# LEARNING MULTIPLE TIMESCALES FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Existing approaches to offline goal-conditioned reinforcement learning (GCRL) struggle with long-horizon tasks. Discounting shrinks value differences between distant states until they fall below the function approximation error, leaving the agent with no signal for ranking states. Temporal abstraction—treating $k$ environment steps as a single transition—restores this signal at long range, but no single fixed $k$ suits all state-goal distances: large $k$ preserves value differences across long temporal distances while collapsing distinctions between nearby states, and small $k$ does the reverse. We make this trade-off explicit and introduce Generalized Implicit Temporal Abstraction (GITA), which conditions a single value function on $k$. GITA trains one policy by aggregating advantage-weighted supervision across multiple $k$ values, so scales assigning larger positive advantages to a state-goal pair contribute more strongly to its update. GITA does not need to choose between local resolution and long-range signal; it retains both without committing to a single $k$. On OGBench, GITA outperforms a broad range of offline GCRL baselines, raising average success rate across all tasks by 25 percentage points (73% relative improvement) over HIQL. It also improves over the strongest fixed-$k$ method, OTA, by 7 percentage points (14% relative).

## 1 INTRODUCTION

Offline goal-conditioned reinforcement learning (GCRL) aims to learn policies that reach arbitrary goals using only previously collected data (Levine et al., 2020). They typically do so by learning a goal-conditioned value function (Schaul et al., 2015). In long-horizon tasks, discounting shrinks the value difference between neighboring states as the goal moves farther away, and once that difference falls below the function approximator error, the learned values can no longer rank actions correctly (Opryshko et al., 2026).

Temporal abstraction (Sutton et al., 1999) mitigates this issue by treating multiple primitive steps as a single transition. Even the simplest instantiation, fixing a stride of $k$ steps, shortens the value-propagation chain for a given state-goal distance by a factor of $k$. However, a large $k$ that restores signal for distant goals also collapses distinctions between nearby states, as nearby states whose remaining distances fall within the same abstract-horizon bin become indistinguishable under the abstract transition. The right $k$, therefore, depends on goal distance, and no fixed choice serves every state-goal pair.

In this paper, we introduce Generalized Implicit Temporal Abstraction (GITA), a GCRL method that learns a single $k$-conditioned value function spanning multiple temporal abstraction scales. From this value function, GITA derives an advantage at every scale and trains one policy on their aggregate via Advantage-Weighted Regression (AWR; Peng et al., 2019). Because AWR weights each sample by its exponentiated advantage, scales assigning larger positive advantages at a given state-goal pair contribute more strongly to the update. GITA therefore combines multiple effective horizons *per state-goal pair* with no explicit selection mechanism—the sense in which the abstraction is implicit. It lifts the constraint that every fixed-scale method inherits by having nearby pairs resolved at small $k$, distant ones at large $k$, within one value function.

Empirically, we characterize how $k$ governs value discrimination and demonstrate GITA's effectiveness on OGBench (Park et al., 2025). In control environments where we can cleanly vary the distance between states, we show that distinguishing the values of two states at distance $d$ requires



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

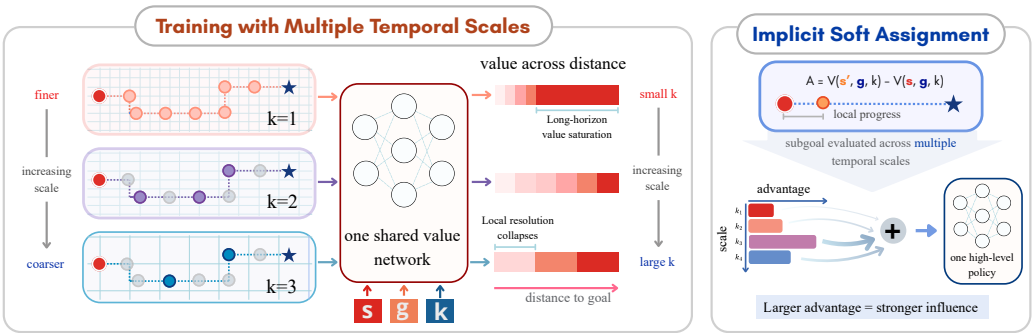


Figure 1: **Overview of the proposed multi-scale temporal abstraction framework.** *Left*: a shared value function conditioned on $k$ learns multiple abstraction scales. *Right*: scale-specific advantages set the relative contribution of each $k$ value, inducing an implicit soft assignment when training the shared high-level policy.

$k$ to grow with $d$: below a threshold, the values of all sufficiently distant states collapse together. Conversely, a large enough $k$ saturates the values of states closer than a threshold, collapsing those instead. No single $k$ escapes both failures, which is precisely the trade-off GITA is designed to resolve. On OGBench, GITA outperforms existing hierarchical baselines, including HIQL (Park et al., 2023) and OTA (Ahn et al., 2025), as well as a broad range of other GCRL methods across multiple morphologies, datasets, and environment types. In aggregate, GITA improves over HIQL by 25 percentage points (p.p.) of success rate (73% relative) and over OTA by 7 p.p. (14% relative).

## 2 Preliminaries

We consider the offline GCRL setting (Kaelbling, 1993; Levine et al., 2020) defined over a Markov Decision Process (MDP), $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \rho_0, r, \gamma, \mathcal{G} \rangle$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P(s' \mid s, a)$ is the transition dynamics, $\rho_0$ is the initial-state distribution, and $\gamma \in [0, 1)$ is the discount factor. The goal space $\mathcal{G} \subseteq \mathcal{S}$ allows any valid state to serve as a goal (Schaul et al., 2015; Andrychowicz et al., 2017), and $r(s, g) = -\mathbb{1}[s \neq g]$ is a goal-conditioned reward function. We are given an offline dataset $\mathcal{D}$ of unlabeled trajectories $\tau = (s_0, a_0, s_1, \ldots, s_T)$ collected by an unknown behavior policy, and no further environment interaction is permitted. Writing $p^\pi(\tau \mid g) = \rho_0(s_0) \prod_t \pi(a_t \mid s_t, g) P(s_{t+1} \mid s_t, a_t)$ for the trajectory distribution induced by $\pi$, we want to learn a goal-conditioned policy that maximizes $J(\pi) = \mathbb{E}_{g \sim p_\mathcal{G},\, \tau \sim p^\pi(\cdot \mid g)} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, g) \right]$, where $p_\mathcal{G}$ and $H$ are the evaluation goal distribution and horizon.

