularizers improve their intended forms of subgoal alignment. For reachability regularization, we directly measure *Subgoal Success Rate* by rolling out the low-level policy toward the selected subgoal. As shown in Table 8, the success rate increases from 51% to 65%, indicating that the high-level policy selects more executable subgoals.

Meanwhile, Reach Q increases from -51.96 to -51.28, whereas the overall Critic Q decreases from -50.70 to -51.58. Thus, the higher subgoal success rate is not explained by a uniform increase in critic values, supporting the use of Reach Q as a relative, task-conditioned signal for subgoal executability rather than a calibrated reachability probability.

For MI regularization, we quantify how strongly low-level actions encode the assigned subgoal by comparing z_{true} with $\hat{z} = f_\phi(s, a)$ inferred from each state–action pair. Their cosine similarity increases from 0.889 to 0.972 (Table 8), showing that low-level actions become more consistent with the assigned subgoal. Together, these results quantitatively validate improved alignment in both directions: more executable subgoal selection at the high level and stronger subgoal-conditioned execution at the low level.

Table 8: More quantitative validation of subgoal alignment.

Reg.	**Metric**	**Without**	**With**	Δ
Reach.	Subgoal Success Rate (%)	51	65	+14
Reach.	Reach Q	-51.96	-51.28	+0.68
Reach.	Critic Q	-50.70	-51.58	-0.88
MI	Cosine Similarity	0.889	0.972	+0.083

E Hyperparameter stability across environments

All main experiments use the same alignment weights, $\lambda_r = 1.0$ and $\lambda_{\text{mi}} = 0.5$, without per-task tuning. To assess hyperparameter stability, we sweep each weight across Puzzle-4×4, Scene-Play, and PointMaze-Large. Table 9 reports the ranges over which performance remains stable and the relative standard deviation within each range. Both weights have stable ranges across all three environments, with relative standard deviations of 3.5–11.0%. The global setting lies within every reported range, indicating that the main results do not depend on environment-specific tuning.

Table 9: Hyperparameter stability across environments.

	Puzzle-4×4	Scene-Play	PointMaze-Large
λ_r stable range	$[0.5, 1.5]$	$[0.4, 1.0]$	$[0.6, 2.0]$
Relative std.	4.6%	9.9%	9.2%
λ_{mi} stable range	$[0.25, 0.75]$	$[0.35, 0.7]$	$[0.3, 0.7]$
Relative std.	3.5%	10.1%	11.0%

F Failure Mode Definitions

We provide the operational definitions used for the failure analysis in Fig. 8. We first retain only unsuccessful episodes, i.e., trajectories with a success indicator no larger than 0.5, and assign each failed trajectory to a single failure category according to its trajectory statistics.

Let p_0 and p_T denote the initial and final agent positions, respectively. We define the net displacement as

$$D = \|p_T - p_0\|_2,$$

and the traveled path length as

$$L = \sum_{t=0}^{T-1} \|p_{t+1} - p_t\|_2.$$

The failure modes are defined as follows.

Stuck. A trajectory is classified as *stuck* when the agent exhibits little overall motion throughout the episode. Specifically, we require $D < 0.5$ and $L < 1.5$. This captures cases where the agent remains trapped in a local region or repeatedly executes ineffective actions without meaningful movement.

Timeout. A trajectory is classified as *timeout* when its episode length reaches at least 95% of the maximum environment horizon without completing the task. This typically corresponds to trajectories that continue executing but fail to reach the task objective within the allowed horizon.

Oscillation. We identify oscillatory behavior by comparing the traveled path length with the net displacement. A trajectory is classified as *oscillation* when

$$\frac{L}{D+\epsilon} > 8,$$

indicating that the agent travels substantially farther than its eventual displacement. Such trajectories typically correspond to repeated back-and-forth motion or switching between intermediate decisions without sustained progress. For trajectories that also reach the episode horizon, oscillation takes precedence over timeout when this criterion is satisfied.

This rule-based categorization provides a consistent diagnostic view across methods and environments without relying on manual trajectory annotation.

G Environment Details

G.1 OGBench

G.1.1 Benchmark Overview

OGBench (Park et al., 2025a) is a benchmark for offline goal-conditioned reinforcement learning. It provides diverse long-horizon control tasks across locomotion and manipulation domains, and is designed to evaluate goal reaching, trajectory stitching, sequential reasoning, and generalization. In our experiments, we focus on state-based goal-conditioned environments from the locomotion and manipulation suites.

