Switching Successor Measures for Hierarchical Zero-shot Reinforcement Learning

Stefan Stojanovic[1], **Alexandre Proutiere**[1,2]
stesto@kth.se, alepro@kth.se

[1]**KTH, Stockholm, Sweden**
[2]**Digital Futures, Stockholm, Sweden**

Abstract

Hierarchical reinforcement learning can improve generalization by decomposing long-horizon decision-making into simpler subproblems. However, existing approaches often rely on restrictive design choices, such as fixed temporal abstractions or goal-conditioned objectives, which largely confine them to goal-reaching tasks and limit their applicability to general reward functions. In this paper, we introduce *switching successor measures*, an extension of successor measures that enables hierarchical control in zero-shot reinforcement learning without additional supervision, fixed horizons, or manually designed subgoals. We show that switching successor measures arise naturally from classical successor measures while preserving their underlying structure. Building on this result, we propose FB π-Switch, an algorithm that extracts both a high-level subgoal-selection policy and a low-level control policy directly from forward-backward (FB) representations, allowing hierarchical behavior to emerge from a single learned representation. Experiments on both goal-conditioned and general reward-based tasks show that FB π-Switch improves over non-hierarchical baselines and matches state-of-the-art hierarchical methods in goal-conditioned settings. These results demonstrate that structured successor representations provide a flexible foundation for hierarchical zero-shot reinforcement learning beyond goal-reaching tasks. Our project website is available at: https://stestokth.github.io/switching-successors/.

1 Introduction

A central challenge in reinforcement learning (RL) is to design methods that are flexible and generalize well across a wide range of reward functions, enabling agents to adapt to new tasks without retraining from scratch. Goal-conditioned reinforcement learning (GCRL) has shown that structuring tasks around goals can significantly improve generalization (Chane-Sane et al., 2021; Eysenbach et al., 2022; Wang et al., 2025). Hierarchical approaches further extend this idea by decomposing long-horizon problems into sequences of shorter subproblems. For instance, methods such as HIQL (Park et al., 2023) demonstrate that conditioning on intermediate subgoals, rather than only on global task specifications, can substantially improve performance. However, these approaches rely on strong design choices. In particular, HIQL enforces locality through fixed subgoal horizons, rather than allowing such locality to emerge from the learning process itself.

Moreover, these methods are primarily studied in goal-conditioned settings, where tasks reduce to reaching specific states. In contrast, general RL involves arbitrary reward functions that cannot always be expressed as goal-reaching objectives. A prominent line of work addresses this setting through forward-backward (FB) representations of successor measures, which aim to represent value functions and policies in a shared embedding space (Touati & Ollivier, 2021; Touati et al., 2023; Tir-

inzoni et al., 2025). Despite their promise, these approaches rely on a strong global factorization assumption of the form $F(s, a, z)^\top B(g)$ for successor measures, which must simultaneously capture the dynamics of the environment and all downstream reward functions. Recent works have shown that such global low-dimensional structure can be difficult to learn and may be too restrictive in complex environments (Cetin et al., 2025; Agarwal et al., 2024; Frans et al., 2024; Dubail et al., 2025). This suggests that successor representations may be most reliable locally: they can capture short-range transitions effectively, while their accuracy may deteriorate as the effective planning horizon increases. Supporting this view, recent work on compositional planning (Farebrother et al., 2026) shows that combining local behaviors can improve generalization, although it requires an additional generative model. Finally, several recent methods treat representation learning as a first stage in zero-shot RL pipelines, with policy learning performed separately using standard offline algorithms (Ghosh et al., 2023; Park et al., 2024; Frans et al., 2024). While effective, this separation does not fully exploit the structure encoded in successor representations for joint planning and control.

Together, these observations highlight an important gap: there is currently no unified approach that leverages successor representations to derive hierarchical policies directly from the learned representation, while retaining the ability to generalize across arbitrary reward functions. This motivates our central question: *Can successor representations serve as a unified foundation from which both reward embeddings and hierarchical control emerge in zero-shot reinforcement learning?*

We address this question by introducing *switching successor measures* that allow us to derive hierarchical policies directly from successor representations, rather than introducing hierarchy as a separate supervised object. Concretely, in each state s, the agent first selects a latent subgoal w using a high-level policy $\pi^h(\cdot|s, r)$ conditioned on the reward function r, and then acts according to a low-level policy $\pi^\ell(\cdot|s, w)$ conditioned on the selected subgoal. Our main contributions are:

(a) We introduce *switching successor measures* as a principled framework for extracting hierarchical structure from successor-based representations. A key insight is that the switching successor measures needed for high-level planning can be derived directly from a single classical successor measure. We establish this remarkable connection in Theorem 1, showing that hierarchy is already implicitly encoded in standard successor representations and can be recovered without additional learning.

(b) We propose *FB π-Switch*, a hierarchical zero-shot RL algorithm in which both the high-level subgoal-selection policy and the low-level control policy are derived from FB successor representations. The method follows a three-stage training procedure; in particular, the high-level policy learning stage can be incorporated into existing FB algorithms.

(c) We empirically evaluate FB π-Switch on both goal-conditioned and more general reward-based tasks. The method achieves strong performance without explicit locality-inducing hyperparameters, and consistently outperforms other baselines, including FB with an induced high-level policy.

2 Related work

Successor representations and skill-based methods. Successor features (Barreto et al., 2017) provide a temporal-difference framework for zero-shot learning by decoupling dynamics from rewards. This idea has been extended to successor measures via FB representations, enabling zero-shot adaptation to new tasks (Touati et al., 2023). Subsequent work improves these models through online embedding adaptation (Sikchi et al., 2025) and more stable training objectives (Zheng et al., 2025). Alternative parameterizations have also been explored, e.g., intention-conditioned value functions (ICVFs) (Ghosh et al., 2023) and proto successor measures (Agarwal et al., 2024), which relax the linear reward assumption but require a discrete policy codebook. A related line of work learns reusable skills without explicit successor structure, including unsupervised skill discovery in online (Eysenbach et al., 2019) and offline settings (Ajay et al., 2021), as well as methods that improve skill transfer (Frans et al., 2024). More broadly, GCRL methods such as UVFA (Schaul et al., 2015)

and contrastive approaches (Eysenbach et al., 2022) learn policies conditioned on goals and can be viewed as a special case of successor-based formulations. However, these approaches typically rely on conditioning on fixed goals and do not explicitly capture hierarchical composition.

Hierarchical reinforcement learning. Hierarchical RL studies temporal abstraction, such as options, skills, and goals. The option framework (Sutton et al., 1999) provides a principled formulation for hierarchical RL, with extensions including gradient-based methods (Comanici & Precup, 2010) and actor-critic architectures (Bacon et al., 2017). Composing pretrained policies has also been explored in prior work (Liaw et al., 2017; Peng et al., 2017), while subgoal-based methods such as HIRO (Nachum et al., 2018) delegate goal generation to a high-level policy under a fixed reward function. Subgoal-based approaches have been widely studied in both online (Huang et al., 2019; Chane-Sane et al., 2021; Gürtler et al., 2021) and offline goal-conditioned RL. In particular, HIQL (Park et al., 2023) extends hierarchical RL to the offline regime, with subsequent work improving subgoal generation (Huang et al., 2024; Chen et al., 2024; Park et al., 2025b) and value estimation (Ke et al., 2025; Giammarino et al., 2025). Hierarchical structure can also be induced in skill-based frameworks, e.g., by optimizing for midpoints in latent space (Park et al., 2024), or by combining successor representations with generative models and planning (Farebrother et al., 2026).

3 Preliminaries

Problem setting. We consider an infinite-horizon Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, P is the transition kernel, and $\gamma \in (0, 1)$ is the discount factor. Given a reward function $r : \mathcal{S} \to \mathbb{R}$, we define the Q-function of a policy, represented by a distribution over actions $\pi(\cdot|s)$ in each state s, by $Q^\pi(s, a; r) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t)|s_0 = s, a_0 = a\right]$. A policy π is optimal for reward r if it maximizes the corresponding Q-function for all states and actions. We focus on the unsupervised setting, where the reward function is not specified a priori, and the objective is to learn optimal policies for arbitrary downstream reward functions. We assume access to an offline dataset $\mathcal{D}$ of transitions (s_t, a_t, s_{t+1}), collected under an unknown behavior policy. The states in $\mathcal{D}$ are distributed according to an unknown marginal ρ. The dimensionality of all learned representations is denoted by d. Finally, $\mathcal{Z} \subseteq \mathbb{R}^d$ denotes the latent intent space, from which z is sampled; the sampling distributions may vary across different stages of our algorithm.

Successor measure. Define the *successor measure* under policy π (Blier et al., 2021) and the corresponding *state-successor measure*[1] as:

$$M^\pi_{s,a}(s') = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s'\}|s_0 = s, a_0 = a\right], \quad M^\pi_s(s') = \mathbb{E}_{a\sim\pi(\cdot|s)} M^\pi_{s,a}(s').$$

For a state-dependent reward function r, the (r, π)-value functions can be written as $Q^\pi(s, a; r) = \langle M^\pi_{s,a}, r\rangle$ and $V^\pi(s; r) = \langle M^\pi_s, r\rangle$, where the inner product $\langle\cdot,\cdot\rangle$ is taken over states. In particular, for indicator rewards r_g, we have $V^\pi(s; r_g) = M^\pi_s(g)$ for all s. For brevity, we sometimes use a single index g to denote the indicator reward r_g; in this case, we write $V^\pi(s; g)$ in place of $V^\pi(s; r_g)$, and use $V^\star(s; g)$ as shorthand for $V^{\pi_g}(s; r_g)$. Finally, let $H^\pi_s(w)$ denote the hitting time of state w under policy π and starting from s, defined by $H^\pi_s(w) = \min\{t \geq 0 : s_t = w|s_0 = s, \pi\}$.

Learning successor measures. A standard approach to learning successor representations is the FB factorization of Touati et al. (2023), which models the successor measures of a parametrized family $(\pi_z)_{z\in\mathcal{Z}}$ of policies via the factorization $M^{\pi_z}_{s,a}(s') = F(s, a, z)^\top B(s')\rho(s')$. Such representations have the key property that, if the policy π_z satisfies for every state s, $\pi_z(s) \in \arg\max_a F(s, a, z)^\top z$, then π_z is optimal for the reward function $r_z(s) = \langle B(s), z\rangle$. The representations are obtained by

[1] $M^\pi_s(s')$ is called an intention-conditioned value function in Ghosh et al. (2023); we avoid this terminology since it corresponds to a value function only under indicator rewards supported solely at the goal state.

minimizing a squared Bellman error with respect to the measure ρ:

$$\min_{F,B} \mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D} \\ s'\sim\rho, z\sim\mathcal{Z}}} \left[\frac{\mathbf{1}\{s_t = s'\}}{\rho(s')} + \gamma \overline{F}(s_{t+1}, \pi_z(s_{t+1}), z)^\top \overline{B}(s') - F(s_t, a_t, z)^\top B(s') \right]^2, \quad (1)$$

where we use $\overline{F}, \overline{B}$ to denote that gradients are not propagated through the parameters of these networks (they are target networks). Thus, each training iteration of the FB algorithm consists of two intertwined steps: updating the networks F, B, and updating the policy π_z. In practice, an additional orthonormalization loss is imposed on B to fix the scale ambiguity. Importantly, the objective can be evaluated without explicit estimation of the marginal $\rho(s')$ (Appendix A.2).

An alternative approach learns successor-related value functions via offline regression using a direction-weighted expectile loss. Ghosh et al. (2023) extends the IQL objective of Kostrikov et al. (2022) to the intention-conditioned setting by minimizing over latent intentions $\mathcal{Z}$:

$$\min_{\theta} \mathbb{E}_{\substack{(s_t,s_{t+1})\sim\mathcal{D} \\ s'\sim\rho, z\sim\mathcal{Z}}} \Big[|\tau - \mathbf{1}\{\Delta < 0\}| \left(\mathbf{1}\{s_t = s'\} + \gamma V_\theta^{\pi_z}(s_{t+1}; s') - V_\theta^{\pi_z}(s_t; s') \right)^2 \Big], \quad (2)$$

where $\tau \in [\frac{1}{2}, 1]$ is a fixed expectile parameter and $\Delta = r_z(s_t) + \gamma V_\theta^{\pi_z}(s_{t+1}; r_z) - V_\theta^{\pi_z}(s_t; r_z)$. Here, r_z is an intent-specific reward function and $\pi_z(s) \in \arg\max_a r_z(s) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)} V_\theta^{\pi_z}(s'; r_z)$ the corresponding policy. The weighting term depends on the sign of Δ, emphasizing transitions with positive temporal-difference residuals under π_z; the squared term corresponds to the standard TD error for policy π_z and a target state s'. Note that Ghosh et al. (2023) does not use these value estimates for policy learning; instead, they serve purely for representation learning, after which a separate offline RL algorithm is applied (see Appendix A.4 for more details).

Policy extraction. We use advantage-weighted regression (AWR) (Peng et al., 2019) to train all policies, following Park et al. (2023). Given an advantage function $A(s, a)$, AWR learns a policy by maximizing:

$$\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\exp\left(\beta A(s, a)\right) \log \pi(a \mid s) \right], \quad (3)$$

where $\beta > 0$ is an inverse temperature parameter controlling the sharpness of the weighting. In our setting, the advantage function is defined in terms of the learned successor-based value functions, and the policy may condition on (s, z) and output either primitive actions or latent variables.

4 Switching successor measures

We introduce and study *switching successor measures*, a central object for hierarchical zero-shot reinforcement learning. Our starting point is the hierarchical goal-conditioned framework introduced in HIQL (Park et al., 2023). In the hierarchical stage of HIQL, the high-level policy is learned by maximizing a subgoal-based advantage objective. The latter quantifies the benefit of moving to state w (the subgoal) within the first k time steps when starting at state s and under reward function r_g:

$$A(s, w, g) = \gamma^k V^\star(w; g) + \sum_{t=0}^{k-1} \gamma^t r_g(s_t) - V^\star(s; g), \quad (4)$$

where the trajectory $(s_0 = s, s_1, \ldots, s_k = w)$ is drawn from the offline dataset. The terms highlighted in orange are omitted in HIQL under the goal structure, as discussed in Park et al. (2023). The framework relies on the equivalence $V^\pi(s; r_g) = M_s^\pi(g)$, valid in the goal-conditioned setting.

First we highlight two issues with the objective in (4):

(i) **Short-term return.** In the goal-conditioned framework of HIQL, a constant per-step reward, e.g., 0 or -1, allows the short-term return term to be omitted. More generally, however, the short-term return $\sum_{t=0}^{k-1} \gamma^t r_g(s_t)$ depends on both the reward and the states, and cannot be treated as a constant when r_g is replaced with a general reward function.

(ii) **Bias induced by the subgoal horizon k.** Defining the subgoal w via the advantage $A(s, w, g)$ implicitly assumes that it is reached in exactly k steps under the optimal policy. In practice, however, offline datasets rarely contain sufficiently many optimal trajectories. This creates a mismatch between the true k-step reachable set and the one observed in the data, which introduces bias into value estimation. This issue is further exacerbated in anisotropic environments, where progress depends strongly on direction. In such settings, a fixed horizon k may correspond to subgoals that are unevenly distant in terms of true task progress, even when they are all k-step reachable. Figure 1 illustrates this phenomenon. There, $\mathcal{N}_k(s)$ denotes the set of states reachable from s in k steps, while $\widehat{\mathcal{N}}_k(s)$ denotes the corresponding set inferred from the dataset. The subgoal $w' \in \widehat{\mathcal{N}}_k(s)$, selected under HIQL, is suboptimal, as it does not lie on the optimal trajectory from s to the goal g. A better choice would be w, which lies along the optimal path. If the dataset were exhaustive and contained all optimal trajectories, the optimal subgoal would instead be $w^\star$.