Hierarchical approaches, the main class of methods we study, facilitate long-horizon goal-conditioned control by introducing intermediate subgoals between the current state and the final goal. HIQL (Park et al., 2023) uses a high-level policy $\pi^h(s_{t+m} \mid s_t, g)$ to predict a subgoal $s_{t+m}$ at offset $m$, and a low-level policy $\pi^\ell(a_t \mid s_t, s_{t+m})$ to select primitive actions toward it. Both policies rely on a goal-conditioned value function trained with the expectile temporal-difference objective:

$$\mathcal{L}_V = \mathbb{E}_{(s_t, s_{t+1}) \sim \mathcal{D},\, g \sim p_\mathcal{G}} \left[ L_2^\nu \left( r(s_t, g) + \gamma \bar{V}(s_{t+1}, g) - V(s_t, g) \right) \right], \tag{1}$$

where $s_{t+1}$ denotes the immediate successor state of $s_t$ in the dataset, $V$ is the goal-conditioned state-value function, $\bar{V}$ is a target network, and $L_2^\nu(u) = |\nu - \mathbb{1}(u < 0)|\, u^2$, with $\nu > 0.5$, denotes the expectile loss.

HIQL extracts both policies with AWR (Peng et al., 2019). The high- and low-level advantages are

$$\begin{aligned} A^h(s_t, s_{t+m}, g) &= V(s_{t+m}, g) - V(s_t, g), \\ A^\ell(s_t, s_{t+1}, s_{t+m}) &= V(s_{t+1}, s_{t+m}) - V(s_t, s_{t+m}), \end{aligned} \tag{2}$$

giving the policy objectives

$$\begin{aligned} J(\pi^h) &= \mathbb{E}_{\mathcal{D},\, g \sim p_\mathcal{G}} \left[ \exp\left(\beta_h A^h\right) \log \pi^h(s_{t+m} \mid s_t, g) \right], \\ J(\pi^\ell) &= \mathbb{E}_{\mathcal{D}} \left[ \exp\left(\beta_\ell A^\ell\right) \log \pi^\ell(a_t \mid s_t, s_{t+m}) \right], \end{aligned} \tag{3}$$

where $\beta_h, \beta_\ell > 0$ are inverse temperatures, and $A^\ell$ denotes the low-level advantage.

The high-level policy is trained from the advantage $A^h = V(s_{t+m}, g) - V(s_t, g)$, which should be positive when the candidate subgoal $s_{t+m}$ represents progress toward $g$. Recovering this ordering requires the value function to resolve a difference between two states that are only $m$ steps apart. As $g$ moves farther away, one-step TD learning must propagate information over correspondingly longer horizons, and repeated discounting shrinks that difference until it is dominated by estimation error (Ahn et al., 2025). The advantage then carries no reliable signal about which subgoals make progress, and high-level policy learning degrades.

Option-aware temporal abstraction (OTA; Ahn et al., 2025) mitigates this issue by replacing the one-step value updates in Eq. (1) with temporally-extended transitions. For a fixed abstraction factor $k$, OTA looks ahead $k$ primitive steps along the trajectory in $\mathcal{D}$ containing $s_t$ and uses $s_{t+k}$ as the successor state. Rather than forming a $k$-step TD target with accumulated rewards and discount $\gamma^k$, OTA treats this temporally-extended transition as a single update with TD error $\delta_k = r(s_{t+k}, g) + \gamma \bar{V}_k^h(s_{t+k}, g) - V_k^h(s_t, g)$, where $V_k^h$ denotes the temporally-extended value function. Each update therefore propagates value information across $k$ primitive steps while applying a single discount $\gamma$, reducing the number of backups needed to connect $s_t$ to $g$ from $d(s_t, g)$ to roughly $\lceil d(s_t, g)/k \rceil$, where $d(s_t, g)$ denotes the number of primitive steps from $s_t$ to $g$. Although a similar contraction can be induced by rescaling the discount as $\widetilde{\gamma} = \gamma^{1/k}$, Ahn et al. (2025) show that this alone does not reproduce the gains of temporally-extended value updates.

However, OTA fixes a single $k$ for all state-goal pairs while learning $V_k^h$. This makes $k$ a problem-dependent hyperparameter that must be tuned per environment, as Ahn et al. (2025) do. Moreover, since the number of backups separating $s_t$ from $g$ scales as $\lceil d(s_t, g)/k \rceil$, the $k$ that keeps this quantity small for distant goals is far larger than necessary for nearby ones, and states within $k$ steps become indistinguishable under $V_k^h$.

We have focused here on the methods GITA builds on most directly. Appendix A situates GITA within the broader offline GCRL and temporal abstraction literature.