Overall, we evaluate on 11 environments, each containing 5 distinct tasks, resulting in 55 tasks in total. The evaluated environments are:

- `pointmaze-medium-navigate-v0`, `pointmaze-large-navigate-v0`;
- `antmaze-medium-navigate-v0`, `antmaze-large-navigate-v0`;
- `humanoidmaze-medium-navigate-v0`, `humanoidmaze-large-navigate-v0`;
- `cube-single-play-v0`, `cube-double-play-v0`;
- `puzzle-4x4-play-v0`, `puzzle-4x5-play-v0`;
- `scene-play-v0`.

G.1.2 Environment Categories

Maze navigation. The maze navigation suite includes `PointMaze`, `AntMaze`, and `HumanoidMaze`. In `PointMaze`, the agent controls a simple 2-D point mass and must navigate through a maze to reach a target goal location. `AntMaze` increases the control difficulty by using an 8-DoF quadruped Ant agent, requiring both low-level locomotion and high-level navigation. `HumanoidMaze` further increases the challenge with a 21-DoF Humanoid agent, where full-body control, balance, and long-horizon navigation must be solved jointly. These environments include maze layouts of different scales, such as medium and large, where larger mazes require longer-horizon planning and more effective intermediate goal selection.

Object manipulation. The object manipulation suite includes `Cube` and `Scene`. In `Cube`, a robot arm manipulates colored cube blocks and must arrange them into target configurations. The task variants differ in the number of cubes, such as `cube-single` and `cube-double`, and require skills such as reaching, moving, stacking, swapping, or rearranging objects. In `Scene`, the robot interacts with multiple objects, including cubes, drawers, windows, and buttons. Some objects also have lock states controlled by buttons, so successful completion may require executing a sequence of dependent manipulation skills.

Puzzle manipulation. The `Puzzle` environments are based on the "Lights Out" puzzle. A robot arm must press buttons on a grid, where pressing one button changes the state of both the selected button and its neighboring buttons. Thus, the agent must solve a combination of discrete combinatorial reasoning and continuous robotic control. We evaluate on `puzzle-4x4-play-v0`

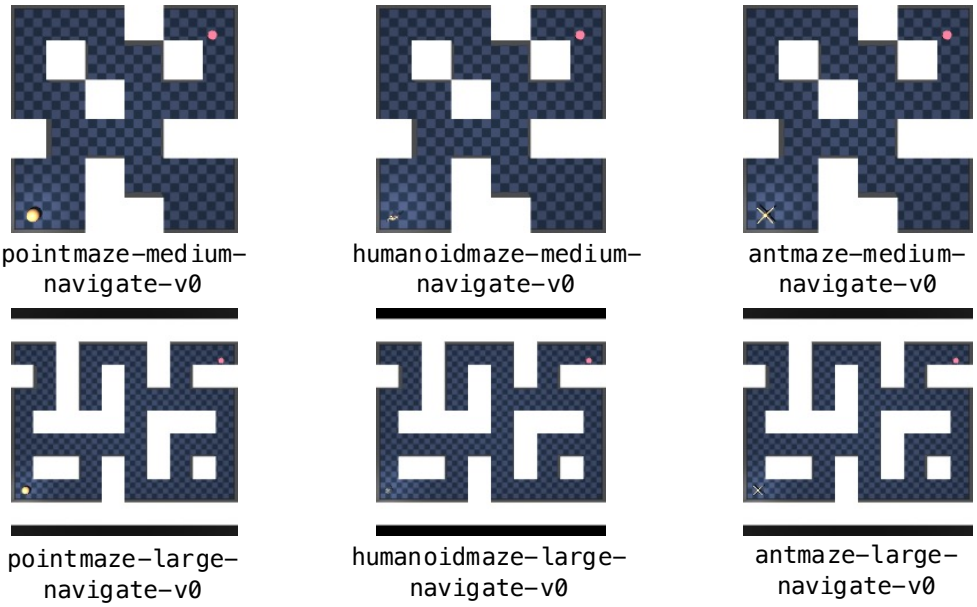

Figure 9: Locomotion tasks of OGBench.

and `puzzle-4x5-play-v0`, where larger grids introduce more complex state dependencies and longer action sequences.