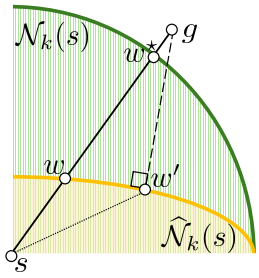


Figure 1: For $w, w' \in \widehat{\mathcal{N}}_k(s)$, it may hold that $V^\star(w'; g) > V^\star(w; g)$ even though w is the optimal subgoal.

Our framework resolves both issues by (i) accounting for the short-term return via our switching successor measures, and (ii) replacing fixed-horizon subgoal selection with hitting-time switching.

Addressing short-term return. To design high-level policies, we aim to capture the effect of temporarily following a subgoal-conditioned policy and then switching back to a globally efficient policy. Concretely, starting from state s and given an intermediate subgoal w, the agent follows π_w until time k, and thereafter follows π, meant to be an efficient policy for the given reward function r. To optimize subgoal selection, a natural objective is the k-step advantage function:

$$A_s^{\pi_w|_k \to \pi}(r) := \mathbb{E}_{\pi_w}\left[\gamma^k V^\pi(s_k; r) + \sum_{t=0}^{k-1} \gamma^t r(s_t) \,|s_0 = s\right] - V^\pi(s; r). \tag{5}$$

Instead of working with intermediate rewards and the state s_k along the trajectory, we can express the k-step advantage using successor measures. Define the k-step switching successor measure as:

$$M_s^{\pi_w|_k \to \pi}(s') := \sum_{t=0}^{k-1} \gamma^t \mathbb{P}_{\pi_w}(s_t = s'|s_0 = s) + \gamma^k \sum_{\tilde{s} \in \mathcal{S}} \mathbb{P}_{\pi_w}(s_k = \tilde{s}|s_0 = s) M_{\tilde{s}}^\pi(s'). \tag{6}$$

When $\pi_w = \pi$, this reduces to the standard successor measure, $M^{\pi|_k \to \pi}(\cdot) \equiv M^\pi(\cdot)$. The first term in (6) corresponds to the k-step truncated successor measure under π_w, which we denote by $M_s^{\pi_w|_k}(s')$. Moreover, $A_s^{\pi_w|_k \to \pi}(r) = \langle M_s^{\pi_w|_k \to \pi} - M_s^\pi, r\rangle$, and in the goal-conditioned setting, this reduces to $A_s^{\pi_w|_k \to \pi}(r_g) = M_s^{\pi_w|_k \to \pi}(g) - M_s^\pi(g)$.

Addressing subgoal horizon. Our notion of k-step advantage considers a fixed number of steps and there is no reason why, starting from $s_0 = s$ under π_w, we would reach w after k steps. To build intuition, however, consider the idealized case where $s_k = w$ deterministically under π_w when $s_0 = s$. In this very specific case, we have $M_s^{\pi_w|_k}(s') = M_s^{\pi_w}(s') - \gamma^k M_w^{\pi_w}(s')$, which enables us to rewrite (5) purely in terms of value functions:

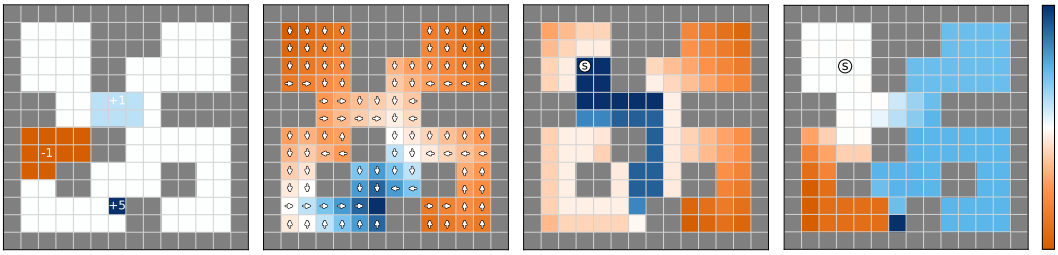
$$A_s^{\pi_w|_k \to \pi}(r) = V^{\pi_w}(s; r) + \gamma^k \big(V^\pi(w; r) - V^{\pi_w}(w; r)\big) - V^\pi(s; r).$$

Next, we relax the assumption $s_k = w$ and study policies that switch upon reaching the subgoal w. More precisely, consider a policy that first follows π_w until reaching state w, and then switches to π. Denote by $M^{\pi_w \to \pi}$ the successor measure obtained by plugging[2] $k = H_s^{\pi_w}(w)$ into (6):

$$M_s^{\pi_w \to \pi}(s') := \mathbb{E}\left[M_s^{\pi_w|_k \to \pi}(s') \mathbf{1}\{k = H_s^{\pi_w}(w)\}\right]. \tag{7}$$

The following theorem relates switching and standard successor measures:

[2]Definition (7) is equivalent to $M_s^{\pi_w \to \pi}(s') = \mathbb{E} M_s^{\pi_w|_{H_s^{\pi_w}(w)} \to \pi}(s')$, but we use the former for clarity.



(a) $r(s)$ (b) $V^\star(s; r)$ (c) $A_s^{\pi_w \to \pi^\star}(r)$ (d) $\mathbb{E}A_s^{\pi_w|_k \to \pi^\star}(r)|_{k<H_s^{\pi_w}(w)}$

Figure 2: Depicted quantities as functions of the magenta variables: (a) reward function with positive and negative regions, and a goal state with reward $+5$; (b) optimal value function corresponding to this reward, shown with one choice of optimal actions. Note that after passing through reward-bearing regions, the value of downstream states depends only on the final reward, not on the rewards encountered earlier; (c) switching advantage function defined in (8), which highlights the states on the optimal trajectory induced by the reward; (d) contributions to the switching advantage arising from rewards collected *before* switching (reaching w).

Theorem 1. *For any $s, w, s' \in \mathcal{S}$ and policy π, the following identity holds:*

$$M_s^{\pi_w \to \pi}(s') = M_s^{\pi_w}(s') + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}\Big(M_w^{\pi}(s') - M_w^{\pi_w}(s')\Big).$$

The proof is deferred to Appendix B.1. We next formalize the advantage of switching from π_w to π at w. For a given reward r, the *switching advantage function* is defined as:

$$A_s^{\pi_w \to \pi}(r) := \mathbb{E}\left[A_s^{\pi_w|_k \to \pi}(r)\mathbf{1}\{k = H_s^{\pi_w}(w)\}\right]. \tag{8}$$

Corollary 1. *For any $s, w \in \mathcal{S}$, any policy π, and any reward function r, holds:*

$$A_s^{\pi_w \to \pi}(r) = V^{\pi_w}(s; r) + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}\Big(V^{\pi}(w; r) - V^{\pi_w}(w; r)\Big) - V^{\pi}(s; r).$$

The switching advantage function serves as the objective for designing the high-level policy. Given a reward function r and an efficient base policy π, we define a high-level policy that selects subgoals w by maximizing the switching advantage function $A_s^{\pi_w \to \pi}(r)$ over w. The main quantities in this section are illustrated in Figure 2 for a discrete maze environment; see Appendix C for details.

5 FB π-Switch: Hierarchical learning with switching successor measures

In this section, we present a practical offline learning algorithm that leverages the hierarchical successor measures introduced in the previous section. Training of **FB π-Switch**, illustrated in Figure 3, consists of three stages: (1) jointly learning state-successor measure and representations, followed by (2) high-level policy learning and (3) low-level policy learning.

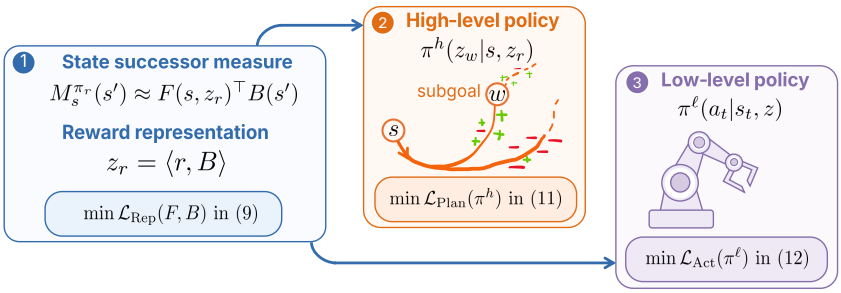


Figure 3: FB π-Switch training pipeline: learning state successor measures, high- and low-level control.

During testing, given a task defined by reward r and its embedding z_r, at each time step we first sample a subtask $z \sim \pi^h(\cdot \mid s, z_r)$ and then execute an action $a \sim \pi^\ell(\cdot \mid s_t, z)$. In the following, we describe the three training stages in detail and define their corresponding loss functions.

Step 1: Learning state-successor representations. We begin by learning action-free, reward-conditioned state-successor representations that can later be reused for hierarchical planning and control. Compared to the original FB objective (1), our formulation differs in two aspects: (i) it removes dependence on actions and policy-dependent bootstrapping, and (ii) it enables a single-step learning procedure, in contrast to the two coupled optimization steps required in FB framework (see Section 3). To achieve this, we marginalize over actions and replace the squared Bellman error with an expectile regression objective, following the idea of (2). This requires addressing two questions:

(a) *What does action-free FB represent?* If $M^{\pi_z}_{s,a}(s') = F(s,a,z)^\top B(s')$ holds, then the corresponding state-successor measures also admit a similar FB-representation. In particular,

$$M^{\pi_z}_{s}(s') = \mathbb{E}_{a\sim\pi_z(\cdot|s)} M^{\pi_z}_{s,a}(s') = \mathbb{E}_{a\sim\pi_z(\cdot|s)} F(s,a,z)^\top B(s') = F(s,z)^\top B(s').$$

where we define the action-marginalized successor features $F(s,z) := \mathbb{E}_{a\sim\pi_z(\cdot|s)} F(s,a,z)$. This allows us to directly optimize (F,B) for the state-successor measure $M^{\pi_z}_{s}(s')$. Furthermore, the induced value function admits a linear representation:

$$V^{\pi_z}(s;r) = \sum_{s'} M^{\pi_z}_{s}(s') r(s') = \sum_{s'} F(s,z)^\top B(s') r(s') = F(s,z)^\top z_r$$

where $z_r = \langle B, r\rangle = \mathbb{E}_{s'\sim\rho} \frac{1}{\rho(s')} B(s') r(s')$. Unlike the original FB formulation, the weighting factor $1/\rho(s')$ appears explicitly here. This difference arises from the fact that FB assumes the approximation $M^{\pi_z}_{s,a}(s') \approx F(s,a,z)^\top B(s')\rho(s')$, which absorbs the dependence on ρ into the representation. In contrast, when using non-quadratic losses, removing this dependence on ρ from the objective is no longer straightforward (see Appendix A.2 for more details).

(b) *How to learn without explicit policy feedback?* In the original FB algorithm, the bootstrap target depends explicitly on the policy via $F(s_{t+1}, \pi_z(s_{t+1}), z)$, coupling representation and policy learning. Since we aim for a one-step, action-free objective, this form of bootstrapping is unavailable. We instead replace explicit policy feedback with an implicit greedy backup via intention-conditioned expectile regression, akin to (2). To learn the representation of $M^{\pi_z}_{s_t}(s')$, we minimize:

$$\begin{aligned} \mathcal{L}_{\mathrm{Rep}}(F,B) &= \mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{D}, s'\sim\rho, z\sim\mathcal{Z}} \big|\tau - \mathbf{1}\{\Delta(F,B) < 0\}\big| \delta(F,B)^2 \\ \text{with} \quad \Delta(F,B) &:= r_z(s_t) + \gamma F(s_{t+1},z)^\top z - F(s_t,z)^\top z, \\ \delta(F,B) &:= \mathbf{1}\{s_t = s'\} + \gamma \overline{F}(s_{t+1},z)^\top \overline{B}(s') - F(s_t,z)^\top B(s'). \end{aligned} \tag{9}$$

Here, $r_z(s)$ is the reward associated with embedding z, defined as $r_z(s) = B(s)^\top (\mathbb{E}_\rho BB^\top/\rho)^{-1} z$. In practice, we enhance this loss with standard techniques such as double target networks, orthonormalization regularization, and improved sampling strategies for z and s', described in Appendix C.

Step 2: Learning high-level policy. Given successor representations, we learn a policy that plans over subgoals rather than actions. Using Corollary 1, we define an FB-approximation of $A^{\pi_w\to\pi_z}_s(r_z)$:

$$A_{\mathrm{FB}}(s,w,z) := F(s,z_w)^\top z + \frac{F(s,z_w)^\top z_w}{F(w,z_w)^\top z_w} \Big(F(w,z) - F(w,z_w)\Big)^\top z - F(s,z)^\top z \tag{10}$$

where $z_w = B(w)$ denotes the embedding of the subgoal w, and z is the reward embedding. We learn a high-level policy $\pi^h(z_w \mid s, z_r)$ by maximizing A_{FB} using the AWR objective from (3):

$$\mathcal{L}_{\mathrm{Plan}}(\pi^h) = -\mathbb{E}_{s,w\sim\mathcal{D}, z\sim\mathcal{Z}} \Big[\exp\Big(\beta A_{\mathrm{FB}}(s,w,z)\Big) \log \pi^h(z_w \mid s, z)\Big]. \tag{11}$$

Step 3: Learning low-level policy. Finally, we learn a low-level policy operating in the action space. As in Step 2, we adopt the AWR objective (3) and train the policy using one-step transitions. Concretely, the low-level policy $\pi^\ell(a_t \mid s_t, z)$ is learned by minimizing[3]:

$$\mathcal{L}_{\text{Act}}(\pi^\ell) = -\mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D} \\ z\sim\mathcal{Z}}}\Big[\exp\Big(\beta(F(s_{t+1},z)^\top z - F(s_t,z)^\top z)\Big)\log \pi^\ell(a_t \mid s_t, z)\Big]. \quad (12)$$

The final FB π-Switch objective is the sum of representation, planning, and control losses: $\mathcal{L}_{\text{Rep}}(F,B) + \mathcal{L}_{\text{Plan}}(\pi^h) + \mathcal{L}_{\text{Act}}(\pi^\ell)$. Further training details are provided in Appendix C.