## 3 No Fixed Abstraction Suits All State-Goal Distances

We characterize how the temporal abstraction factor affects the high-level advantage used for subgoal selection. Let $d = d^\star(s, g)$ denote the minimum number of primitive steps needed to go from a state $s$ to a goal $g$, and let $s'$ be a candidate subgoal $\Delta$ steps closer to the goal, such that $d^\star(s', g) = d^\star(s, g) - \Delta$. For an abstraction factor $k > 0$, $k$ primitive transitions are represented by one abstract transition, which incurs reward $-1$ and a single discount $\gamma$ under our analytical convention. Relaxing the abstract horizon $\lceil d^\star(s, g)/k \rceil$ to $d^\star(s, g)/k$ so that it varies smoothly with $k$, as a continuous surrogate for the discrete-horizon problem, yields the temporally-abstract value function

$$\widetilde{V}_k(s, g) = -\frac{1 - \gamma^{d^\star(s,g)/k}}{1 - \gamma}. \tag{4}$$

Throughout, we assume $d > \Delta > 0$, $k > 0$, and $0 < \gamma < 1$. Specifically, Proposition 1 formalizes the relationship between the distance between different states and the difference between their induced value functions across different temporal scales $k$. Proofs can be found in Appendix B.

**Proposition 1.** *Let $s$ and $s'$ denote two states such that $s'$ is $\Delta$ primitive steps closer to the goal than $s$. When computed with a fixed stride of $k$ steps, the advantage of a subgoal $s'$ in relation to $s$ is*

$$\widetilde{A}_k(s, s', g) = \widetilde{V}_k(s', g) - \widetilde{V}_k(s, g) = \underbrace{\gamma^{(d-\Delta)/k}}_{\textit{reachability}} \cdot \underbrace{\frac{1 - \gamma^{\Delta/k}}{1 - \gamma}}_{\textit{resolution}}. \tag{5}$$

This proposition formalizes the intuition we have provided above. A coarser abstraction allows the advantage to better discriminate distant states when considering distant goals, because the *reachability factor* increases toward 1 as $k$ grows. However, the same coarser abstraction blurs the distinction between nearby states, since the *resolution factor* decreases toward 0 as $k$ grows. We formalize these results in the corollary below.

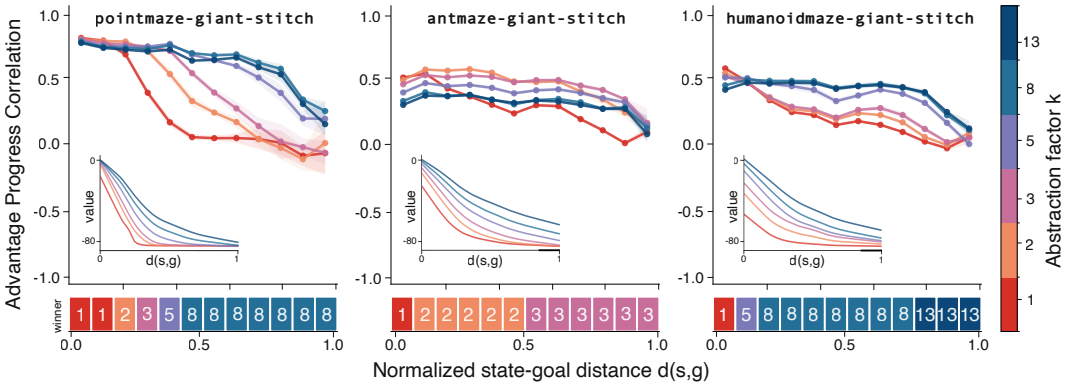


Figure 2: **No single abstraction factor is best at all distances.** APC for different abstraction factors $k$ as the goal becomes more distant (mean $\pm$ std over four seeds). The strip beneath each panel indicates the factor with the highest correlation in each distance regime; the best factor consistently grows with distance, but its range differs across environments. Inset shows the learned value $V_k^h(s, g)$ as a function of geodesic distance.

**Corollary 1.1.** *When computed with a fixed stride of $k$ steps, the advantage of a subgoal $s'$ in relation to $s$ vanishes either when the value of $k$ is too small or too large. Formally:*

$$\lim_{k\to 0^+} \widetilde{A}_k(s, s', g) = 0, \qquad \lim_{k\to\infty} \widetilde{A}_k(s, s', g) = 0. \tag{6}$$

Because $d^\star(s, g)$, the minimum number of primitive steps needed to go from a state $s$ to a goal $g$ varies depending on $s$, Proposition 1 also allows us to conclude that no single $k$ can be equally effective in distinguishing between different pairs of states. We formalize this below.

**Corollary 1.2.** *Let $d = d^\star(s, g)$, and $\Delta = d^\star(s, g) - d^\star(s', g)$. For a state $s'$ where $\Delta \ll d$, the optimal abstraction factor in the continuous relaxed model, $k^\star_{\text{opt}}(d, \Delta)$, grows with the state-goal distance:*

$$k^\star_{\text{opt}}(d, \Delta) \approx -d \log \gamma. \tag{7}$$

*This choice keeps the remaining state-goal distance approximately within one effective discount horizon:*

$$\frac{d - \Delta}{k^\star_{\text{opt}}} \approx -\frac{1}{\log \gamma} \approx \frac{1}{1 - \gamma}, \qquad \gamma \to 1. \tag{8}$$

These predictions hold empirically. On a subset of OGBench environments, we train independent value functions $V_k(s, g)$, one per $k$, and evaluate them on the $(s, s', g)$ tuples used to train the high-level actor. Since subgoal selection depends only on the ordering induced by $A_k(s, s', g) = V_k(s', g) - V_k(s, g)$, we measure the Spearman correlation between $A_k$ and $d(s, g) - d(s', g)$, which we call the Advantage-Progress Correlation (APC). We use geodesic shortest paths (Dijkstra, 1959) as a proxy for $d^\star$. See Appendix C for more details.

Figure 2 shows a clear distance-dependent ordering across abstraction factors. Small factors achieve high APC at short state-goal distances but degrade rapidly as distance grows, consistent with the collapse of the reachability factor: in `pointmaze`, $k = 1$ falls to APC $\approx 0$ by $d \approx 0.4$, while $k = 8$ still retains APC $\approx 0.5$ at twice that distance. Larger factors bring distant pairs back into a value-sensitive range but attain lower APC at short distances, consistent with the loss of the resolution factor in Proposition 1. No factor dominates across the distance range; instead, the best-performing factor increases monotonically with state-goal distance in every environment, qualitatively consistent with the growth of $k^\star_{\text{opt}}$ with $d$ predicted by Corollary 1.2. That range itself differs across environments—the best factor peaks at $k = 3$ in `antmaze` but $k = 13$ in `humanoidmaze`—so a single fixed choice cannot serve all environments.