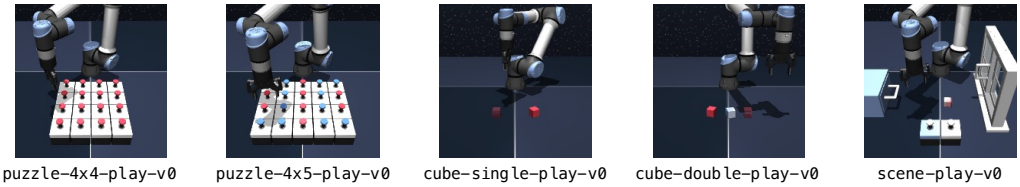

Figure 10: Manipulation tasks of OGBench.

G.1.3 Hierarchy-Compatible Environments

Some OGBench environments naturally match the structure of hierarchical reinforcement learning. In these tasks, the final goal is difficult to reach through a short sequence of primitive actions. Instead, the solution typically requires intermediate states, reusable skills, or temporally extended behaviors. In such cases, a high-level policy can select meaningful subgoals, while a low-level policy solves shorter-horizon control problems conditioned on these subgoals.

Embodied maze navigation. `AntMaze` and `HumanoidMaze` are representative hierarchy-compatible environments. Although their final objective is navigation, solving them requires two coupled levels of decision making. At the low level, the agent must learn stable locomotion behaviors, such as moving forward, turning, and maintaining balance. At the high level, it must reason over the maze structure and select intermediate waypoints that gradually lead to the target. This separation is especially important for Ant and Humanoid agents because their action spaces and dynamics are much more complex than those of a point mass. Therefore, subgoal-based decomposition can reduce the effective planning horizon and make long-horizon navigation more tractable.

Structured manipulation. `Cube` and `Puzzle` are also naturally compatible with hierarchy, but for different reasons. In `Cube`, the robot often needs to execute a sequence of object-centric skills, such as reaching a cube, moving it to an intermediate position, stacking it, or rearranging multiple cubes into the desired configuration. For `cube-double`, dependencies between objects further

strengthen the hierarchical structure because manipulating one cube may be a prerequisite for manipulating another.

In `Puzzle`, the agent must decide which button or local puzzle state to target next, while also physically reaching and pressing the selected button. A natural hierarchy is therefore to let the high-level policy plan over button states or puzzle configurations, and let the low-level policy execute the corresponding reaching and pressing motions. These tasks provide clear intermediate objectives and temporally extended action segments, making them well suited for hierarchical policies.

G.1.4 Less Hierarchy-Compatible Environments

We also include environments where hierarchical structure is weaker, more ambiguous, or less consistently aligned with the task dynamics. These tasks are important because an effective HRL method should not only exploit natural temporal abstraction, but also avoid degradation when hierarchy is unnecessary or difficult to learn.

PointMaze. `PointMaze` has less explicit hierarchical structure than `AntMaze` or `HumanoidMaze`. The agent is a simple 2-D point mass, and its primitive actions are already closely aligned with the navigation objective. Although the large maze still requires long-horizon planning, the low-level control problem is relatively simple: moving toward a nearby location is almost directly executable. As a result, the distinction between high-level subgoals and low-level actions becomes weaker. A strong flat goal-conditioned policy can often learn effective goal-reaching behavior without explicitly decomposing the task into separate planning and control levels. Therefore, `PointMaze` is useful for testing whether hierarchical methods remain robust when the benefit of hierarchy is less pronounced.

Scene. `Scene` is also challenging for hierarchy, but for a different reason. The environment contains rich interactions with cubes, drawers, windows, and buttons, which may suggest long-horizon structure. However, the task distribution is diverse and less regular than that of `Cube` or `Puzzle`. Different goals may involve different objects, interaction orders, and dependencies, making it difficult to define a consistent set of reusable subgoals. Moreover, some subgoals may be meaningful only under specific object configurations. For example, opening a drawer, moving a cube, or pressing a button may be useful for one goal but irrelevant for another. Thus, although `Scene` contains complex interactions, its hierarchical structure is less cleanly aligned with a fixed decomposition into high-level subgoals and low-level skills. This makes it a challenging environment where hierarchical policies may suffer from infeasible or poorly aligned subgoals.