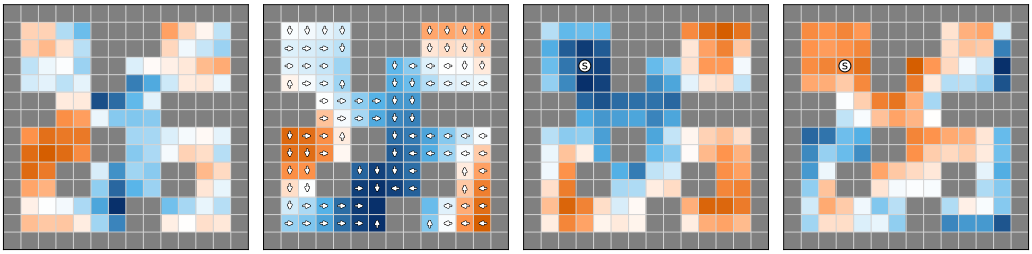
5.1 Hierarchical post-tuning for pretrained FB models

Although we present a full hierarchical algorithm, the hierarchical component can also be added on top of pretrained behavior foundational models (BFMs). Suppose we are given a pretrained successor representation $M^{\pi_z}_{s,a}(s') \approx F(s,a,z)^\top B(s')\rho(s')$ together with a latent-conditioned policy $\pi(a|s,z)$, as in prior work (Touati et al., 2023; Pirotta et al., 2023; Tirinzoni et al., 2025). In this case, hierarchy can be introduced by applying only Step 2 of our algorithm. Step 1 is implicitly satisfied since $M^{\pi_z}_{s}(s') = \mathbb{E}_{a\sim\pi(\cdot|s,z)} M^{\pi_z}_{s,a}(s')$, and Step 3 requires no additional training, as the pretrained policy $\pi(a|s,z)$ can be directly used as the low-level controller. However, pretraining a BFM is generally more challenging, since the representations are learned directly in the state-action space. A more detailed description is provided in Appendix C.3.

6 Experiments

We evaluate our hierarchical learning methods in three settings: discrete control, standard goal-conditioned continuous control, and continuous control with general reward functions.

An illustrative example: learning in a discrete maze. We consider the environment from Section 4 and learn representations by minimizing the representation loss (9) (check Appendix C for details). Figure 4 shows the learned counterparts of the quantities illustrated in Figure 2.



(a) $\text{Proj}_B r(s)$ (b) $F(s,z_r)^\top z_r$ (c) $A_{\text{FB}}(s,w,r)$ (d) $A_{\text{FB}}(s,w,r)|_{<w}$

Figure 4: Depicted quantities as functions of the magenta variables: (a) Reward projected onto learned features: $\text{Proj}_B r(s) = \langle r, B^\top(BB^\top)^{-1}B(s)\rangle$. (b) Learned value function for embedding $z_r = \langle r, B\rangle$. (c) Learned switching advantage (10). (d) Learned analogue of Figure 2(d), corresponding to the pre-hitting terms in (10).

This example highlights an important observation. As shown in Figure 4, the successor representations can be successfully learned from data and captures meaningful structure across rewards (a), values (b), and advantage-related quantities (c). However, the same example also reveals a challenge: as shown in Figure 4 (d), learning the local component of the advantage function is difficult, as it requires subtracting a term supported only on states with nonzero reward. To address this in practice, we use a proxy, $\widehat{A}_{\text{FB}}$, obtained by dropping the term $-\frac{F(s,z_w)^\top z_w}{F(w,z_w)^\top z_w}F(w,z_w)^\top z_r$ from (10). In addition, we omit the $\rho^{-1}(s)$ scaling when constructing reward embeddings, using

[3]Note that we omit the constant reward term from the objective and absorb the discount factor into parameter β.

AntMaze	HIQL		ICVF	One-Step FB	FB		FB π-Switch	
	vanilla	$-\pi^h$			vanilla	$+\pi^h$	vanilla	$-\pi^h$
Medium	93 ± 1	73 ± 3	46 ± 37	35 ± 7	47 ± 1	50 ± 12	87 ± 6	70 ± 4
Large	73 ± 6	23 ± 4	28 ± 3	23 ± 18	21 ± 5	20 ± 5	66 ± 9	37 ± 7
Giant	5 ± 3	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	1 ± 1	1 ± 1
Teleport	42 ± 4	32 ± 6	23 ± 6	8 ± 2	25 ± 7	13 ± 7	40 ± 6	29 ± 4

Table 1: Average (binary) success rate (%) across the five test-time goals with `navigate-v0` datasets. The results are averaged over 50 episodes and 5 seeds, and the standard deviations are reported after $\pm$ sign.

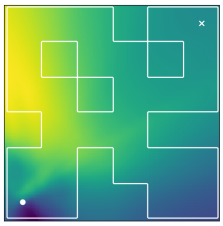
$z_r = \mathbb{E}_{s\sim\rho}[r(s)B(s)]$ instead[4]. Additional results, including a comparison between A_{FB} and $\widehat{A}_{\mathrm{FB}}$, are provided in Appendix D.1.

Overall, this example demonstrates that FB representations can be learned in practice, while also highlighting the difficulty of accurately estimating the local component of the advantage function. In subsequent experiments, we therefore omit the subtracted term when training the high-level policy.

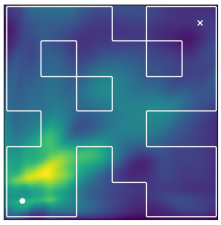
Continuous control for goal-reaching tasks. Next, we evaluate our method on AntMaze from OGBench (Park et al., 2025a), a standard GCRL benchmark with 29-dimensional states and 8-dimensional actions. We note that this is not purely a navigation problem; the agent must also learn a complex locomotion policy in addition to planning paths through the maze. Although our framework can be applied to more general tasks than goal-reaching, this setting provides a controlled and widely used environment for comparison. We compare against HIQL, ICVF, FB, and One-Step FB (Zheng et al., 2026). For FB π-Switch, we evaluate both hierarchical and non-hierarchical variants (without the high-level policy π^h). We also include a hierarchical variant of FB (Section 5.1), denoted FB $+\pi^h$. Additional methodological comparisons are provided in Appendix A.4.

All methods are trained under a common setup with sparse rewards and no terminal masking. Models use comparable architectures and are trained for 1M steps. Hierarchical variants are initialized from pretrained non-hierarchical models and trained for an additional 500k steps, updating only the high-level policy. This enables direct comparison between variants sharing the same representation. Evaluation is performed on five standard tasks per maze, with episodes of length 1000 (App. C).

Table 1 reports success rates across four environments. FB π-Switch achieves performance comparable to HIQL, despite not being specifically designed for GCRL. Adding a high-level policy consistently improves performance, highlighting the benefit of hierarchical structure. Notably, even without hierarchy, FB π-Switch outperforms other non-hierarchical baselines. All methods struggle in the Giant setting due to long horizons and sparse rewards. Since performance depends on learned representations, sufficiently strong representations are essential.



(a) HIQL



(b) FB π-Switch

Figure 5: Comparison of π^h: our method induces optimal path subgoals.

To better understand these results, we ablate HIQL by (i) removing the high-level policy, (ii) replacing its critic with a bilinear form, and (iii) using intention-based values $V^{\pi_g}(s; s')$. These modifications progressively align HIQL with FB π-Switch without hierarchy, yielding comparable performance (Appendix D.2). This suggests that these adaptations do not significantly affect performance in GCRL settings.

We also observe qualitative differences in learned high-level policies (Figure 5). HIQL uses representations conditioned on both state and goal, learning subgoals that lead to similar immediate actions. In contrast, our method uses goal-only representations, yielding subgoals that lie along coherent trajectories toward the goal. This results in more structured, path-consistent behavior (see Appendix D.2 for more visualizations).

[4]Note that all our loss functions are invariant to scaling of z.

Finally, we analyze the role of hierarchy in FB-based methods. While adding a high-level policy to standard FB yields only marginal improvements, we find that this is not due to a lack of meaningful subgoals. To isolate the effect, we construct hybrid agents combining high- and low-level policies from FB and FB π-Switch. Results in Table 2 show that hierarchical FB can learn useful subgoals, and that the limited gains are primarily due to the quality of the low-level policy rather than the hierarchical decomposition itself.

π^h	π^ℓ	Medium	Large	Teleport
FB+π^h	FB π-Switch	+30%	+90%	+74%
FB π-Switch	FB+π^h	+19%	−23%	−14%

Table 2: Relative improvement in success rate (%) over the base FB $+\pi^h$ model when combined with high- or low-level policies of FB π-Switch.

Continuous control for general tasks. Motivated by the discrete example in Figure 2, we extend the AntMaze benchmark by defining reward functions over multiple spatial regions instead of a single goal, inducing trade-offs between local and global objectives. This yields a controlled yet more general setting with diverse reward landscapes (Figure 7). We focus on the Large and Giant mazes, where long-horizon structure is critical. Using the models from the previous section, we normalize returns and, following Agarwal et al. (2021), report the interquartile mean (IQM) to aggregate performance across tasks, with stratified bootstrap 95% confidence intervals in Figure 6. Full results available in Appendix D.3.

FB π-Switch achieves the best average performance. In the Giant setting, the non-hierarchical variant also performs well, likely due to strong local rewards guiding behavior. Notably, FB π-Switch remains robust across both environments, suggesting that the learned high-level policy improves robustness. We observe that not all hierarchical variants are equally effective, reinforcing that the quality of the low-level policy is crucial for realizing the benefits of hierarchy.

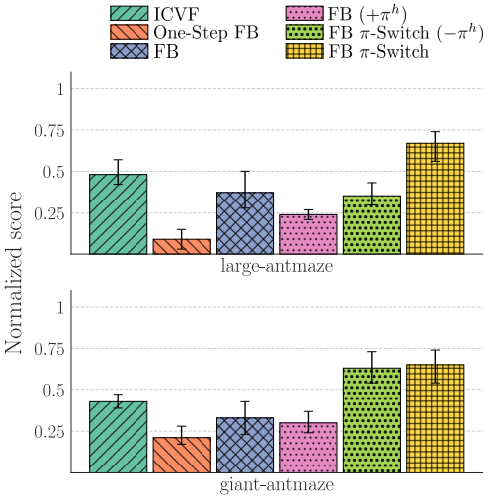


Figure 6: Normalized returns across the five tasks in Figure 7, evaluated over 50 episodes (1000 steps) and 5 seeds per task. We report IQM with stratified bootstrap 95% confidence intervals.

Importantly, unlike goal-conditioned methods such as HIQL that that are limited to indicator rewards, our framework naturally extends to distributed rewards, enabling composition across regions and improved performance on more general tasks.

7 Conclusion

In this work, we introduced switching successor measures as a framework for modeling hierarchical structure in zero-shot RL, and proposed FB π-Switch, a method that jointly learns high- and low-level policies from shared representations. Without relying on expert trajectories or hand-crafted temporal abstractions, it enables hierarchical behavior to emerge directly from the representations. Empirical results across goal-reaching and more general tasks show that FB π-Switch achieves competitive performance and consistently improves over non-hierarchical baselines.

Our findings suggest that hierarchical learning with successor representations can effectively bridge local and global decision-making, but several open questions remain. Future work could explore the dependence of hierarchical behavior on the quality of the base model, and which task properties, such as reward sparsity and long horizons, most strongly favor this hierarchical structure. Another direction is comparing this approach to generative model-based methods, which often excel in long-horizon planning but introduce additional complexity. Finally, extending and evaluating our framework in multi-subgoal planning remains an interesting open question.

Acknowledgments

We thank Bastien Dubail for providing helpful feedback on the proof of the main theorem. This research was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, the Swedish Research Council (VR), and Digital Futures. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021.

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an RL agent. *arXiv preprint arXiv:2411.19418*, 2024.

Siddhant Agarwal, Caleb Chuck, Harshit Sikchi, Jiaheng Hu, Max Rudolph, Scott Niekum, Peter Stone, and Amy Zhang. A unified framework for unsupervised reinforcement learning algorithms. In *Workshop on Reinforcement Learning Beyond Rewards@ Reinforcement Learning Conference*, 2025.

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*, volume 38, pp. 99833–99861, 2025.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. Provably efficient Q-learning with low switching cost. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via auto-regressive features and advantage weighting. In *Reinforcement Learning Conference*, 2025.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 1430–1440. PMLR, 2021.

Chang Chen, Junyeob Baek, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. PlanDQ: Hierarchical plan orchestration via D-conductor and Q-performer. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 6397–6412. PMLR, 2024.

Jinwoo Choi, Sang-Hyun Lee, and Seung-Woo Seo. Chain-of-goals hierarchical policy for long-horizon offline goal-conditioned RL. *arXiv preprint arXiv:2602.03389*, 2026.

Gheorghe Comanici and Doina Precup. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pp. 709–714, 2010.

Bastien Dubail, Stefan Stojanovic, and Alexandre Proutière. Shift before you learn: Enabling low-rank representations in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 72243–72305, 2025.

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

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

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Marc G Bellemare, Alessandro Lazaric, and Ahmed Touati. Compositional planning with jumpy world models. In *ICLR 2026 the 2nd Workshop on World Models: Understanding, Modelling and Scaling*, 2026.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 13927–13942. PMLR, 2024.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 11321–11339. PMLR, 2023.

Vittorio Giammarino, Ruiqi Ni, and Ahmed H Qureshi. Physics-informed value learner for offline goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 137769–137796, 2025.

Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. In *Reinforcement Learning Conference*, 2025.

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.

Nicklas Hansen, Jyothir SV, Vlad Sobal, Yann LeCun, Xiaolong Wang, and Hao Su. Hierarchical world models as visual whole-body humanoid controllers. In *International Conference on Learning Representations*, 2025.

Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. Successor feature landmarks for long-horizon goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26963–26975, 2021.

RenMing Huang, Shaochong Liu, Yunqiang Pei, Peng Wang, Guoqing Wang, Yang Yang, and Hengtao Shen. Goal-reaching policy learning from non-expert observations via effective subgoal guidance. In *Proceedings of the 8th Conference on Robot Learning*, volume 270, pp. 1744–1762. PMLR, 2024.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1724–1735, 2020.

Scott Jeen, Tom Bewley, and Jonathan Cullen. Zero-shot reinforcement learning from low quality data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 16894–16942, 2024.

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.

Martin Klissarov, Akhil Bagaria, Ziyan Luo, George Konidaris, Doina Precup, and Marlos C Machado. Discovering temporal structure: An overview of hierarchical reinforcement learning. *arXiv preprint arXiv:2506.14045*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. DDCO: Discovery of deep continuous options for robot learning from demonstrations. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 418–437. PMLR, 2017.

Richard Liaw, Sanjay Krishnan, Animesh Garg, Daniel Crankshaw, Joseph E Gonzalez, and Ken Goldberg. Composing meta-policies for autonomous driving using hierarchical deep reinforcement learning. *arXiv preprint arXiv:1711.01503*, 2017.

Tao Ma, Xuzhi Yang, and Zoltan Szabo. To switch or not to switch? Balanced policy switching in offline reinforcement learning. *arXiv preprint arXiv:2407.01837*, 2024.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 37076–37096. PMLR, 2024.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Natinael Solomon Neggatu, Jeremie Houssineau, and Giovanni Montana. Evaluation-time policy switching for offline reinforcement learning. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1520–1528, 2025.

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

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 39737–39761. PMLR, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025a.

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, 2025b.

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*, 36(4):1–13, 2017.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731*, 2019.

Alfredo Reichlin, Miguel Vasco, Hang Yin, and Danica Kragic. Learning goal-conditioned policies from sub-optimal offline data via metric learning. *arXiv preprint arXiv:2402.10820*, 2024.

Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205, pp. 1838–1849. PMLR, 2022.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1312–1320. PMLR, 2015.

Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205, pp. 2262–2272. PMLR, 2022.

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models. In *Reinforcement Learning Conference*, 2025.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 43905–43941, 2025.

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.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 21272–21307. PMLR, 2022.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Learning Representations*, 2025.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13–23, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations*, 2023.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *Advances in Neural Information Processing Systems*, volume 38, pp. 157643–157670, 2025.