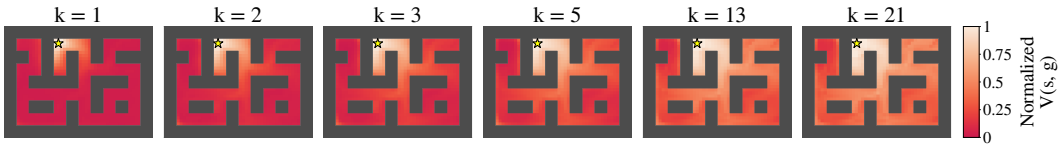


Figure 3: **Varying $k$ directly controls the temporal abstraction scale.** Normalized value function $V(s, g; k)$ on `pointmaze-large-navigate`, evaluated at every navigable cell for $k \in \{1, 2, 3, 5, 13, 21\}$. The star marks the fixed reference state. As $k$ increases, the value-sensitive region expands over longer distances while local resolution progressively degrades.

## 4 Generalized Implicit Temporal Abstraction

Section 3 established that, for locally separated states, the best abstraction factor grows with state-goal distance, so any fixed $k$ is suboptimal for most pairs. **G**eneralized **I**mplicit **T**emporal **A**bstraction (GITA) removes the need to commit to any one $k$. One value function conditioned on $k$ represents all scales at once, and a single high-level policy is trained from the advantages this value function induces at every scale, weighted so that the scales best suited to each state-goal pair dominate its update. An overview of this mechanism is shown in Figure 1. The low-level policy is unchanged from HIQL, predicting primitive actions toward the subgoal the high-level policy proposes.

### 4.1 One Network, Multiple Abstraction Scales

GITA introduces the abstraction factor $k$ as an additional input to the high-level value network, learning a single parameter-shared $V^h_\theta(s, g; k)$ where $k$ is the number of primitive steps represented by each abstract transition. This makes the abstraction scale a controllable input rather than a fixed architectural choice: $k$ is sampled during training, and $V^h_\theta$ can be queried at any scale afterwards. From here we write $k$ as an explicit argument rather than a subscript to emphasize the conditioning.

Given a trajectory $\tau = (s_0, a_0, s_1, \ldots, s_T)$, let $s_t$ denote the current state, $g \sim p_{\mathcal{D}}$ a sampled goal relabeled either from a future state of $\tau$ or from a random state in $\mathcal{D}$, and $s_{t+k}$ the state reached $k$ primitive steps after $s_t$ along $\tau$, or the first state at which the goal is reached or the trajectory terminates, whichever comes first. For each training example, GITA independently samples an abstraction factor from a geometric distribution,

$$k \sim \mathrm{Geom}(1-\alpha), \qquad p(k) = (1-\alpha)\alpha^{\,k-1}, \quad k \geq 1, \tag{9}$$

where $\alpha \in (0, 1)$ sets the mean scale $\mathbb{E}[k] = 1/(1-\alpha)$. Geometric sampling favors shorter scales while retaining a long tail over larger ones, covering both fine and coarse temporal resolutions.

GITA trains the abstraction-conditioned value function with the temporal-difference objective

$$\mathcal{L}_{V^h_\theta} = \mathbb{E}_{s_t \sim \mathcal{D},\, g \sim p_{\mathcal{D}},\, k \sim p(k)} \left[ L^\nu_2 \left( r(s_{t+k}, g) + \gamma \bar{V}^h_{\bar{\theta}}(s_{t+k}, g; k) - V^h_\theta(s_t, g; k) \right) \right], \tag{10}$$

where $\bar{V}^h_{\bar{\theta}}$ is a target network and the successor is evaluated at the same $k$ that generated it, so each scale is trained against its own bootstrapped target. Since all scales are represented by a shared network, they are not learned independently, allowing information to generalize across scales.

### 4.2 High-Level Policy Training with Multi-Scale Advantages

The high-level policy is trained by evaluating each training tuple under several temporal scales at once. We fix a finite set $\mathcal{K}$ of abstraction factors as a hyperparameter and evaluate every tuple $(s_t, s_{t+m}, g)$ under all $k \in \mathcal{K}$, where $m$ is the subgoal offset in primitive steps and is held fixed independently of $k$. This is done so the candidate subgoal is the same, only the scale at which its advantage is measured varies. The policy itself is *not* conditioned on the abstraction factor, giving a single $\pi^h_\phi(s_{t+m} \mid s_t, g)$ shared across all scales.

For every tuple and every $k \in \mathcal{K}$, we compute the abstraction-conditioned high-level advantage

$$A^h(s_t, s_{t+m}, g; k) = V^h_\theta(s_{t+m}, g; k) - V^h_\theta(s_t, g; k). \tag{11}$$

Each abstraction scale therefore contributes a distinct evaluation of the same observed subgoal. We then aggregate the corresponding Advantage-Weighted Regression (AWR) weights across scales,

$$W(s_t, s_{t+m}, g) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \exp\big(\beta_h A^h(s_t, s_{t+m}, g; k)\big)\,, \tag{12}$$

and train the high-level policy according to

$$\mathcal{L}_{\pi^h_\phi} = -\mathbb{E}_{s_t, s_{t+m} \sim \mathcal{D}, g \sim p_{\mathcal{D}}} \left[ W(s_t, s_{t+m}, g) \log \pi^h_\phi(s_{t+m} \mid s_t, g) \right]. \tag{13}$$

Applying the exponential before averaging makes $W$ a soft aggregation of scale-specific AWR weights. Each term $\exp(\beta_h A^h)$ is a weight in its own right, so scales distorted by saturation or excessive abstraction contribute weights near $\exp(0)$, while the scale with the clearest positive advantage drives the update. By Proposition 1, that scale is the one nearest $k^\star_{\text{opt}}$, since factors below it are suppressed by the reachability factor and factors above it by the resolution factor. This is where the *implicit* in GITA comes from: Section 3 prescribes an abstraction factor that grows with $d^\star(s, g)$, but $d^\star$ is unavailable, and GITA never estimates it. Every scale in $\mathcal{K}$ is placed on equal footing and the weighting resolves the competition on its own.