Overall, our OGBench evaluation includes both hierarchy-compatible and less hierarchy-compatible environments. The former, including `antmaze`, `humanoidmaze`, `cube`, and `puzzle`, evaluates whether a method can exploit natural temporal abstraction through subgoal-based planning. The latter, including `pointmaze` and `scene`, tests whether the method remains robust when hierarchy is weak, ambiguous, or less aligned with the task dynamics. Together, these environments provide a balanced testbed for assessing both the benefits and potential limitations of hierarchical goal-conditioned reinforcement learning.

G.2 D4RL

We further evaluate our method on four dexterous manipulation tasks from D4RL (Fu et al., 2020): `pen-expert-v2`, `hammer-expert-v2`, `door-expert-v2`, and `relocate-expert-v2`. These tasks belong to the Adroit manipulation suite and are designed to evaluate high-dimensional robotic control with a dexterous hand. The Adroit platform consists of a multi-DoF ShadowHand mounted on a robotic arm, resulting in challenging continuous-control problems with complex contact dynamics and sparse task completion criteria.

In `pen`, the agent controls the hand to reorient a pen so that its orientation matches a randomized target orientation. In `hammer`, the agent must grasp a hammer and use it to drive a nail into a board. In `door`, the agent must unlatch and open a door, where the latch introduces friction and requires interaction with the environment. In `relocate`, the agent must move a ball to a randomized target position using the dexterous hand. These environments test different manipulation abilities, including object reorientation, tool use, articulated-object interaction, and object relocation.

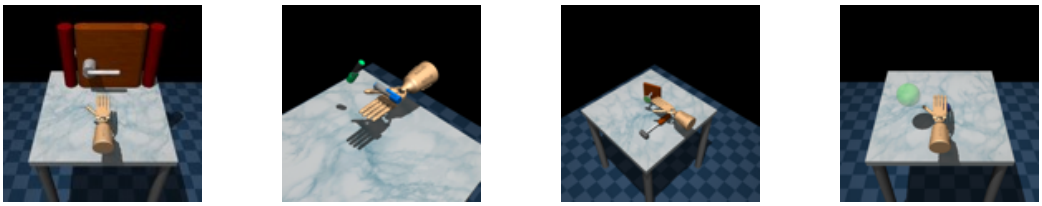

Figure 11: Manipulation tasks of D4RL.

Following the D4RL benchmark setting, each task provides offline datasets collected from different behavior sources, including human demonstrations, cloned policies, and expert policies. We use these tasks to evaluate whether the learned policy can generalize from fixed offline data in high-dimensional manipulation settings.

H Baseline Details

We provide details of the baselines used in our experiments. Let $\mathcal{D}$ denote the offline dataset of trajectories $\tau = (s_0, a_0, \ldots, s_H)$. For goal-conditioned methods, the goal g is sampled from the dataset using hindsight relabeling, e.g., from future states in the same trajectory or from other dataset states, following standard goal-conditioned offline RL practice.

Goal-conditioned behavioral cloning (GCBC) (Lynch et al., 2020; Ghosh et al., 2019). GCBC trains a goal-conditioned policy by directly imitating dataset actions conditioned on relabeled goals. Given a state-action pair (s, a) and a sampled goal g, GCBC maximizes the likelihood of the dataset action:

$$\mathcal{J}_{\mathrm{GCBC}}(\pi) = \mathbb{E}_{(s,a)\sim p^{\mathcal{D}}(s,a),\, g\sim p^{\mathcal{D}}(g|s,a)} \left[\log \pi(a \mid s, g)\right]. \tag{9}$$

This method does not perform value learning or policy improvement, and therefore serves as a pure imitation baseline for goal-conditioned control.

Goal-conditioned implicit V-learning (GCIVL) (Kostrikov et al., 2021). GCIVL learns an action-free goal-conditioned value function $V(s, g)$ using implicit value learning. The value function estimates the reachability of goal g from state s and is trained with an expectile temporal-difference loss:

$$\mathcal{L}_V = \mathbb{E}_{(s,a,s')\sim p^{\mathcal{D}},\, g\sim p^{\mathcal{D}}(g|s,a)} \left[L_2^\kappa \left(V(s, g) - r(s, g) - \gamma V(s', g)\right)\right], \tag{10}$$

where $L_2^\kappa(u) = |\kappa - I(u < 0)|u^2$ is the expectile loss. The policy is then extracted by advantage-weighted regression:

$$\mathcal{J}_\pi = \mathbb{E}_{(s,a,s')\sim p^{\mathcal{D}},\, g} \left[\exp\left(\alpha A(s, s', g)\right) \log \pi(a \mid s, g)\right], \tag{11}$$

where

$$A(s, s', g) = r(s, g) + \gamma V(s', g) - V(s, g). \tag{12}$$

Compared with GCBC, GCIVL uses value estimates to emphasize actions that make larger progress toward the goal.