Chengjie Wu, Hao Hu, Yiqin Yang, Ning Zhang, and Chongjie Zhang. Planning, fast and slow: Online reinforcement learning with action-free offline data via multiscale planners. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 53515–53541. PMLR, 2024.

Grace Zhang, Ayush Jain, Injune Hwang, Shao-Hua Sun, and Joseph J Lim. QMP: Q-switch mixture of policies for multi-task behavior sharing. In *International Conference on Learning Representations*, 2025.

Chongyi Zheng, Royina Karegoudra Jayanth, and Benjamin Eysenbach. Can we really learn one representation to optimize all rewards? *arXiv preprint arXiv:2602.11399*, 2026.

Kexin Zheng, Lauriane Teyssier, Yinan Zheng, Yu Luo, and Xianyuan Zhan. Towards robust zero-shot reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 131049–131084, 2025.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Additional discussions

A.1 Further related work

In this section, we highlight additional works related to our approach. For a more comprehensive overview, we refer the reader to the survey on hierarchical RL by Klissarov et al. (2025).

Goal-conditioned RL. Many goal-conditioned RL methods build on hindsight experience replay (Andrychowicz et al., 2017), which relabels states encountered along trajectories as goals to address sparse rewards. Successor features have also been applied in goal-conditioned RL (Ramesh et al., 2019; Hoang et al., 2021). Another line of work introduces metric (Reichlin et al., 2024) or quasi-metric (Myers et al., 2024) structure into value representations to improve robustness. Option-based extensions of HIQL (Ahn et al., 2025) reduce the effective horizon by minimizing TD-error over multi-step targets rather than one-step transitions. Other approaches focus on online adaptation of learned goal-based hierarchies (Wu et al., 2024).

Successor measures. Generative formulations of successor measures have been studied under the name of γ-models (Janner et al., 2020). Empirical work has demonstrated that FB models can produce diverse behaviors in complex domains such as humanoid control (Tirinzoni et al., 2025). Other studies investigate the impact of dataset quality on FB methods, showing sensitivity to limited or homogeneous data (Jeen et al., 2024). Alternative representations of successor measures have been explored beyond the FB parameterization, see Agarwal et al. (2025) for an overview.

Hierarchical methods. Another line of work develops hierarchical skill models using explicit dynamics or world models (Shi et al., 2022; Hansen et al., 2025; Gürtler & Martius, 2025), where high-level policies operate over predicted skill outcomes. Related directions include model-based planning (Sobal et al., 2025) and autoregressive generation of subgoal sequences (Choi et al., 2026). Hierarchical structure has also been explored in imitation learning, for example through options (Krishnan et al., 2017) or latent skill models that combine imitation learning with offline RL for skill composition (Rosete-Beas et al., 2022).

Policy switching. Policy switching has been studied in several contexts. Thakoor et al. (2022) derive a generalized policy improvement result under a geometric horizon model. Other work considers switching between agents at test time (Neggatu et al., 2025) or learning switching mechanisms over multiple policies (Zhang et al., 2025). Additional directions analyze the cost or frequency of switching in exploration (Bai et al., 2019) and offline learning (Ma et al., 2024). However, these approaches do not study switching in the context of learned successor representations or hierarchical subgoal-based policies.

A.2 Further remarks on the FB representation

Here we recall key properties of the FB representation of Touati et al. (2023). The (state-action) successor measure is defined as $M^{\pi_z}_{s,a}(s') = F(s,a,z)^\top B(s')\rho(s')$ and its corresponding policy as $\pi_z(s) = \arg\max_a F(s,a,z)^\top z$. According to Theorem 2 in Touati et al. (2023), for any reward function r and $z_r = \mathbb{E}_\rho[r(s)B(s)]$, the optimal Q-function can be written as $Q^\star_r(s,a) = F(s,a,z_r)^\top z_r$. This shows that the FB representation can simultaneously express: (i) successor measures $M^{\pi_z}_{s,a}(s')$ through $F(s,a,z)^\top B(s')\rho(s')$, and (ii) optimal Q-values for arbitrary reward functions $Q^\star_r(s,a)$ through $F(s,a,z_r)^\top z_r$. Both are based on the same linear structure in the feature space spanned by $B(\cdot)$. Our framework builds on this observation by approximating both (state) successor measure and (state) value function using the FB parameterization.

Next, we provide additional remarks regarding the loss functions of the FB algorithm. First, an orthonormalization loss is added in order to regularize backward representation:

$$\mathcal{L}_{\text{norm}}(B) = \|\mathbb{E}_{s\sim\rho} B(s)B(s)^\top - \text{Id}\|_{\text{F}}^2 = \mathbb{E}_{s,s'\sim\rho}[(B(s)^\top B(s'))^2 - 2\|B(s)\|_2^2] + \text{Const},$$

where Id is the identity matrix. An auxiliary loss on forward representations has been proposed in Touati et al. (2023), however we do not include it since prior work reports it has little impact.

Next, we explain derivation of the loss function in (1). Given a state distribution ρ, define the norm $\|M^\pi\|_\rho^2 := \mathbb{E}_{(s,a)\sim\mathcal{D},s'\sim\rho}\big(M_{s,a}^\pi(s')/\rho(s')\big)^2$. For simplicity, we assume a finite state-space, which allows using point values such as $\rho(s')$ and indicator functions $\mathbf{1}\{s_t = s'\}$. As discussed in Touati et al. (2023), this extends to continuous spaces with appropriate measure-theoretic definitions.

Touati et al. (2023) defines the objective $\mathcal{L}(F,B) := \big\|F^\top B\rho - \big(\text{Id} + \gamma P_\pi F^\top B\rho\big)\big\|_\rho^2$, which expands to the FB loss used in our work (Equation (1)):

$$\mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D}\\ s'\sim\rho, z\sim\mathcal{Z}}}\Big(\frac{\mathbf{1}\{s_t = s'\}}{\rho(s')} + \gamma\overline{F}(s_{t+1},\pi_z(s_{t+1}),z)^\top\overline{B}(s') - F(s_t,a_t,z)^\top B(s')\Big)^2$$

Finally, under a quadratic objective, this loss function simplifies (up to a constant) to:

$$\mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D}\\ s'\sim\rho, z\sim\mathcal{Z}}}\Big[\big(\gamma\overline{F}(s_{t+1},\pi_z(s_{t+1}),z)^\top\overline{B}(s') - F(s_t,a_t,z)^\top B(s')\big)^2 - 2F(s_t,a_t,z)^\top B(s_t)\Big]$$

Remark: Touati et al. (2023) defines rewards and successor measures with respect to the next state s_{t+1}, whereas we use the more common convention based on the current state s_t. This leads to a slight difference in the loss: their formulation replaces the identity operator with the transition operator P, and introduces a term $F(s_t,a_t,z)^\top B(s_{t+1})$ instead of $F(s_t,a_t,z)^\top B(s_t)$ in the final expression.

A.3 Analysis of Corollary 1 from the perspective of prior work

We begin by recalling the result from Corollary 1:

$$A_s^{\pi_w\to\pi}(r) = V^{\pi_w}(s;r) + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}\Big(V^\pi(w;r) - V^{\pi_w}(w;r)\Big) - V^\pi(s;r).$$

We now relate this expression to two closely related lines of work.

Simplified objectives for GCRL in Park et al. (2023). As discussed earlier in Section 4, HIQL adopts a simplified advantage-based formulation. For comparison, consider a hypothetical variant of HIQL in which subgoal horizons are not fixed but sampled as $k \sim \text{Geom}(1-\gamma)$. In this case, the HIQL objective can be interpreted as maximizing the advantage:

$$A(s,w,r) = (1-\gamma)M_s^{\pi_\beta}(w)V^\pi(w;r) - V^\pi(s;r),$$

where π_β denotes the behavior policy used to collect the offline dataset. Compared to Corollary 1, this formulation omits the local term $V^{\pi_w}(s;r) - \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)}V^{\pi_w}(w;r)$, which is constant in the goal-conditioned setting and is equal to zero under sparse rewards, since only the goal state carries non-zero reward. Moreover, HIQL assumes absorbing goal states, which implies $M_w^{\pi_w}(w) = 1/(1-\gamma)$. Finally, the term $M_s^{\pi_w}(\cdot)$ is effectively approximated using sampled future states $s_{t+k} \sim (1-\gamma)M_s^{\pi_\beta}(\cdot)$, corresponding to the future states of the behavior policy rather than those induced by the optimal goal-reaching policy for subgoal w.

Sampling successor measure via generative models in Farebrother et al. (2026). We next compare our formulation with compositional policy evaluation using generative successor models. Here, we explicitly denote the dependence on the discount factor γ using $V^{\pi|\gamma}(s;r)$ and $M_s^{\pi|\gamma}(w)$.

Consider Lemma 1 in Farebrother et al. (2026) (also reported in Thakoor et al. (2022)), for a compositional policy $\pi = \pi_{z_1} \xrightarrow{\alpha} \pi_{z_2}$, where the first policy π_{z_1} uses a discount factor $\gamma(1-\alpha)$ and the second uses γ. For $n = 2$, an unbiased estimate of the value function is given by:

$$\widehat{V}^{\pi|\gamma}(s;r) = \frac{1}{1-\gamma(1-\alpha)}\left[r(s_1^+) + \frac{\alpha\gamma}{1-\gamma} r(s_2^+)\right],$$

where[5] $s_1^+ \sim M_s^{\pi_{z_1}|\gamma(1-\alpha)}$ and $s_2^+ \sim M_{s_1^+}^{\pi_{z_2}|\gamma}$. The expectation of this estimate is:

$$\begin{aligned}\frac{1}{1-\gamma(1-\alpha)}&\mathbb{E}_{w\sim M_s^{\pi_{z_1}|\gamma(1-\alpha)}}\left[r(w) + \frac{\alpha\gamma}{1-\gamma}\mathbb{E}_{s'\sim M_w^{\pi_{z_2}|\gamma}} r(s')\right] \\ &= V^{\pi_{z_1}|\gamma(1-\alpha)}(s;r) + \alpha\gamma\sum_w M_s^{\pi_{z_1}|\gamma(1-\alpha)}(w) V^{\pi_{z_2}|\gamma}(w;r). \end{aligned} \tag{13}$$

This expression is structurally similar to Corollary 1. However, there are several key differences. First, we follow π_w until reaching w instead of following the first policy with geometrically discounted horizon parameterized by $\gamma(1-\alpha)$. Second, since switching occurs upon reaching w, the sum over all subgoals w in (13) collapses to a single non-zero term. As a result, our method does not require a generative model to estimate the switching advantage function.

A.4 Further comparisons with prior methods

Intention conditioned value functions (ICVF) (Ghosh et al., 2023). The first stage of our algorithm is closely related to the representation learning stage of ICVF. However, ICVF assumes a structured factorization of the form $M_s^{\pi_z}(g) = \phi(s)^\top T(z)\psi(g)$, which corresponds to the factorization $F(s,z) = \phi(s)^\top T(z)$ and $B(g) = \psi(g)$, where $T(z) \in \mathbb{R}^{d\times d}$ is a learned matrix. In practice, Ghosh et al. (2023) restricts intentions z to goal states, effectively reducing the method to GCRL setting. Moreover, ICVF is not a zero-shot method: its representation learning stage (corresponding to our Stage 1) is used solely to learn state embeddings ϕ, after which a separate offline RL algorithm is trained for each downstream reward function. In contrast, our approach learns a single value representation that directly supports inference of both high- and low-level policies without additional RL training. Finally, ICVF does not incorporate a hierarchical structure, as it does not model a high-level policy over intentions.

Functional reward encodings (FRE) (Frans et al., 2024). Another promising direction in zero-shot RL is learning compact, informative reward representations. FRE addresses this by learning reward embeddings via transformer-based variational autoencoders, trained using a variational lower bound on mutual information. Importantly, this representation learning stage uses only sampled state–reward pairs and does not involve value functions or RL objectives. In a second stage, the learned embeddings are used as conditioning variables for standard offline RL algorithms such as IQL to train policies for downstream tasks. This approach is largely orthogonal to our framework: while FRE focuses on learning expressive reward embeddings as a separate preprocessing step, we directly exploit learned successor measures to infer both reward structure and high- and low-level policies within a single framework. Moreover, our method is significantly simpler in terms of architecture, relying on standard function approximators such as MLPs, whereas FRE requires more expressive sequence models such as transformer-based autoencoders.

Hilbert foundation policies (HILP) (Park et al., 2024). Similar to the previous methods, HILP follows a two-stage pipeline: first learning state representations used to construct reward embeddings, and then applying offline RL to train policies conditioned on these embeddings. Interestingly,

[5] Note that $M_s^{\pi|\gamma}$ is a proper probability distribution in our framework only after scaling by $(1-\gamma)$.

HILP also employs expectile regression, but assumes that value functions can be represented as Euclidean distances in a latent Hilbert space. However, the learned value functions are restricted to goal-aligned settings, i.e., $V^{\pi_g}(s; r_g)$, where policy and reward correspond to the same goal. This structure enables a simple form of hierarchical planning by selecting midpoints w that minimize distances $s \to w$ and $w \to g$ as subgoals. In practice, this is implemented by sampling N candidate subgoals from the dataset and choosing the one closest to both the state and the goal. Since HILP does not explicitly leverage learned value functions or successor measures, its sample efficiency remains unclear: each iteration requires sampling a new reward embedding z and training a separate policy π_z using an offline RL algorithm. In contrast, our approach (and related FB-based methods) trains policies for all embeddings π_z in parallel by sharing a unified successor representation.

Regularizing FB models with value expectile regression (Zheng et al., 2025). Recent work by Zheng et al. (2025) proposes several modifications to the FB algorithm aimed at improving robustness. In addition to architectural changes such as diffusion-based policy extraction and attention-based representation learning, they also introduce a mechanism to mitigate extrapolation error. Specifically, they use expectile regression to learn state-value estimates $V^{\pi_z}(s; z)$ by regressing them toward $F(s, a, z)^\top z$. These learned values are then used as targets in the auxiliary FB loss, replacing the original term based on $Q^{\pi_z}(s', \pi_z(s'); z)$, which improves stability during training. While our framework also learns state-value functions, there are key differences. First, Zheng et al. (2025) retains an action-conditioned FB representation of the form $F(s, a, z)^\top B(g)$ and additionally learns state-value functions on top of it. In contrast, we only model state-value functions and do not explicitly parameterize Q-functions. Second, we do not employ the auxiliary FB loss; our representations are learned solely using the main objective.

One-step FB learning (Zheng et al., 2026). The key idea in Zheng et al. (2026) is to avoid learning successor measures for every policy π_z and instead focus on estimating it for the behavior policy alone. Policies conditioned on z are then obtained via one-step policy improvement using the successor measure induced by the behavior policy. This design simplifies representation learning by allowing direct use of next actions a_{t+1} from the dataset, avoiding the need to bootstrap through $\pi_z(s_{t+1})$. The resulting policy is extracted as a softmax over actions with logits given by $F_\beta(s, a)^\top z$, where β denotes the behavior policy, corresponding to $Q^{\pi_\beta}(s, a; r_z)$. While this approach simplifies the FB pipeline, it is not directly applicable to our setting. Our framework requires estimates of $V^{\pi_w}(s; r)$ for intermediate policies, which cannot be recovered from representations restricted to the behavior policy alone.