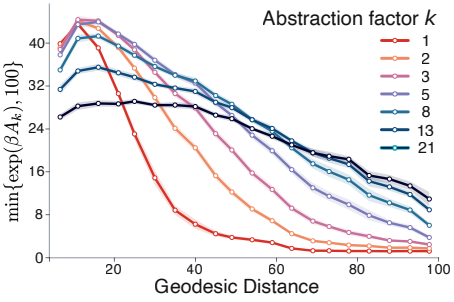


Figure 4: **The abstraction scale receiving the strongest policy weight changes with state-goal distance.** Clipped scale-specific AWR weight $\min\{\exp(\beta_h A^h(s_t, s_{t+m}, g; k)), 100\}$ across geodesic state-goal distance bins in `antmaze-giant-navigate`, for each $k$.

Figure 4 shows the clipped scale-specific AWR weight against state-goal geodesic distance in `antmaze-giant-navigate`. At short distances, smaller abstraction factors receive the largest weights, but their advantage signal decays rapidly as the goal moves farther away, causing their weights to approach the neutral value $\exp(0) = 1$. Larger factors retain non-neutral weights over longer distances and become dominant at longer distances. This mirrors the distance-dependent ordering in Figure 2, which is what makes the implicit assignment work, as the scale that receives the strongest weight in a given distance regime is also the scale that ranks subgoals most reliably there. See Appendix I for additional environments.

At inference, the policy is executed directly as $s_{t+m} \sim \pi^h_\phi(\cdot \mid s_t, g)$ and needs no abstraction factor as input. The low-level policy training objective is unchanged, with the policy predicting a primitive action conditioned on the current state and the generated subgoal, $a_t \sim \pi^\ell_\psi(\cdot \mid s_t, s_{t+m})$.

## 5 Experiments

We evaluate GITA on the locomotion and manipulation tasks of the Offline Goal Conditioned RL Benchmark (OGBench; Park et al., 2025) suite, which includes long-horizon navigation tasks and datasets that require stitching together sub-trajectories to reach distant goals. We first describe our evaluation protocol, then compare GITA against representative offline GCRL approaches, including the best fixed-$k$ alternative.

### 5.1 Evaluation on OGBench

**Experimental Setup.** We consider three locomotion tasks: PointMaze, AntMaze, and HumanoidMaze, controlling a 2-DoF ball, an 8-DoF ant, and a 21-DoF humanoid, respectively. Each comes with layouts of increasing geodesic distance—`medium`, `large`, and `giant`—and the `navigate` and `stitch` datasets; AntMaze additionally includes `explore`. We also consider two manipulation environments, Cube and Scene, where a robotic arm must reach a target configuration. The locomotion tasks probe long-horizon navigation, while the manipulation tasks probe compositional generalization. We use both the state- and pixel-based variants. An episode terminates when the agent comes within a task-specific proximity threshold of the goal state, which defines success,

Table 1: **GITA outperforms fixed-scale and non-abstracted baselines across OGBench tasks.** Success rate (%), mean ± standard deviation over 8 seeds (4 for pixel-based tasks). The best mean, and any result within one standard deviation of it, are in **bold**.

| | | | Non-Hierarchical | | | | | Hierarchical | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Environment** | **Type** | **Size** | **GCBC** | **GCIVL** | **GCIQL** | **QRL** | **CRL** | **HIQL** | **OTA** | **GITA** |
| PointMaze | navigate | large | 29±6 | 45±5 | 34±3 | **86±9** | 39±7 | 58±5 | 85±5 | **93±2** |
| | navigate | giant | 1±2 | 0±0 | 0±0 | 68±7 | 27±10 | 46±9 | 72±6 | **80±3** |
| | stitch | large | 7±5 | 12±6 | 31±2 | **84±15** | 0±0 | 13±6 | 46±7 | **72±13** |
| | stitch | giant | 0±0 | 0±0 | 0±0 | 50±8 | 0±0 | 0±0 | 44±8 | **60±7** |
| AntMaze | navigate | large | 24±2 | 16±5 | 34±4 | 75±6 | 83±4 | **91±2** | **91±1** | **91±1** |
| | navigate | giant | 0±0 | 0±0 | 0±0 | 14±3 | 16±3 | 65±5 | 70±2 | **78±2** |
| | stitch | large | 3±3 | 18±2 | 7±2 | 18±2 | 11±2 | 67±5 | 79±3 | **86±2** |
| | stitch | giant | 0±0 | 0±0 | 0±0 | 0±0 | 0±0 | 2±2 | 29±5 | **38±3** |
| | explore | large | 0±0 | 10±3 | 0±0 | 0±0 | 0±0 | 4±5 | **62±12** | **72±13** |
| HumanoidMaze | navigate | large | 1±0 | 2±1 | 2±1 | 5±1 | 24±4 | 49±4 | **82±2** | 73±3 |
| | navigate | giant | 0±0 | 0±0 | 0±0 | 1±0 | 3±2 | 12±4 | **91±1** | **89±2** |
| | stitch | large | 6±3 | 1±1 | 0±0 | 3±1 | 4±1 | 28±3 | 43±3 | **51±4** |
| | stitch | giant | 0±0 | 0±0 | 0±0 | 0±0 | 0±0 | 3±2 | **61±3** | **63±3** |
| Visual-AntMaze | navigate | large | 4±0 | 5±1 | 4±1 | 0±0 | **84±1** | 53±9 | 63±4 | 75±6 |
| | navigate | giant | 0±0 | 1±1 | 0±0 | 0±0 | **47±2** | 6±4 | 8±3 | 21±5 |
| | stitch | large | 24±3 | 1±1 | 0±0 | 1±1 | 11±3 | 28±2 | 21±2 | **45±8** |
| | stitch | giant | 0±0 | 0±0 | 0±0 | 0±0 | 0±0 | 0±0 | 2±2 | **7±3** |
| Cube | play | single | 6±2 | 53±4 | **68±6** | 5±1 | 19±2 | 15±3 | 12±2 | 15±1 |
| | play | double | 1±1 | 36±3 | **40±5** | 1±0 | 10±2 | 6±2 | 3±1 | 3±1 |
| Scene | play | – | 5±1 | 42±4 | **51±4** | 5±1 | 19±2 | 38±3 | 22±5 | 26±6 |
| Visual-Cube | noisy | single | 14±3 | 75±3 | 48±3 | 10±5 | 39±30 | 99±0 | 99±0 | **100±0** |
| | noisy | double | 5±1 | 17±4 | 22±2 | 6±2 | 6±3 | 59±3 | 65±2 | **75±1** |
| Visual-Scene | noisy | – | 13±2 | 23±2 | 12±4 | 2±0 | 15±2 | 50±1 | 54±2 | **61±1** |