Goal-conditioned implicit Q-learning (GCIQL) (Kostrikov et al., 2021). GCIQL extends implicit Q-learning to goal-conditioned offline RL. It learns a goal-conditioned Q-function $Q(s, a, g)$ and value function $V(s, g)$. The value function is trained with expectile regression:

$$\mathcal{L}_V = \mathbb{E}_{(s,a)\sim p^{\mathcal{D}},\, g} \left[L_2^\kappa \left(Q(s, a, g) - V(s, g)\right)\right], \tag{13}$$

while the Q-function is trained with Bellman regression:

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,s')\sim p^{\mathcal{D}},\, g} \left[\left(Q(s, a, g) - r(s, g) - \gamma V(s', g)\right)^2\right]. \tag{14}$$

The policy is obtained by advantage-weighted behavioral cloning:

$$\mathcal{J}_\pi = \mathbb{E}_{(s,a)\sim p^{\mathcal{D}},\, g} \left[\exp\left(\alpha(Q(s, a, g) - V(s, g))\right) \log \pi(a \mid s, g)\right]. \tag{15}$$

Thus, GCIQL improves over pure imitation by reweighting dataset actions according to their estimated goal-reaching advantage.

Contrastive Reinforcement Learning (CRL) (Eysenbach et al., 2022). CRL learns a goal-conditioned compatibility function $f(s, a, g)$ using a contrastive objective. Positive goals are sampled from reachable future states, while negative goals are sampled from the dataset. The critic is trained by binary noise-contrastive estimation:

$$\mathcal{J}_f = \mathbb{E}_{(s,a)\sim p^{\mathcal{D}},\, g^+\sim p_+^{\mathcal{D}},\, g^-\sim p_-^{\mathcal{D}}} \left[\log \sigma(f(s, a, g^+)) + \log\left(1 - \sigma(f(s, a, g^-))\right)\right], \tag{16}$$

where $\sigma(\cdot)$ is the sigmoid function. The learned compatibility score acts as a goal-reaching value signal. Following standard CRL practice, the policy is extracted by maximizing the learned score while staying close to the dataset action distribution:

$$\mathcal{J}_\pi = \mathbb{E}_{(s,a)\sim p^{\mathcal{D}},\, g} \left[f(s, \mu_\pi(s, g), g) + \alpha \log \pi(a \mid s, g)\right], \tag{17}$$

where $\mu_\pi(s, g)$ denotes the mean action of the policy.

Hierarchical GCBC (HGCBC) (Le et al., 2018; Lynch et al., 2020). HGCBC is a hierarchical extension of GCBC that reduces the effective policy horizon. It trains a high-level policy $\pi_h(z \mid s, g)$ to predict an intermediate subgoal z, and a low-level policy $\pi_l(a \mid s, z)$ to execute actions toward this subgoal. Given a subgoal interval h, the high-level target is a future state $z = s_{t+h}$, while the final goal is sampled as $g = s_{t+k}$ with $k > h$:

$$\mathcal{J}_{\mathrm{HGCBC}} = \mathbb{E}_{\tau\sim\mathcal{D},\, t,\, h,\, k} \left[\log \pi_h(s_{t+h} \mid s_t, s_{t+k}) + \log \pi_l(a_t \mid s_t, s_{t+h})\right]. \tag{18}$$

This baseline introduces temporal abstraction through subgoals but still relies only on behavioral cloning, without value-based policy improvement.