B Theoretical results

We first provide a short derivation of Corollary 1 from Theorem 1 for completeness. Recall the definitions of the switching successor measure in (7) and the switching advantage in (8):

$$M_s^{\pi_w \to \pi}(s') = \mathbb{E}\left[M_s^{\pi_w|_k \to \pi}(s')\mathbf{1}\{k = H_s^{\pi_w}(w)\}\right], \; A_s^{\pi_w \to \pi}(r) = \mathbb{E}\left[A_s^{\pi_w|_k \to \pi}(r)\mathbf{1}\{k = H_s^{\pi_w}(w)\}\right].$$

Using the standard relation between value functions and successor measures, namely $A_s^{\pi_w|_k \to \pi}(r) = \langle M_s^{\pi_w|_k \to \pi} - M_s^{\pi}, r\rangle$, we obtain:

$$\begin{aligned}
A_s^{\pi_w \to \pi}(r) &= \mathbb{E}\left[\langle M_s^{\pi_w|_k \to \pi} - M_s^{\pi}, r\rangle \mathbf{1}\{k = H_s^{\pi_w}(w)\}\right] \\
&= \mathbb{E}\left[\langle M_s^{\pi_w|_k \to \pi}, r\rangle \mathbf{1}\{k = H_s^{\pi_w}(w)\}\right] - \langle M_s^{\pi}, r\rangle \\
&= \sum_{s'} \mathbb{E}\left[M_s^{\pi_w|_k \to \pi}(s')\mathbf{1}\{k = H_s^{\pi_w}(w)\}\right] r(s') - V^{\pi}(s; r) \\
&= \sum_{s'} M_s^{\pi_w \to \pi}(s') r(s') - V^{\pi}(s; r)
\end{aligned}$$

Applying Theorem 1 to the first term yields:

$$\begin{aligned}
\sum_{s'} M_s^{\pi_w \to \pi}(s') r(s') &= \sum_{s'} M_s^{\pi_w}(s') r(s') + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)} \Big(\sum_{s'} M_w^{\pi}(s') r(s') - \sum_{s'} M_w^{\pi_w}(s') r(s') \Big) \\
&= V^{\pi_w}(s; r) + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)} \Big(V^{\pi}(w; r) - V^{\pi_w}(w; r) \Big)
\end{aligned}$$

which completes the proof of Corollary 1.

Before proving Theorem 1, we recall the definition of hitting time from Section 3: $H_s^{\pi}(w) = \min\{t \geq 0 : s_t = w | s_0 = s, \pi\}$. We use the following result:

Lemma 1 (Lemma B.1 in Myers et al. (2024))**.** *We have for any $s, w \in \mathcal{S}$ and any policy π:*

$$M_s^{\pi}(w) = \mathbb{E}\left[\gamma^{H_s^{\pi}(w)}\right] M_w^{\pi}(w)$$

It is also useful to contrast Theorem 1 with prior work. In particular, Lemma 3.3 of Myers et al. (2024) shows that:

$$M_s^{\pi_w \to \pi}(s') \geq \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)} M_w^{\pi}(s').$$

This provides a lower bound on the switching successor measure, where the term on the right hand side captures discounted visitation after reaching the subgoal w. In contrast, Theorem 1 establishes an equality, expressing the switching successor measure exactly in terms of standard successor measures.

B.1 Proof of Theorem 1

Proof. We construct a (generally non-Markovian) policy, denoted $\overline{\pi}$, that follows the subgoal policy π_w until the state w is reached, and then switches to π afterwards:

$$\overline{\pi}(a_t|s_t) = \begin{cases} \pi(a_t|s_t), & w \in \{s_0, s_1, \ldots, s_t\} \\ \pi_w(a_t|s_t), & \text{otherwise.} \end{cases}$$

Let $H_s^{\overline{\pi}}(w)$ denote the hitting time of state w under policy $\overline{\pi}$ starting from s. We can assume the hitting time is finite. If w is never reached, the result holds trivially, since in that case $\overline{\pi} \equiv \pi_w$ and $M_s^{\pi_w}(w) = 0$. By construction, $H_s^{\overline{\pi}}(w) = H_s^{\pi_w}(w)$ almost surely for all s.

Using the law of total probability and definition (7) of switching successor measure, we have:

$$M_s^{\overline{\pi}}(s') = \sum_{k=0}^{\infty} \mathbb{P}(H_s^{\pi_w}(w) = k)\mathbb{E}\Big[M_s^{\pi_w|_k \to \pi}(s')\Big| H_s^{\pi_w}(w) = k\Big]. \tag{14}$$

We now analyze the inner expectation:

$$\begin{aligned}
\mathbb{E}\Big[M_s^{\pi_w|_k \to \pi}(s')\Big| H_s^{\pi_w}(w) = k\Big] &= \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\overline{\pi}}(s_t = s' | s_0 = s, H_s^{\pi_w}(w) = k) \\
&= \sum_{t=0}^{k-1} \gamma^t \mathbb{P}_{\pi_w}(s_t = s' | s_0 = s, H_s^{\pi_w}(w) = k) + \sum_{t=k}^{\infty} \gamma^t \mathbb{P}_{\pi}(s_t = s' | s_0 = s, H_s^{\pi_w}(w) = k) \\
&= \sum_{t=0}^{k-1} \gamma^t \mathbb{P}_{\pi_w}(s_t = s' | s_0 = s, H_s^{\pi_w}(w) = k) + \gamma^k \underbrace{\sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi}(s_{t+k} = s' | s_k = w)}_{M_w^{\pi}(s')}.
\end{aligned} \tag{15}$$

where for the first term we use that before reaching w the agent follows π_w, and for the second term we use the stationary property of MDPs. Now, we consider only the part in (14) corresponding to the second term:

$$\sum_{k=0}^{\infty} \mathbb{P}(H_s^{\pi_w}(w) = k)\gamma^k M_w^{\pi}(s') = \mathbb{E}\Big[\gamma^{H_s^{\pi_w}(w)}\Big]\ M_w^{\pi}(s')$$

This closely resembles the result in the proof of Lemma 3.3 in Myers et al. (2024). Next, we consider the first term corresponding to $t < k$ in (15). We have:

$$\begin{aligned}
&\sum_{k=0}^{\infty} \mathbb{P}(H_s^{\pi_w}(w) = k) \sum_{t=0}^{k-1} \gamma^t \mathbb{P}_{\pi_w}(s_t = s' \mid s_0 = s, H_s^{\pi_w}(w) = k) \\
&= \sum_{k=0}^{\infty} \sum_{t=0}^{k-1} \gamma^t \mathbb{P}_{\pi_w}(s_t = s', H_s^{\pi_w}(w) = k \mid s_0 = s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi_w}(s_t = s', H_s^{\pi_w}(w) > t \mid s_0 = s) = (\star),
\end{aligned}$$

where we apply Fubini's theorem. We then decompose the joint probability:

$$\mathbb{P}_{\pi_w}(s_t = s', H_s^{\pi_w}(w) > t \mid s_0 = s) = \mathbb{P}_{\pi_w}(s_t = s' \mid s_0 = s) - \mathbb{P}_{\pi_w}(s_t = s', H_s^{\pi_w}(w) \leq t \mid s_0 = s).$$

Summing over time steps gives:

$$\begin{aligned}
(\star) &= \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi_w}(s_t = s' \mid s_0 = s) - \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi_w}(s_t = s', H_s^{\pi_w}(w) \leq t \mid s_0 = s) \\
&= M_s^{\pi_w}(s') - \mathbb{E}\Big[\gamma^{H_s^{\pi_w}(w)}\Big]\ M_w^{\pi_w}(s')
\end{aligned}$$

where the second term was treated analogously as in equations leading to (15), since the hitting time precedes the current time.

Next, we invoke Lemma 1 to obtain $\mathbb{E}\big[\gamma^{H_s^{\pi_w}(w)}\big] = M_s^{\pi_w}(w)/M_w^{\pi_w}(w)$ for any w for which $M_w^{\pi_w}(w) > 0$. Note that $M_w^{\pi_w}(w) \geq 1$ under the standard definition of successor measures including the $t = 0$ term, and is therefore strictly positive. Summing the two components in (15) gives:

$$M_s^{\overline{\pi}}(s') = M_s^{\pi_w}(s') - \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)} M_w^{\pi_w}(s') + \frac{M_s^{\pi_w}(w)}{M_w^{\pi_w}(w)} M_w^{\pi}(s')$$

Finally, identifying $\overline{\pi}$ with $\pi_w \to \pi$ completes the proof. □

C Experimental details

C.1 Environment setups and datasets

Discrete environment.

We provide additional details on the environment and experiments shown in Figures 2 and 4. We consider a discrete variant of the Medium maze environment from OGBench, consisting of 104 states and 5 actions (stay or move in one of the four cardinal directions), with deterministic transitions and discount factor $\gamma = 0.98$. The reward function used in these experiments is illustrated in Figure 2(a).

In Figure 2, we visualize quantities computed exactly from the known dynamics. In particular, panel (b) shows the exact value function under the optimal policy for this reward function. Since the environment is small and fully discrete, these quantities can be computed exactly. This also allows us to empirically validate the results of Theorem 1 and Corollary 1 in this setting.

For Figure 4, we use the same environment but replace exact computation with a learning-based approach. Specifically, we apply the representation learning component of FB π-Switch to learn successor measures and visualize the learned quantities. Evaluation in this setting is qualitative, based on comparisons between learned quantities and their exact counterparts.

Given the low-dimensional state and action spaces, we use a latent dimension of 24, which is sufficient for this setting. We construct an offline dataset of 100,000 trajectories, each with a maximum length of 100 steps, collected using a uniformly random policy. Training is performed for 250 epochs, each consisting of 1000 gradient steps, with a batch size of 32. We use the Adam optimizer with learning rate 10^{-3}. During training, we gradually decrease the proportion of latents derived from states relative to those sampled uniformly from a sphere. These experiments are implemented in PyTorch and executed on CPU.

Continuous environments.

In Section 6, we use AntMaze environments from OGBench (Park et al., 2025a). Specifically, we consider the Medium, Large, Giant, and Teleport variants, which share the same 29-dimensional state space and 8-dimensional action space, with actions bounded in $[-1, 1]$, but differ in maze size and task difficulty. We use raw observations without normalization.

We use the `antmaze-giant-navigate-v0` dataset from OGBench. This dataset is generated by a noisy expert policy that repeatedly navigates to randomly sampled goals, resulting in diverse trajectories that cover the state space, and is commonly used as a standard benchmark dataset (Park et al., 2025a; Zheng et al., 2026). For the Medium, Large, and Teleport environments, episodes have a maximum length of 1000 steps, and the dataset consists of 1000 episodes (approximately 1M transitions). For the Giant environment, we use 500 episodes of length 2000 (also approximately 1M transitions). In our experiments, episodes are truncated at 1000 steps. In goal-conditioned evaluations, episodes terminate early upon reaching a neighborhood of the goal, defined as entering a radius of 0.5 in the (x,y) plane around the goal.

For the general reward setting, we design task-specific reward functions as shown in Figure 7. These tasks are defined over a discretized grid of the maze, with rewards taking values in $\{0, +1, -1, +5, +10\}$. Typically, each task includes a high-reward region, along with additional regions of moderately positive or negative reward. A reward is assigned whenever the agent enters a grid cell with a non-zero reward. We restrict this setting to the Large and Giant environments. The Medium environment is excluded due to its small size, which would lead to overly dense rewards when multiple reward regions are introduced. The Teleport environment is also omitted, as the resulting behaviors are less interpretable.

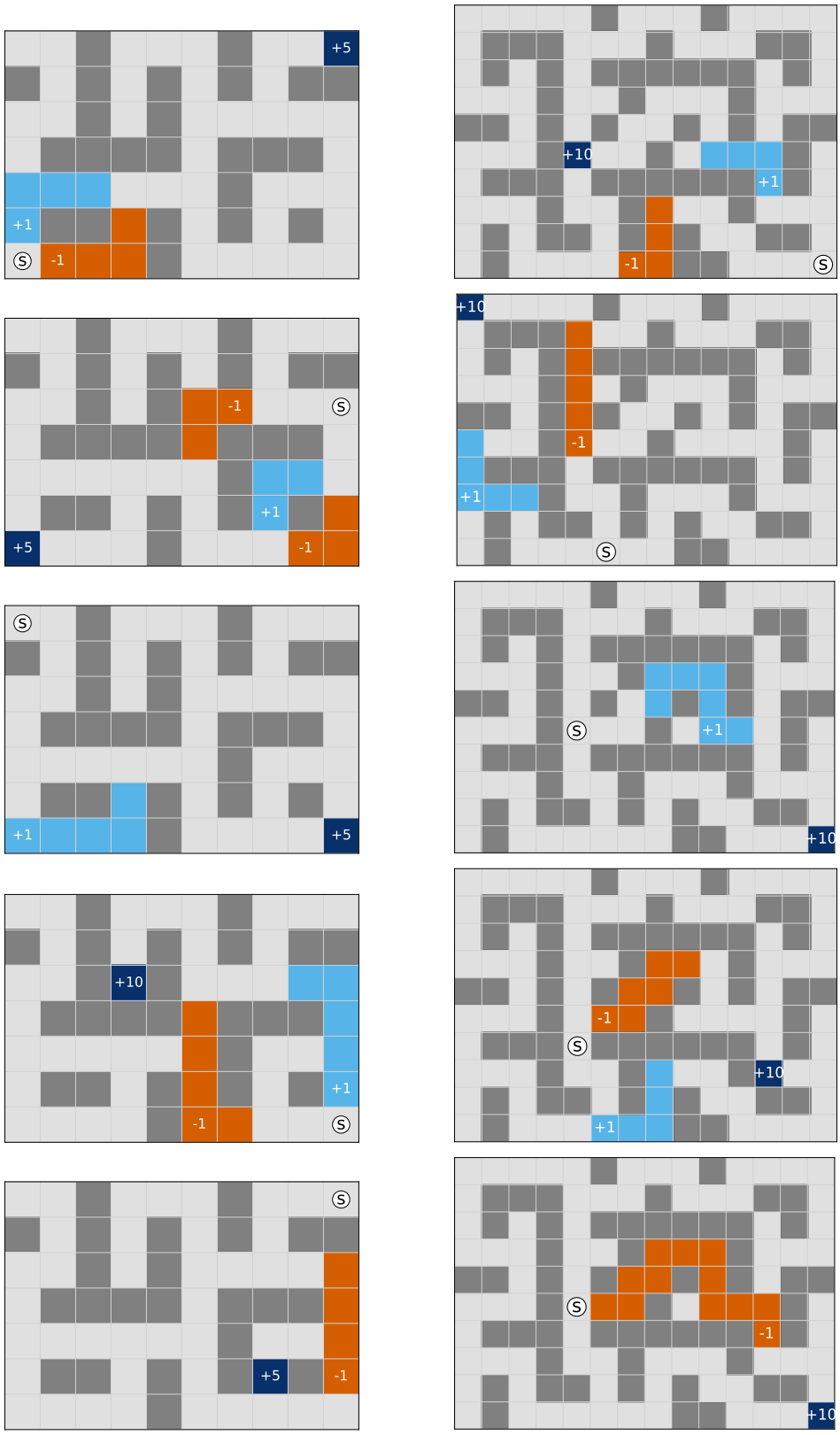


Figure 7: Tasks 1–5 (top to bottom) for Large AntMaze (left) and Giant AntMaze (right). Initial states are indicated by a white circle labeled "s". Rewards take values in $\{0, +1, -1, +5, +10\}$.