or when the environment's time limit is reached. Following the OGBench evaluation protocol, we report the average success rate across five goal-reaching tasks per environment, with 50 evaluation episodes per task, averaged over 8 random seeds for state-based tasks and 4 for pixel-based tasks.

We compare against the offline GCRL baselines provided by OGBench (Park et al., 2025): goal-conditioned behavior cloning (GCBC; Ghosh et al., 2021), goal-conditioned implicit V- and Q-learning (GCIVL and GCIQL; Kostrikov et al., 2022; Park et al., 2023), quasimetric RL (QRL; Wang et al., 2023), contrastive RL (CRL; Eysenbach et al., 2022), hierarchical implicit Q-learning (HIQL; Park et al., 2023), and fixed-scale option-aware temporally abstracted value learning (OTA; Ahn et al., 2025). The comparison with OTA is the most informative, as it directly contrasts GITA's multi-scale formulation with fixed-scale temporal abstraction. HIQL abstracts only the policy, leaving its value function at $k = 1$, so the three together trace the progression from no value-level abstraction to fixed-scale to multi-scale. Note also that OTA tunes $k$ per environment, whereas GITA uses a single candidate set $\mathcal{K}$ throughout, avoiding per-environment tuning of the candidate set. See Appendix G for further details about the baselines considered.

**Performance on OGBench.** Table 1 reports success rates across all tasks considered. Aggregated over tasks, GITA outperforms every baseline under a two-sided paired Wilcoxon signed-rank test with Holm-Bonferroni correction for multiple comparisons (all adjusted $p < 0.05$; see Appendix F). The locomotion results trace a clear progression from HIQL to OTA to GITA: abstracting the

Table 2: **Every variant degrades sharply somewhere; GITA does not.** Success rate (%) on the maze tasks, mean ± standard deviation over 8 seeds.

| Environment | Type | Size | Unconditioned Value | Highest Advantage Factor | Mean Advantage Before Exponentiation | GITA with Geometric $k$ | GITA |
|---|---|---|---|---|---|---|---|
| PointMaze | navigate | large | 69±10 | 92±4 | 94±3 | 95±2 | 92±6 |
| | navigate | giant | 40±13 | 77±6 | 84±5 | 81±5 | 82±5 |
| | stitch | large | 37±16 | 67±16 | 80±9 | 65±15 | 73±17 |
| | stitch | giant | 58±9 | 54±15 | 55±16 | 45±10 | 56±13 |
| AntMaze | navigate | large | 89±2 | 92±3 | 91±2 | 90±2 | 91±2 |
| | navigate | giant | 62±4 | 78±4 | 77±3 | 80±4 | 79±3 |
| | stitch | large | 74±4 | 82±4 | 85±2 | 85±3 | 86±3 |
| | stitch | giant | 38±6 | 29±5 | 39±7 | 25±6 | 42±6 |
| | explore | large | 66±14 | 65±11 | 73±13 | 67±23 | 87±9 |
| HumanoidMaze | navigate | large | 75±7 | 74±4 | 72±5 | 74±5 | 72±5 |
| | navigate | giant | 89±2 | 87±4 | 89±4 | 90±3 | 90±4 |
| | stitch | large | 36±12 | 45±6 | 52±6 | 52±4 | 49±2 |
| | stitch | giant | 69±4 | 62±7 | 67±5 | 65±5 | 68±3 |

value function at a fixed scale improves over no abstraction, and conditioning on the scale improves further still. The second gap is the one that supports our central claim, that no single temporal abstraction suits all state-goal distances. The gap remains substantial in the `giant` layouts, where the wider range of state-goal distances within a single environment places strong pressure on methods committed to a single scale. GITA also improves on most `large` layouts, where a fixed scale is already miscalibrated for part of the distance range.

The exception is state-based manipulation. On Cube and Scene, the non-hierarchical GCIQL outperforms GITA by a wide margin, but it also outperforms HIQL and OTA, so the gap reflects a broader difficulty of hierarchical methods, rather than of multi-scale abstraction specifically. OGBench attributes this to the `play` datasets, whose open-loop, non-Markovian trajectories with temporally correlated noise can make policy and subgoal prediction ambiguous. GITA still improves over OTA on every pixel-based manipulation task, where the `noisy` datasets are Markovian and provide broader state coverage, yielding more consistent supervision for hierarchical policy learning.