Hierarchical implicit Q-learning (HIQL) (Park et al., 2023). HIQL combines implicit value learning with hierarchical policy extraction. It first trains a single goal-conditioned value function $V(s, g)$ using an implicit V-learning objective:

$$\mathcal{L}_V = \mathbb{E}_{(s,a,s')\sim p^{\mathcal{D}},\, g} \left[L_2^\kappa \left(V(s, g) - r(s, g) - \gamma V(s', g)\right)\right]. \tag{19}$$

Then, it extracts a high-level policy and a low-level policy with advantage-weighted objectives. The high-level policy predicts a latent subgoal representation for a future state, while the low-level policy predicts the action conditioned on this subgoal:

$$\mathcal{J}_h = \mathbb{E}\left[\exp\left(\alpha(V(s_{t+h}, g) - V(s_t, g))\right) \log \pi_h(z_{t+h} \mid s_t, g)\right], \tag{20}$$

$$\mathcal{J}_l = \mathbb{E}\left[\exp\left(\alpha(V(s_{t+1}, s_{t+h}) - V(s_t, s_{t+h}))\right) \log \pi_l(a_t \mid s_t, z_{t+h})\right]. \tag{21}$$

By extracting hierarchical policies from a goal-conditioned value function, HIQL reduces the policy horizon while preserving value-guided action selection.

SHARSA (Park et al., 2026). SHARSA is a horizon-reduction method that combines hierarchical behavioral cloning with n-step SARSA value learning. It trains a high-level policy $\pi_h(w \mid s, g)$ to propose an intermediate subgoal w, and a low-level policy $\pi_l(a \mid s, w)$ to reach the selected subgoal. The value functions are defined at the high level as $Q_h(s, w, g)$ and $V_h(s, g)$, and are trained with n-step SARSA targets:

$$\mathcal{L}_V = \mathbb{E}\left[D\left(V_h(s_t, g), Q_h(s_t, s_{t+n}, g)\right)\right], \tag{22}$$

$$\mathcal{L}_Q = \mathbb{E}\left[D\left(Q_h(s_t, s_{t+n}, g), \sum_{i=0}^{n-1} \gamma^i r(s_{t+i}, g) + \gamma^n V_h(s_{t+n}, g)\right)\right], \tag{23}$$

where $D(\cdot, \cdot)$ can be either a regression loss or a binary cross-entropy loss. At test time, SHARSA samples multiple candidate subgoals from the high-level behavioral policy and selects the one with the largest high-level Q-value:

$$w = \arg \max_{w_i,\ w_i\sim\pi_h(\cdot|s,g)} Q_h(s, w_i, g). \tag{24}$$

The low-level policy then executes an action sampled from $\pi_l(a \mid s, w)$. In this way, SHARSA reduces both the value horizon through n-step SARSA and the policy horizon through hierarchical subgoal-conditioned control.

I Implementation Details

I.1 General Setup

All agents are implemented in JAX/Flax with Optax optimizers. Training is conducted for 1,000,000 gradient steps with a batch size of 1024. We evaluate every 100,000 steps using 50 episodes per task. All experiments are run with 3 random seeds (0, 1, 2). We use the Adam optimizer with a learning rate of 3×10^{-4} across all agents unless otherwise specified. All networks use layer normalization by default.

I.2 Network Architectures

All MLPs use ReLU activations with layer normalization. Unless otherwise noted, hidden dimensions are $(512, 512, 512)$ for both actor and value/critic networks.

Flow-based actors use the `ActorVectorField` architecture, which takes as input the concatenation of the state, goal (or subgoal representation), noisy action, and time step $t \in [0, 1]$. The network predicts the velocity field $v_\theta(x_t, s, g, t)$ for flow matching.

Goal representation network ϕ: an MLP mapping $[s; g]$ to a length-normalized vector of dimension $d_{\text{rep}} = 10$.

I.3 Proposed Method: BRIDGE

I.3.1 Architecture

Table 10: Network architecture of BRIDGE.

Component	Architecture	Input	Output
High-level actor π_H	Flow (ActorVectorField)	(s, g, x_t^H, t)	velocity v^H
Low-level actor π_L	Flow (ActorVectorField)	$(s, \phi(z), x_t^L, t)$	velocity v^L
Value $V(s, g)$	MLP $(512, 512, 512)$	(s, g)	scalar
Critic $Q(s, g, a)$	Ensemble of 2 MLPs	(s, g, a)	scalar
Goal rep ϕ	MLP + LengthNorm	(s, g)	$\mathbb{R}^{10}$
Inverse model $q(z \mid s, a)$	MLP $(512, 512, 512)$	(s, a)	$\mathbb{R}^{10}$

I.3.2 Hyperparameters

Table 11 shows the hyperparameters of BRIDGE.