C.2 Model architecture

We use fully connected neural networks for all model components, as observations in AntMaze are low-dimensional state vectors. All networks are implemented as multi-layer perceptrons (MLPs) with hidden dimensions specified per method (typically 3–4 layers of width 512) and GELU activations. Unless stated otherwise, layer normalization is not applied.

State and goal encoding. We do not use a separate encoder module; instead, state and goal inputs are concatenated and fed directly into the networks. For goal-conditioned components, goals are either raw states (in GCRL baselines) or latent representations inferred via the backward representation network. Latent vectors are normalized to lie on a sphere of radius $\sqrt{d}$, where d is the latent dimension.

Representation networks. We use forward and backward representation networks, both parameterized as MLPs. The backward representation maps states to latent vectors, while the forward representation maps state–latent (and optionally state–action–latent) tuples to latent predictions. The forward representation uses an ensemble of size two (double-critic setup), while the backward representation uses a single network. In the hierarchical FB variant, an additional marginal forward representation network is introduced and trained to match the action-marginalized version of the forward representation.

Actor networks. Policies are modeled as Gaussian distributions with diagonal covariance. The mean is produced by an MLP, while the standard deviation is state-independent and fixed (constant across dimensions). For low-level policies, actions are optionally passed through a $\tanh$ squashing function to enforce bounds in $[-1, 1]$. The high-level policy outputs latent subgoals and follows the same Gaussian parameterization, without squashing.

Parameter sharing and freezing. In all hierarchical variants, the forward representation, backward representation, and low-level actor are initialized from pretrained models and kept frozen during training. Only the newly introduced components (e.g., high-level actor and marginal forward representation) are updated. This is implemented via parameter masking in the optimizer.

Critic structure. The forward representation serves as a latent-space critic. We use a double critic architecture (ensemble of size two) to improve stability, analogous to double Q-learning. The outputs are vector-valued (dimension equal to the latent size), and scalar values are obtained via inner products with reward latents.

C.3 Training protocol

General training setup. We build on implementations of baseline algorithms from Park et al. (2025a); Zheng et al. (2026). Training is performed fully offline on fixed datasets, without any environment interaction during learning. All methods use pre-collected trajectory datasets provided by OGBench, and we do not use any form of online data collection or replay buffer. Mini-batches are sampled uniformly from the dataset for gradient-based optimization. We use double target networks to alleviate maximization bias. The two value estimates are aggregated in a model-dependent manner, either by taking their mean or a minimum over the ensemble, depending on the specific algorithm.

Terminal state masking. In standard goal-conditioned reinforcement learning (Ghosh et al., 2023; Park et al., 2023), one typically estimates quantities of the form $M_s(g)$, which implicitly correspond to $M_s^{\pi_g}(g)$. In this case, the queried state coincides with the goal ($s' = g$), so defining rewards and terminal masking with respect to g is consistent. However, in a more general setting when learning $M_s^{\pi_g}(s')$, applying masking based on either the goal g or the queried state s' leads to a mismatch: the policy is conditioned on g, while the Bellman recursion and termination are defined with respect to a different state variable. As a result, the induced recursion no longer corresponds to a consistent successor measure under π_g.

Thus, in our setting we do not use terminal-state masking. This design choice is motivated by our objective of supporting general reward functions beyond goal-conditioned settings, where termination is not tied to reaching a goal state. As a result, performance is lower than in standard goal-conditioned benchmarks; for example, OGBench reports HIQL success rates of 96, 91, 65, and 42 percent on AntMaze Medium, Large, Giant, and Teleport, respectively, compared to our HIQL agents, which achieve 93, 73, 5, and 42 percent.

Reward transformations. We do not apply any reward transformations in our experiments such as replacing the sparse indicator reward at the goal with a -1 per-step reward and a 0 reward at the goal. This choice is consistent with our goal embedding framework: in the goal-conditioned setting, the reward embedding of reaching a goal g corresponds directly to $B(g)$, whereas dense reward formulations would require aggregating contributions over all non-goal states, which complicates the representation. Finally, we set the reward temperature to 0, ensuring a consistent treatment of rewards and allowing negative rewards to be properly accounted for by the agents.

Baseline implementations. We use baseline implementations from Park et al. (2025a) and Zheng et al. (2026). All baselines are evaluated using the same offline dataset, training budget and evaluation protocol as our method.

HIQL and its variants: We follow Park et al. (2025a) and keep all hyperparameters unchanged, including `subgoal_steps`=25 and `rep_dim`=10. Value goals are sampled with mixing proportions (current state, future state in trajectory, random state) = (0.2, 0.5, 0.3), while actor goals use (0, 1.0, 0). When training HIQL without high level policy ($-\pi^h$) in the ablation study in Table 6, the actor is trained using the same goal sampling distribution as the value function, i.e., (0.2, 0.5, 0.3). For the bilinear variant, we use an inner-product value function of the form $\phi(s,g)^\top\psi(g)$ with embedding dimension 128. The embedding $\psi(g)$ is normalized to unit norm and, no orthonormality regularization is applied. For the variant with intentions, with probability of 0.5 the goal and intention are set to the same state, and we learn a value function $V(s,g,s')$. Intention goals follow the same sampling scheme as value goals. For direction error, we take the minimum over the two value estimates, while for TD-error we use their average.

ICVF: We follow Zheng et al. (2026) and use their default hyperparameters. Intention goals are sampled using the same mixture as value goals, with distribution (0, 0.625, 0.375) together with geometric sampling for future goals. In actor training, we include an additional behavior cloning (BC) loss with coefficient $\alpha = 0.3$ for Medium and Teleport, and $\alpha = 3.0$ for Large and Giant.

OneStep FB: We use BC coefficient $\alpha = 0.1$ for Teleport and $\alpha = 0.03$ for the remaining environments. We set the reward temperature to 0 instead of 10.0. All other hyperparameters follow Zheng et al. (2026).

FB: We follow Zheng et al. (2026) but introduce several modifications for consistency in our setting. We remove actions from the backward representation, remove the $(1-\gamma)$ term in the diagonal loss to match the original FB formulation, set the reward temperature to 0, use an orthonormality coefficient of 1.0, set BC coefficient $\alpha = 0.03$, and normalize the Q-loss used in actor training.

Marginalization loss of FB $+\pi^h$. As discussed in Section 5.1, the standard FB formulation learns a forward representation of the form $F(s,a,z)$, which explicitly depends on actions. To enable the use of the switching advantage approximation in (10), we introduce a marginalized forward representation $F'(s,z)$ by averaging over actions. This is enforced via the following objective:

$$\mathcal{L}_{\text{marg}}(F') = \mathbb{E}_{s,z}\left[\left(F'(s,z) - \frac{1}{N}\sum_{i=1}^{N} F(s,a_i,z)\right)^2\right], \quad a_i \sim \pi_z(\cdot \mid s)$$

where N=8 denotes the number of actions sampled from the policy. For constructing the latent variables used in this loss, we set `latent_mix_prob`=0.5, corresponding to an equal probability of sampling either embeddings of permuted observations or random vectors drawn uniformly from the surface of a sphere with radius $\sqrt{d}$. The resulting marginalized representation is then used to train the high-level policy according to the objective in (11).

Latent space sampling. For the high-level actor loss in FB π-Switch and FB $+\pi^h$, subgoals (high-actor targets) are obtained via geometric sampling over indices of future states in the trajectory conditioned on the current state, and then mapped into the latent space via backward networks. High-actor goals are sampled uniformly from future states in the trajectory (from the next state until episode termination), mixed with isotropic Gaussian samples that are normalized to lie on a sphere, with mixing ratio `critic_latent_mix_prob`=0.5.

The policy is trained using a clipped exponential advantage weighting scheme, where advantages are passed through an exponential function with inverse temperature `high_alpha`, and clipped at 5.0 for numerical stability. The corresponding hyperparameter values are reported in Tables 4 and 5.

C.4 Hyperparameters

In Table 3, we report the hyperparameters shared across all methods. In Tables 4 and 5, we report all hyperparameters of our hierarchical methods, FB π-Switch and FB $+\pi^h$, to facilitate reproducibility.

Hyperparameter	Value	Description
`num_timesteps`	1M	total training steps
`episode_length`	1,000	maximum steps per episode
`discount`	0.995 (giant), 0.99 (other)	discount factor γ
`batch_size`	1024	minibatch size
`latent_dim`	128 (other), 10 (HIQL)	latent representation size
`gc_negative`	False	choice of goal reward parameterization: $\{0$ at goal, -1 otherwise$\}$ (True) or $\{1$ at goal, 0 otherwise$\}$ (False)

Table 3: General/shared hyperparameters across agents used in our experiments.

C.5 Evaluation protocol

For each agent and environment, we evaluate performance over 50 episodes per task, across 5 random seeds and 5 tasks. At evaluation time, we use a deterministic policy. For goal-reaching evaluation, goals are drawn from a fixed evaluation set shared across all methods.

For each task, we precompute a fixed set of 100,000 transitions, which are relabeled according to the corresponding reward function. The agents then use these relabeled state–reward pairs to infer the latent representations. We set the reward temperature to $T = 0$, since nonzero temperatures would otherwise distort the treatment of negative rewards, as negative rewards are effectively mapped to zero under the temperature-based transformation.

We remove goal noise at environment reset, since latents are computed only once and are not updated upon resets. In contrast, standard GCRL methods (e.g., HIQL) condition directly on the true (noisy) goal at each reset, which can give them an advantage over methods that operate in a fixed latent representation. This design choice ensures a fair comparison by providing access to the same goal information across methods, putting goal- and latent-conditioned approaches on equal footing.

Performance in goal-reaching tasks. As is standard in goal-conditioned reinforcement learning, we evaluate performance using average success rate over a set of test goals. A trajectory is considered successful if the agent reaches the goal before episode termination, where success is defined as the agent entering a 0.5-radius ball in Euclidean distance in the (x, y) plane around the goal location (for AntMaze environments). We compute success as a binary indicator per episode and report the mean success rate across all evaluated goals. Final results are averaged over multiple random seeds, and we report standard deviation across seeds to quantify variability.

Aggregation over tasks with general rewards. We report performance using the interquartile mean (IQM), following recommendations in reinforcement learning evaluation (Agarwal et al., 2021).

Hyperparameter	Value	Description
lr	3×10^{-4}	learning rate
actor_hidden_dims	(512, 512, 512)	actor network hidden dimensions
forward_repr_hidden_dims	(512, 512, 512)	F-representation network hidden dim.
backward_repr_hidden_dims	(512, 512, 512)	B-representation network hidden dim.
actor_layer_norm	False	use layer normalization for actors
forward_repr_layer_norm	True	use layer normalization for F-representations
backward_repr_layer_norm	True	use layer normalization for B-representations
activation	gelu	activation function
tau	0.005	target network update rate
alpha	3.0	AWR temperature for low-level policy
const_std	True	use constant std in actor
reward_temperature	0.0	reward weighting temperature
num_zero_shot_samples	100,000	number of zero-shot samples for latent inference
expectile	0.7	expectile parameter
value_p_curgoal	0.2	prob. of using current state as value goal
value_p_trajgoal	0.5	prob. of using future state in traj. as value goal
value_p_randomgoal	0.3	prob. of using random state as value goal
value_geom_sample	True	use geometric sampling for value goals
low_actor_p_curgoal	0.2	prob. of using current state as low-actor goal
low_actor_p_trajgoal	0.5	prob. of using future state in traj. as low-actor goal
low_actor_p_randomgoal	0.3	prob. of using random state as low-actor goal
high_actor_p_curgoal	0.0	prob. of using current state as high-actor goal
high_actor_p_trajgoal	1.0	prob. of using future state in traj. as high-actor goal
high_actor_p_randomgoal	0.0	prob. of using random state as high-actor goal
actor_geom_sample	False	use geometric sampling for actor goals
normalize_latent	True	normalize backward representations
orthonorm_coeff	1×10^{-4}	orthonormalization coefficient for B-representations
actor_latent_mix_prob	0.5	prob. of sampling unif. random actor latent
critic_latent_mix_prob	0.5	prob. of sampling unif. random critic latent
high_alpha	0.1	AWR temperature for high-level policy
high_adv_threshold	5.0	threshold for adv. clipping before exponentiation

Table 4: Hyperparameters of the FB π-Switch algorithm. Abbreviations: prob. = probability, traj. = trajectory, std. = standard deviation, unif. = uniformly.

Specifically, we first compute normalized returns for each task by linearly scaling (undiscounted) episode returns using the minimum and maximum returns observed across all methods and seeds within that task, ensuring comparability across tasks with different reward scales. This normalization is performed per task before any aggregation. We then compute the interquartile mean over the pooled set of normalized returns from all tasks and seeds. In our setting, we evaluate on 5 tasks, so the IQM is computed over the combined distribution of normalized returns across these tasks.

To quantify uncertainty, we compute 95% confidence intervals using a stratified bootstrap procedure: we resample seeds within each task independently, preserving the task structure, and then recompute the IQM over the resampled data. Repeating this procedure yields a bootstrap distribution of IQM estimates, from which we extract the 2.5th and 97.5th percentiles as the confidence interval bounds. This ensures that uncertainty reflects both variability across seeds and heterogeneity across tasks.

C.6 Compute details

All experiments are run on NVIDIA A40 GPUs, with independent runs executed in parallel across multiple devices. We use JAX-based implementations for all models and optimization. Each model is trained for up to 1M gradient steps, taking at most 2–3 hours per run depending on the method. Evaluation is performed using 5 random seeds per method, with all seeds executed independently in parallel. We use fixed hyperparameters across runs to ensure reproducibility.

Hyperparameter	Value	Description
lr	1×10^{-4}	learning rate
actor_hidden_dims	(512, 512, 512, 512)	actor network hidden dimensions
forward_repr_hidden_dims	(512, 512, 512, 512)	F-representation network hidden dim.
backward_repr_hidden_dims	(512, 512, 512, 512)	B-representation network hidden dim.
actor_layer_norm	False	use layer normalization for actors
forward_repr_layer_norm	False	use layer normalization for F-representations
backward_repr_layer_norm	False	use layer normalization for B-representations
activation	gelu	activation function
tau	0.005	target network update rate
alpha	0.03	behavioral cloning coefficient
const_std	True	use constant std in actor
reward_temperature	0.0	reward weighting temperature
num_zero_shot_samples	100,000	number of zero-shot samples for latent inference
tanh_squash	True	use tanh squashing for actor outputs
normalize_q_loss	True	normalize Q loss
repr_agg	mean	aggregation method for representations
high_actor_p_curgoal	0.0	prob. of using current state as high-actor goal
high_actor_p_trajgoal	1.0	prob. of using future state in traj. as high-actor goal
high_actor_p_randomgoal	0.0	prob. of using random state as high-actor goal
actor_geom_sample	False	use geometric sampling for actor goals
normalize_latent	True	normalize backward representations
orthonorm_coeff	1.0	orthonormalization coefficient for B-representations
margin_latent_mix_prob	0.5	prob. of sampling unif. random latent for margin.
critic_latent_mix_prob	0.5	prob. of sampling unif. random critic latent
high_alpha	1×10^{-3}	AWR temperature for high-level policy
high_adv_threshold	5.0	threshold for adv. clipping before exponentiation
num_actions_sampled	8	number of sampled actions for marginalization

Table 5: Hyperparameters of the FB $+\pi^h$ algorithm. Abbreviations: prob. = probability, traj. = trajectory, std. = standard deviation, unif. = uniformly, margin. = marginalization.