### 5.2 Ablation Studies

Table 2 compares four variants of GITA, each isolating one component: whether the value function is conditioned on $k$, how scale-specific advantages are combined, and how the scales are chosen.

**Unconditioned Value** tests whether exposure to multiple temporal scales during value learning suffices on its own, or whether the value function must explicitly represent the abstraction scale. To test this, we remove $k$ from the input to the high-level value function while still sampling $k \sim \mathrm{Geom}(1-\alpha)$ to construct the abstract transitions in eq. (10), so the network learns one value function averaged across scales. Only one advantage is then defined per tuple, so the aggregation in eq. (12) reduces to standard AWR; this variant therefore removes both components at once and serves as the cumulative comparison against GITA.

**Highest Advantage Factor** tests whether policy supervision is better treated as a hard selection problem, with one scale chosen per sample, than as an aggregation across scales. We test this by retaining the abstraction-conditioned value function but weighting each tuple using only the scale with the largest advantage, $W = \exp(\beta_h \max_{k \in \mathcal{K}} A^h_k)$, discarding the rest.

**Mean Advantage Before Exponentiation** tests whether the implicit selection induced by exponentiating before averaging matters, or whether exposing the policy to multiple scales suffices. We

isolate this effect by keeping all scale-specific advantages but reversing the order, replacing eq. (12) with $\exp(\beta_h \frac{1}{|\mathcal{K}|} \sum_k A^h)$. Because exponential is convex, this removes the emphasis on large positive scale-specific advantages and instead averages all advantages uniformly before exponentiation.

**GITA with Geometric $k$** tests whether the high-level policy needs a pre-specified candidate set of abstraction factors. We replace the fixed $\mathcal{K}$ with $|\mathcal{K}|$ factors drawn independently from $\mathrm{Geom}(1-\zeta)$ at every policy update. This removes the spacing of $\mathcal{K}$ as a design choice and ensures the policy is supervised only at scales supported during value learning, though the candidate set is now noisier at each update.

We observe the largest drop in performance in *Unconditioned Value*. Exposure to multiple temporal scales during value learning is therefore not sufficient on its own; the scale must be explicitly represented in the value function. GITA also significantly outperforms *Highest Advantage Factor* ($p < 0.05$, Holm-Bonferroni-corrected Wilcoxon signed-rank test; see Table 7), so committing to a single scale per sample loses information that soft aggregation retains, even when that scale is the strongest one available. *Mean Advantage Before Exponentiation* and *GITA with Geometric $k$* are not significantly different overall, but each collapses on one of AntMaze `large-explore` and `giant-stitch` respectively, the two settings with the widest spread of state-goal distances, where GITA does not. Neither the exponentiation order nor the fixed candidate set is thus strictly necessary, but together they are what make the method reliable across regimes.

## 6 Conclusion

In this paper, we introduced Generalized Implicit Temporal Abstraction (GITA), an offline goal-conditioned reinforcement learning (GCRL) method that addresses the trade-off between long-range value propagation and local value discrimination. We showed analytically that the advantage used for subgoal selection factors into a reachability term and a resolution term pulling in opposite directions as the abstraction factor grows, so the best factor increases with state-goal distance, and no fixed choice serves every pair. GITA resolves this by learning a shared value function across temporal scales and training a single policy from their aggregated advantages, implicitly emphasizing the most informative scales for each state-goal pair. Our empirical results establish GITA as a robust and flexible framework for hierarchical offline RL, providing a foundation for future methods that leverage multiple temporal scales to tackle long-horizon tasks.

Directions for extending GITA include: (1) GITA, like other hierarchical methods, is less effective on Cube and Scene, which are short-horizon and compositional rather than long-horizon; incorporating compositionality into the notion of temporal abstraction is a natural next step. (2) In GITA, as in many other approaches, the notion of distance depends on the policy that generated the dataset, and correcting for discrepancies introduced by the off-policy nature of the data is a promising direction. (3) Finally, the representations GITA learns are only implicitly shaped by the value loss; explicitly learning representations that capture an intrinsic notion of distance (e.g., Shehmar et al., 2026) would likely benefit the framework.

### AI use statement

We used LLMs to assist with writing: drafting and revising prose, restructuring paragraphs for clarity, and language refinement, as well as with LaTeX formatting of equations and with refining plotting code. All research contributions — the method, theoretical analysis, experimental design, and results — are the authors' own, and all AI-assisted text and code were verified by the authors.

### Reproducibility statement

All results reported in this work are reproducible using the experimental setup, hyperparameters, and evaluation procedures described in the main paper and appendix. We will release the complete source code upon acceptance. Our implementation is built directly on top of the OGBench codebase and follows its training and evaluation pipeline, making it straightforward to run GITA under the same protocol used for the existing baselines and to compare the resulting performance under identical evaluation conditions.

## REFERENCES


Valliappan Chidambaram Adaikkappan, David Meger, Sai Rajeswar, and Pietro Mazzaglia. Multi-scale predictive representations for goal-conditioned reinforcement learning. *CoRR*, abs/2605.09364, 2026.

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

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

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Jinwoo Choi, Sang-Hyun Lee, and Seung-Woo Seo. Chain-of-goals hierarchical policy for long-horizon offline goal-conditioned RL. In *International Conference on Machine Learning (ICML)*, 2026.

Hojun Chung, Junseo Lee, and Songhwai Oh. Offline reinforcement learning with universal horizon models. In *International Conference on Machine Learning (ICML)*, 2026.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 (1):269–271, 1959.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

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

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *CoRR*, abs/1606.08415, 2016.

Ahad Jawaid. Offline RL with hierarchical action chunking. *Reinforcement Learning Journal*, 7, 2026.