I.3.3 Inference

At inference time, both levels use Euler ODE integration with 10 steps from Gaussian noise to generate subgoals and actions respectively. Optionally, reject sampling is applied: N candidates are generated and scored by the Q-critic, selecting the highest-scoring candidate.

I.3.4 Goal Sampling Strategy

Table 12 shows the goal sampling strategy of BRIDGE.

I.4 Baseline Settings

GCBC (Lynch et al., 2020; Ghosh et al., 2019). GCBC is a flat behavioral cloning baseline that trains a Gaussian MLP policy by maximizing the log-likelihood of dataset actions conditioned on the current state and goal (s, g). The policy uses hidden dimensions $(512, 512, 512)$ with a constant standard deviation and learning rate 3×10^{-4}.

GCIQL (Kostrikov et al., 2021). GCIQL is a flat offline RL baseline based on implicit Q-learning. It learns a goal-conditioned value function $V(s, g)$ through expectile regression from $Q(s, g, a)$, and

Table 11: Hyperparameters of BRIDGE.

Hyperparameter	Value
Learning rate	3×10^{-4}
Batch size	1024
Discount γ	0.99
Target network EMA τ	0.005
Expectile τ_{IQL}	0.9
Low-level AWR temperature α_L	3.0
High-level AWR temperature α_H	3.0
Subgoal temporal distance k	25
Representation dimension d_{rep}	10
Flow integration steps (inference)	10
High-level flow steps (inference)	10
Reachability weight λ_{reach}	1.0
Reachability samples N_{reach}	4
Low-level flow steps (reachability)	5
High-level flow steps (reachability)	5
MI weight λ_{MI}	0.5

Table 12: Goal sampling probabilities.

Parameter	Value	Description
p_{curgoal} (value)	0.2	Current state as goal
p_{trajgoal} (value)	0.5	Future state in trajectory
$p_{\text{randomgoal}}$ (value)	0.3	Random state from dataset
p_{trajgoal} (actor)	1.0	Actor trained on trajectory goals only
Geometric sampling	Yes	Geometric distribution for temporal distance

trains the actor with a DDPG+BC objective weighted by the estimated advantage. We use expectile 0.9, discount 0.99 for medium/large tasks and 0.995 for giant tasks. The actor weight α is set to 0.003 for PointMaze and 0.3 for AntMaze.

GCIVL (Kostrikov et al., 2021). GCIVL is a flat value-based baseline that uses an ensemble of goal-conditioned value functions with a Double-DQN-style expectile loss. The actor is trained with advantage-weighted regression (AWR). We set $\alpha = 10.0$, expectile 0.9, and target update rate $\tau = 0.005$.

CRL (Eysenbach et al., 2022). CRL learns a bilinear goal-conditioned value function through contrastive learning over state and goal representations $\phi(s)$ and $\psi(g)$. The actor is trained with a DDPG+BC objective. We use latent dimension 512, with $\alpha = 0.03$ for PointMaze and $\alpha = 0.1$ for AntMaze.

HGCBC (Le et al., 2018; Lynch et al., 2020). HGCBC is a pure behavioral cloning variant of HIQL. It keeps the same hierarchical policy structure and goal representation network, but removes value functions and advantage weighting.

HIQL (Park et al., 2023). HIQL is a hierarchical offline RL baseline with learned subgoal representations. The high-level policy proposes subgoal representations, while the low-level policy executes actions conditioned on these subgoals. Both levels are trained with AWR using IVL-style value functions. We use $\alpha_H = \alpha_L = 3.0$, representation dimension 10, subgoal interval 25, and discount 0.99 or 0.995 depending on the task horizon.

SHARSA (Park et al., 2026). SHARSA is a hierarchical baseline that applies SARSA-style Q-learning at the high level and uses rejection sampling during inference. The low-level policy is trained with pure flow-based behavioral cloning. We use 32 rejection samples, 10 flow steps, discount 0.999, and binary cross-entropy for the value loss.

I.5 Environment-Specific Hyperparameters

For giant-scale environments (pointmaze-giant, antmaze-giant, humanoidmaze-giant), we increase the discount factor to $\gamma = 0.995$ to account for longer horizons.