D Further experimental results

D.1 Discrete maze environment

In this section, we provide additional results for the discrete maze experiments introduced in Section 6. We begin by recalling the switching advantage defined in Equation (10). One of its components, corresponding to the contribution before reaching an intermediate subgoal w, is visualized in Figure 4(d) and denoted $A_{\mathrm{FB}}(s, w, r)|_{<w}$. This term can be written as:

$$A_{\mathrm{FB}}(s, w, r)|_{<w} = F(s, z_w)^\top z_r - \frac{F(s, z_w)^\top z_w}{F(w, z_w)^\top z_w} F(w, z_w)^\top z_r$$

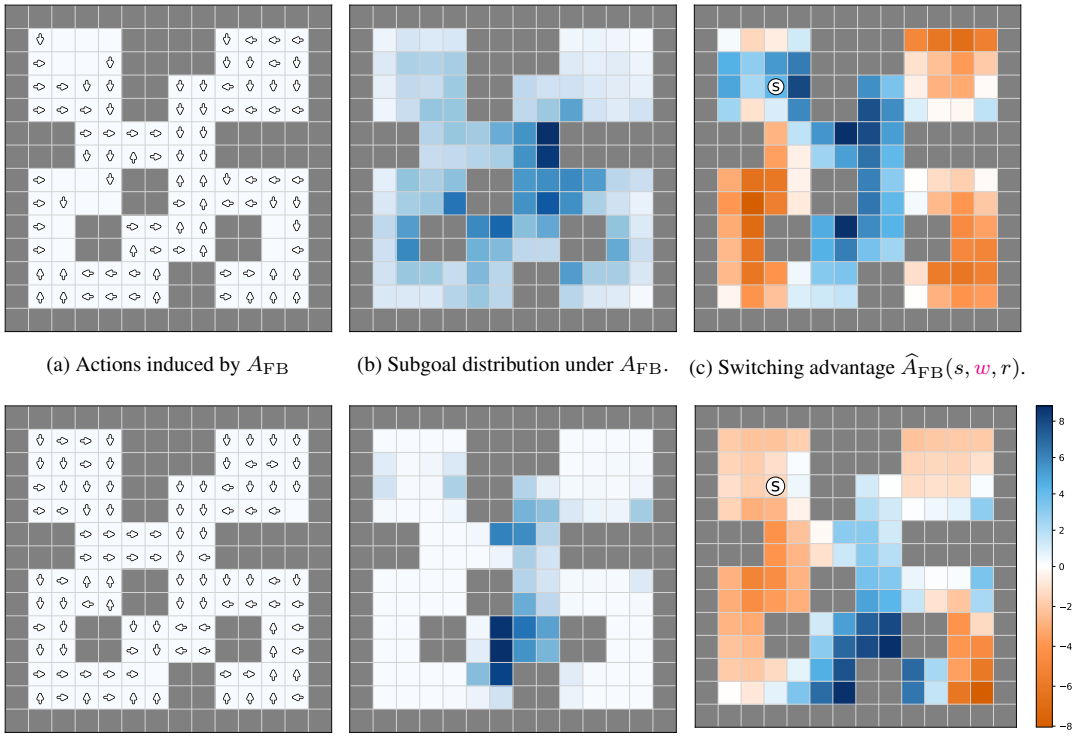
The second component can be interpreted as the expected return obtained by following the policy associated with z_w starting from state w. When the subgoal policy is sufficiently strong, it tends to remain near w, so this term effectively captures the return of staying in the subgoal region. As a result, it becomes highly concentrated on states with nonzero reward, making it difficult to estimate reliably with function approximation.

To mitigate this issue, we introduce a simplified proxy for the switching advantage:

$$\widehat{A}_{\mathrm{FB}}(s, w, z) := F(s, z_w)^\top z + \frac{F(s, z_w)^\top z_w}{F(w, z_w)^\top z_w} F(w, z)^\top z - F(s, z)^\top z$$

Figure 8 compares the original switching advantage A_{FB} with the proxy $\widehat{A}_{\mathrm{FB}}$.

In Figures (a) and (d), actions are extracted by selecting, for each state, the subgoal that maximizes the corresponding advantage and then mapping it to the action that moves the agent closer to that subgoal. We observe that actions extracted from $\widehat{A}_{\mathrm{FB}}$ are more effective.



(a) Actions induced by A_{FB} (b) Subgoal distribution under A_{FB}. (c) Switching advantage $\widehat{A}_{\mathrm{FB}}(s, w, r)$.

(d) Actions induced by $\widehat{A}_{\mathrm{FB}}$ (e) Subgoal distribution under $\widehat{A}_{\mathrm{FB}}$. (f) Difference $(\widehat{A}_{\mathrm{FB}} - A_{\mathrm{FB}})(s, w, r)$.

Figure 8: Effect of simplifying the switching advantage: $\widehat{A}_{\mathrm{FB}}$ (c) improves action selection (a,d) and produces more structured subgoal distributions (b,e) compared to A_{FB}.

The middle column shows the induced subgoal distributions. Under $A_{\mathrm{FB}}(s, w, r)$, subgoals are diffuse and often assigned to irrelevant regions of the state space. In contrast, $\widehat{A}_{\mathrm{FB}}$ concentrates subgoals along paths leading to reward.

Finally, Figure (c) visualizes the proxy $\widehat{A}_{\mathrm{FB}}(s, w, r)$, which can be compared to $A_{\mathrm{FB}}(s, w, r)$ in Figure 4(c), while (f) shows their difference. Overall, the proxy appears to improve the alignment between subgoals and optimal behavior, while potentially simplifying learning.

D.2 Goal-reaching experiments

This section provides additional results for the continuous control for goal-reaching tasks described in Section 6. We first report the full version of Table 1, including results from the HIQL ablation study referenced in the main text.

For clarity, we briefly summarize the variants. FB refers to the standard FB algorithm. FB $+\pi^h$ augments this model with a learned high-level policy, as described in Section 5.1 and Appendix C.3. Vanilla FB π-Switch denotes our proposed method, while FB π-Switch $-\pi^h$ corresponds to its non-hierarchical version. Similarly, HIQL $-\pi^h$ removes the high-level policy from HIQL. Subsequent rows introduce additional modifications in an additive manner: adding a bilinear representation into value function, and then using intention-based value learning, where reward and policy are not necessarily aligned.

The full results are shown in Table 6.

Algorithm	**Variant**	**Medium**	**Large**	**Giant**	**Teleport**
HIQL	vanilla	93 ± 1	73 ± 6	5 ± 3	42 ± 4
	$-\pi^h$	73 ± 3	23 ± 4	0 ± 0	32 ± 6
	+ bilinear	45 ± 6	20 ± 3	0 ± 0	14 ± 6
	+ intentions	66 ± 6	20 ± 4	1 ± 1	11 ± 3
ICVF		46 ± 37	28 ± 3	0 ± 0	23 ± 6
One-Step FB		35 ± 7	23 ± 18	0 ± 0	8 ± 2
FB	vanilla	47 ± 1	21 ± 5	0 ± 0	25 ± 7
	$+\pi^h$	50 ± 12	20 ± 5	0 ± 0	13 ± 7
FB π- Switch	vanilla	87± 6	66 ± 9	1 ± 1	40 ± 6
	$-\pi^h$	70 ± 4	37 ± 7	1 ± 1	29 ± 4

Table 6: Average (binary) success rate (%) across the five test-time goals with `navigate-v0` datasets from Park et al. (2025a). The results are averaged over 50 episodes and 5 seeds, and we report the standard deviations after the $\pm$ sign. Note that operations in column Variant are additive - for example, entries in row +intentions correspond to HIQL algorithm without high-level policy, with bilinear critic and with intentions.

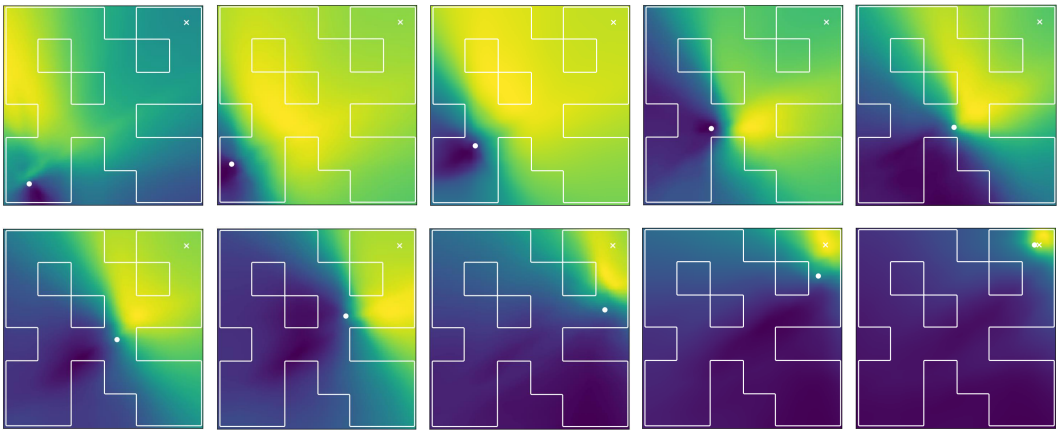
A useful way to interpret these results is to note that several HIQL variants progressively move closer to the non-hierarchical version of FB π-Switch in terms of architectural choices. The remaining differences are primarily due to representation design: HIQL still operates partially in the state space and enforces strict unit normalization of goal representations, whereas FB π-Switch operates fully in latent space and uses the orthonormalization objective from the FB framework. Under these more aligned settings, FB π-Switch tends to achieve stronger performance than the corresponding HIQL-based variants.

The effect of hierarchy is also consistent across methods. Adding a high-level policy leads to clear performance improvements in both HIQL and FB π-Switch. That said, some differences remain. For example, the non-hierarchical HIQL variant degrades more noticeably on the Large maze, while HIQL slightly outperforms FB π-Switch on average in the same environment. One possible expla-

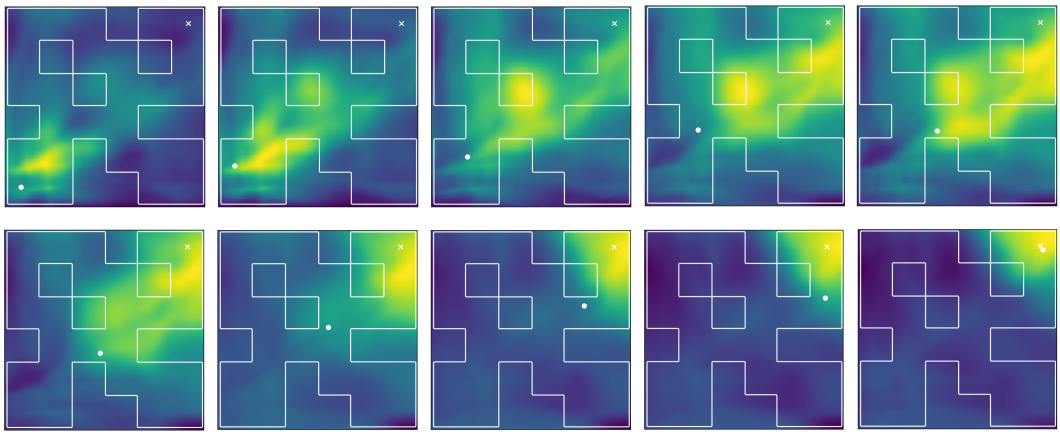
nation is that success rates above roughly 70% on the Large AntMaze tasks are difficult to exceed, creating a performance ceiling that affects both approaches.

Figure 9 shows how high-level subgoal selection evolves during an episode for HIQL, FB π-Switch, and FB $+\pi^h$, highlighting differences in the learned high-level policy mentioned around Figure 5 in Section 6. For a fixed goal state g, we compute $z_g = B(g)$, and for all discretized states x in the maze we compute $z_x = B(x)$. At each time step, we sample $z \sim \pi^h(\cdot|s, z_g)$, and visualize $\langle z_x, z\rangle$ at the corresponding spatial location x.

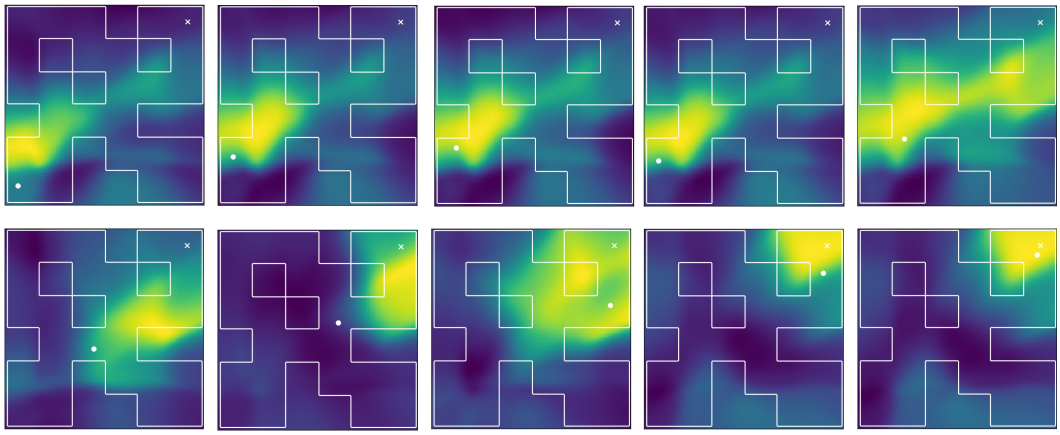
Figures 10 and 11 further visualize the learned successor measures for all models reported in Table 6.



(a) HIQL

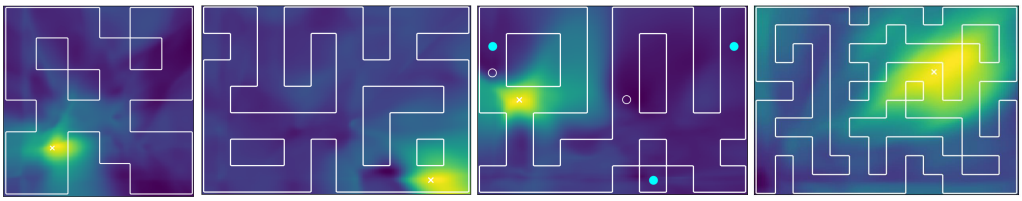


(b) FB π-Switch

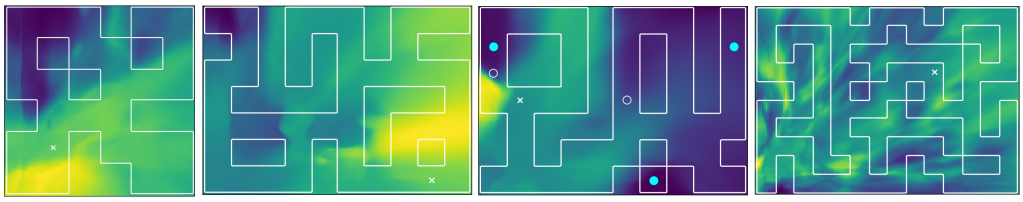


(c) FB $+\pi^h$

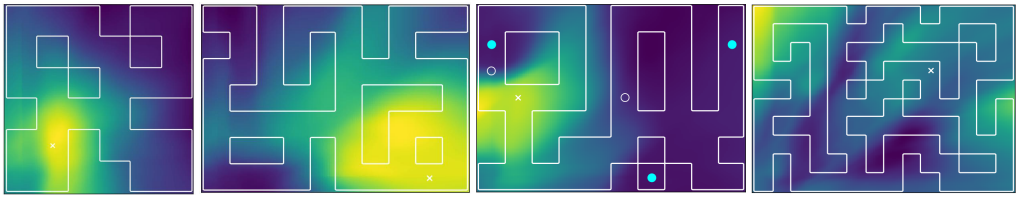
Figure 9: Evolution of high-level policies over a single trajectory for different methods in Medium-Antmaze environment. Within each trajectory, visualizations are temporally equally spaced. The dot indicates the agent's current state, while the cross denotes the goal.



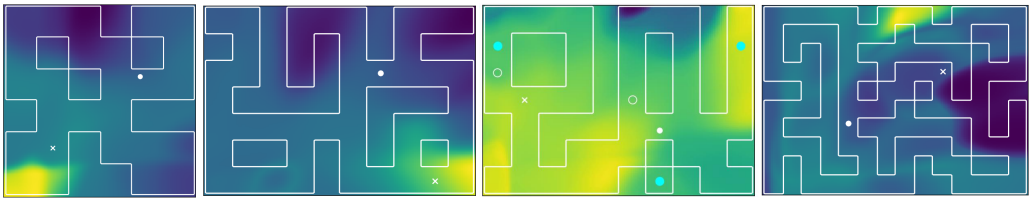
(a) ICVF: state-value successor measures $M_s^{\pi_g}(g)$



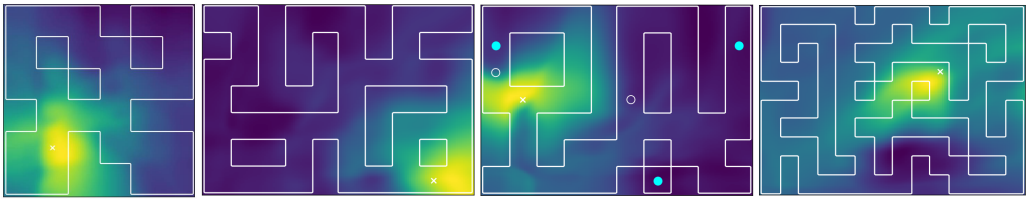
(b) One-Step FB: state-value successor measures $M_s^{\pi_g}(g)$



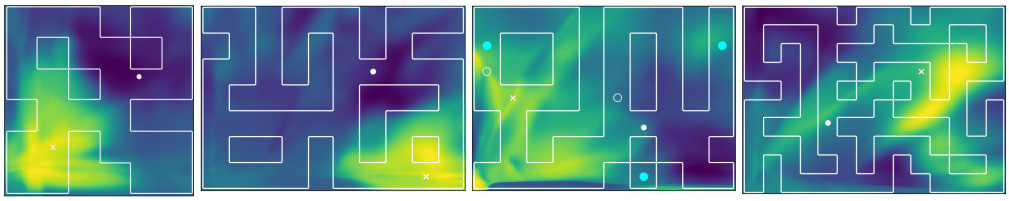
(c) FB: state-value successor measures $M_s^{\pi_g}(g)$



(d) FB: intention-conditioned state-value successor measures $M_s^{\pi_w}(g)$

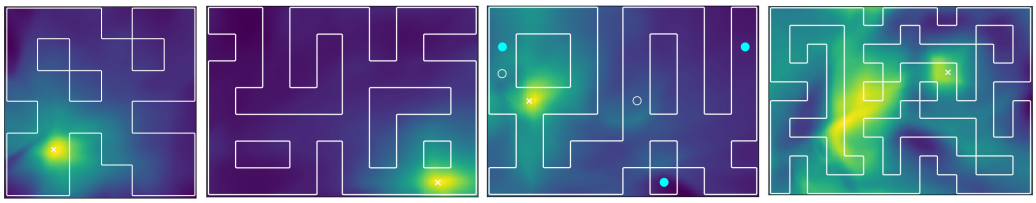


(e) FB π-Switch: state-value successor measures $M_s^{\pi_g}(g)$

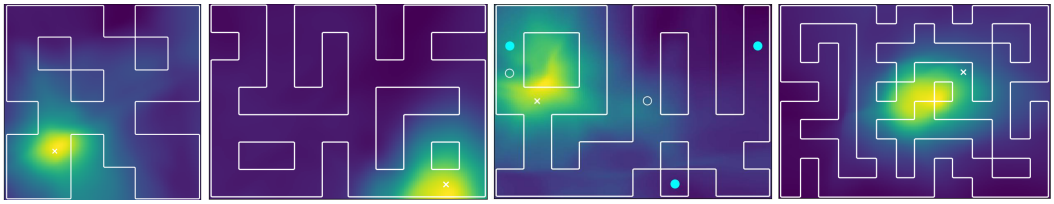


(f) FB π-Switch: intention-conditioned state-value successor measures $M_s^{\pi_w}(g)$

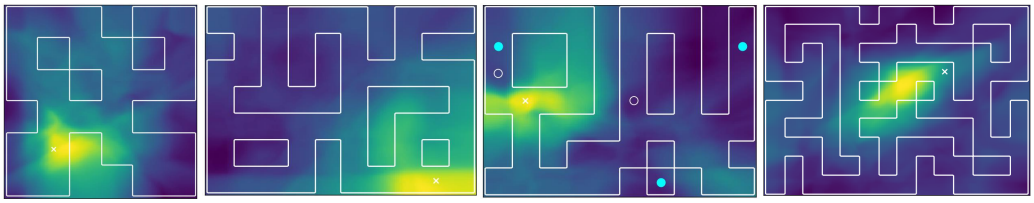
Figure 10: We report results for a single trained model per method (one seed), without aggregation over multiple seeds. All quantities are plotted as functions of the magenta variables. Fixed states s are indicated by dots, while goals g are indicated by crosses. Columns correspond to the Medium, Large, Teleport, and Giant AntMaze environments.



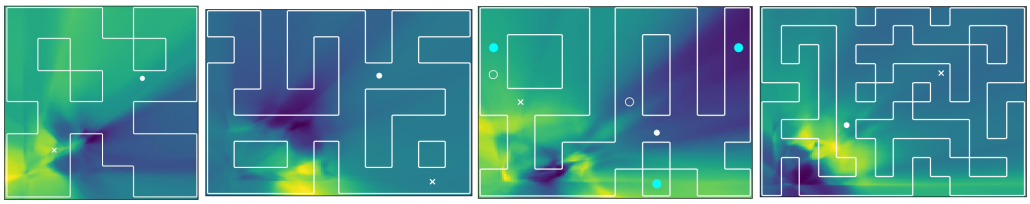
(a) HIQL: state-value successor measures $M_s^{\pi_g}(g)$



(b) HIQL + bilinear: state-value successor measures $M_s^{\pi_g}(g)$



(c) HIQL + bilinear + intentions: state-value successor measures $M_s^{\pi_g}(g)$



(d) HIQL + bilinear + intentions: intention-conditioned state-value successor measures $M_s^{\pi_w}(g)$

Figure 11: We report results for a single trained model per method (one seed), without aggregation over multiple seeds. All quantities are plotted as functions of the magenta variables. Fixed states s are indicated by dots, while goals g are indicated by crosses. Columns correspond to the Medium, Large, Teleport, and Giant AntMaze environments.

D.3 General rewards experiments.

This section provides additional results for the continuous control tasks with general reward functions described in Section 6. We first report the full results underlying Figure 6. For each of the five tasks (shown in Figure 7), we report the average undiscounted episodic return over episodes of length 1000.

It is important to emphasize that the tasks differ substantially in their reward structure. For example, task 3 in both environments contains no negative rewards, whereas tasks 5 include large negative regions combined sparse positive rewards. As a result, raw returns are not directly comparable across tasks. Instead, we normalize performance per task and aggregate using the interquartile mean (IQM), as described in Appendix C.5.

The results are reported in Table 7.

Env.	**Task**	**ICVF**	**One-Step FB**	**FB**		**FB π-Switch**	
				vanilla	$+\pi^h$	vanilla	$-\pi^h$
antmaze large	task 1	282 ± 43	207 ± 56	938 ± 308	523 ± 95	756 ± 185	451 ± 99
	task 2	360 ± 179	-171 ± 34	4 ± 281	-45 ± 75	456 ± 305	131 ± 107
	task 3	1183 ± 262	192 ± 232	115 ± 107	58 ± 75	697 ± 354	209 ± 133
	task 4	518 ± 206	184 ± 352	906 ± 32	916 ± 41	664 ± 97	602 ± 223
	task 5	109 ± 44	-79 ± 96	36 ± 67	-5 ± 11	479 ± 89	337± 238
	IQM score	0.48	0.09	0.37	0.24	0.67	0.35
	LCI (95%)	0.42	0.03	0.28	0.21	0.56	0.30
	UCI (95%)	0.57	0.15	0.50	0.27	0.74	0.43
antmaze giant	task 1	-19 ± 28	36 ± 30	1 ± 55	2 ± 12	101 ± 87	125 ± 167
	task 2	266 ± 50	30 ± 30	158 ± 128	169 ± 117	292 ± 92	326 ± 60
	task 3	185 ± 76	173 ± 107	292 ± 175	292 ± 170	470 ± 268	437 ± 187
	task 4	192 ± 65	-176 ± 152	138 ± 158	-3 ± 134	321 ± 61	362 ± 99
	task 5	-40 ± 80	-292 ± 138	-391 ± 283	-369 ± 210	257 ± 305	42 ± 150
	IQM score	0.43	0.21	0.33	0.30	0.65	0.63
	LCI (95%)	0.39	0.17	0.23	0.24	0.54	0.54
	UCI (95%)	0.47	0.28	0.43	0.37	0.74	0.73

Table 7: Averaged undiscounted returns across 50 evaluations and five test-time rewards from Figure 7 averaged over 5 seeds. We report standard deviations after the $\pm$ sign. We also report interquartile mean scores (IQM) and lower and upper 95% confidence intervals - see Appendix C.5 for more details on the evaluation protocol.

Overall, the results suggest that while several methods achieve strong performance on individual tasks, FB π-Switch is more consistent across tasks. In Antmaze-Large, the lower 95% confidence interval of FB π-Switch is close to or exceeds the upper confidence interval of the best performing agent. A similar trend is observed in Antmaze-Giant, where even the non-hierarchical variant of FB π-Switch remains competitive.

To better understand these differences, we further decompose returns into two components: (i) return accumulated before reaching the highest-rewarding state for the first time, and (ii) return accumulated after reaching it. These results are shown in Figure 12. This decomposition allows us to assess whether agents follow reward-aligned paths and how efficiently they reach the highest-rewarding state.

For Antmaze-Large, FB π-Switch improves over its non-hierarchical variant primarily by reaching the highest-reward state faster, while maintaining similar reward along the way. An interesting observation in Antmaze-Giant is that the non-hierarchical FB π-Switch achieves higher average performance on task 1 by reaching the final state more quickly, but does so at the cost of traversing regions with negative reward more frequently. This highlights the importance of reward design when evaluating behavioral quality beyond aggregate return.

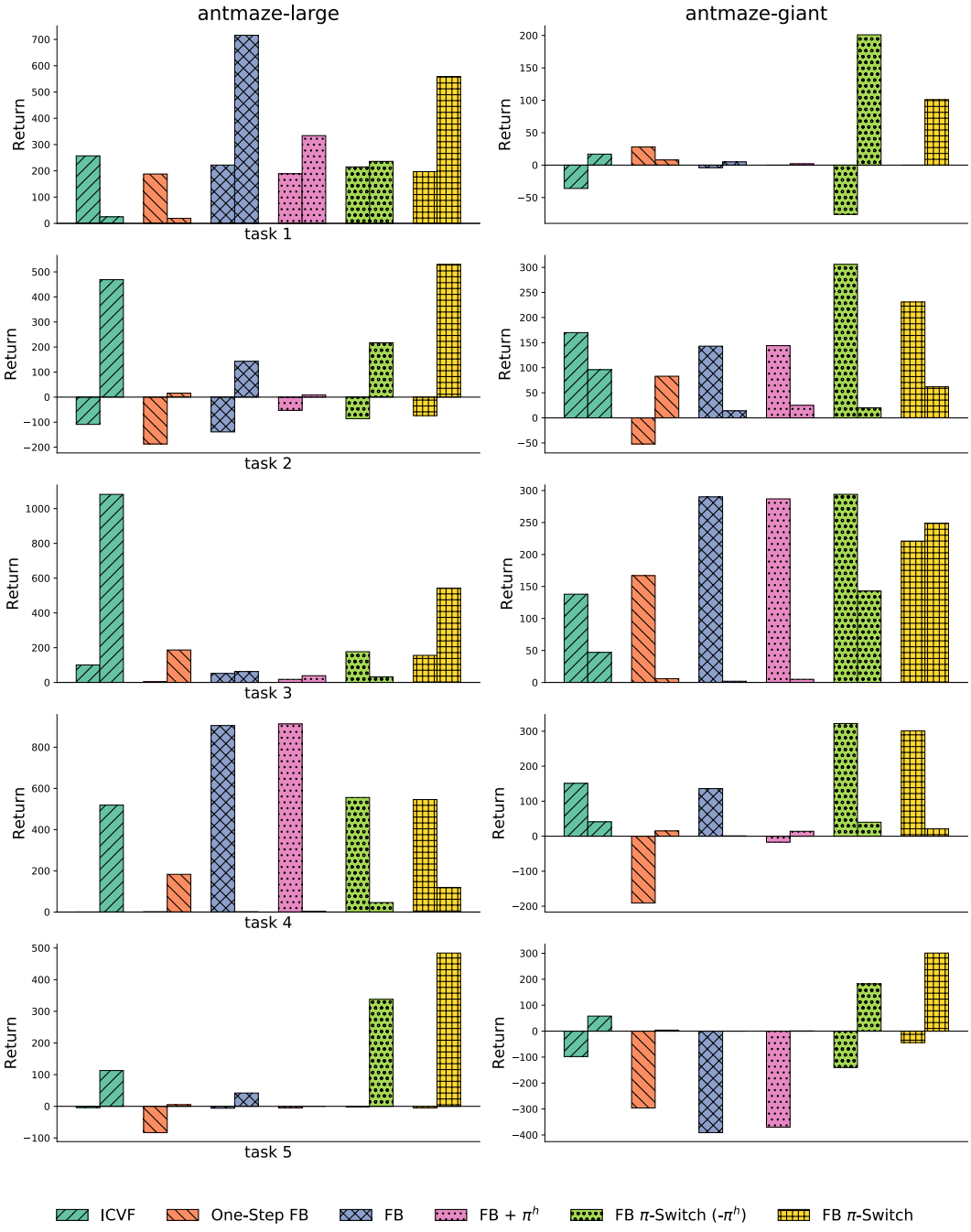


Figure 12: Average episodic return before (left bars) and after (right bars) reaching the highest-reward state for the first time. Results are shown separately for AntMaze-Large (left) and AntMaze-Giant (right).