Hyeonseong Jeon and Youngwoon Lee. Recursive value learning for long-horizon offline goal-conditioned RL. *CoRR*, abs/2609.02237, 2026.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

Kaiqiang Ke, Shenghong He, Chengdong Xu, Yuheng Luo, Xiangyuan Lan, and Chao Yu. Adaptive coarse-to-fine subgoal refinement for long-horizon offline goal-conditioned reinforcement learning. *CoRR*, abs/2605.28127, 2026.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

Evgenii Opryshko, Junwei Quan, Claas Voelcker, Yilun Du, and Igor Gilitschenski. Test-time graph search for goal-conditioned reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2026.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Neural Information Processing Systems (NeurIPS)*, 2023.

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

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

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, 2015.

Dikshant Shehmar, Matthew Schlegel, Matthew E. Taylor, and Marlos C. Machado. Laplacian representations for decision-time planning. In *International Conference on Machine Learning (ICML)*, 2026.

Craig Sherstan, Shibhansh Dohare, James MacGlashan, Johannes Günther, and Patrick M. Pilarski. Gamma-nets: Generalizing value estimation over timescale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

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.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023.

Clarisse Wibault, Alexander Goldie, Antonio Villares, Maike Osborne, and Jakob Foerster. Abstraction for offline goal-conditioned reinforcement learning. *CoRR*, abs/2605.22711, 2026.

Bill Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Scaling goal-conditioned reinforcement learning with multistep quasimetric distances. In *International Conference on Learning Representations (ICLR)*, 2026.

# A Related Work

**Offline goal-conditioned value learning.** Offline goal-conditioned reinforcement learning (GCRL) aims to learn policies for reaching arbitrary goals from fixed datasets. Existing approaches differ in how they estimate long-range goal-conditioned values: contrastive RL learns state–goal representations through contrastive objectives (Eysenbach et al., 2022), whereas quasimetric RL exploits the structure of temporal distances (Wang et al., 2023). Building on the latter, Zheng et al. (2026) fit quasimetric distances with multistep Monte Carlo targets to improve long-horizon learning. Jeon & Lee (2026) instead organize value learning recursively over a balanced decomposition of observed trajectory segments, shortening the chain of bootstrapped dependencies before propagating values across trajectories. These methods address long-range estimation through different objectives.

**Hierarchical goal-conditioned reinforcement learning.** Hierarchical RL addresses long-horizon decision-making by decomposing complex tasks into simpler subtasks. In offline GCRL, Park et al. (2023) introduced a two-level policy that proposes intermediate subgoals and executes primitive actions toward them, training both levels from goal-conditioned values using advantage-weighted regression (Peng et al., 2019). More recent methods modify the structure or execution of this hierarchy. Choi et al. (2026) generate sequences of latent subgoals autoregressively before predicting an action, while Ke et al. (2026) recursively refine distant goals until a learned reachability criterion indicates that the current target can be executed locally. Jawaid (2026) combines high-level subgoal planning with low-level action chunks and temporally extended value backups. Wibault et al. (2026) study relativized options and level-specific representations to reuse experience across similar state–goal contexts. Rather than changing subgoal generation, hierarchy depth, or low-level execution, we retain the two-level policy structure of HIQL and study how temporal abstraction in the high-level value function affects subgoal evaluation.

**Temporal abstraction at multiple scales.** Temporal abstraction has long been formalized through temporally extended actions (Sutton et al., 1999). Ahn et al. (2025) introduced temporally abstracted value learning for offline GCRL, treating a fixed stride of $k$ primitive steps as one abstract transition. This improves long-range value propagation, but a fixed $k$ must trade off sensitivity to distant goals against the ability to discriminate nearby states. Multiple temporal horizons have also been studied in other components of offline RL: Chung et al. (2026) learn predictive models that can sample future states at arbitrary horizons and use a controlled horizon distribution for model-based value learning, while Adaikkappan et al. (2026) use multiscale predictive supervision to align state and goal representations. Outside offline RL, Sherstan et al. (2020) introduced $\Gamma$-nets, which condition a single value estimator on the discount factor $\gamma$, enabling value predictions to generalize across multiple timescales. In contrast, GITA represents multiple temporal abstraction scales directly within a single $k$-conditioned goal-value function, where $k$ controls the number of primitive steps represented by each abstract transition.

# B Proofs for the Fixed-Scale Resolution–Horizon Trade-off

This section provides the proofs for Proposition 1 and its corollaries. Our analysis is an idealized shortest-path model and is not intended to exactly characterize dataset-induced transitions. Let $d = d^\star(s, g)$ denote the minimum number of primitive transitions from state $s$ to goal $g$. We follow the temporally abstracted update convention of OTA (Ahn et al., 2025), which evaluates the sparse reward at the abstract successor $s_{t+k}$, whereas HIQL indexes the one-step reward at $s_t$. This results in a one-step reward-indexing shift at $k = 1$, which does not affect the maximizing abstraction factor in the analysis below. Following the theoretical formulation of OTA (Ahn et al., 2025), the primitive-step value is

$$V^\star(d) = -\sum_{j=0}^{d-1} \gamma^j = -\frac{1-\gamma^d}{1-\gamma}. \tag{14}$$

With abstraction factor $k$, the corresponding abstract horizon is $H_k(d) = \left\lceil \frac{d}{k} \right\rceil$, which gives

$$V_k^\star(d) = -\frac{1-\gamma^{\left\lceil \frac{d}{k} \right\rceil}}{1-\gamma}. \tag{15}$$

Relaxing $\lceil d/k \rceil$ to $d/k$ so that it varies smoothly with $k$, yields

$$\widetilde{V}_k(s, g) = -\frac{1 - \gamma^{d^\star(s,g)/k}}{1 - \gamma}, \qquad k > 0, \quad 0 < \gamma < 1. \tag{16}$$

Consider a candidate subgoal $s'$ that makes $\Delta$ primitive steps of progress toward the goal, such that

$$d^\star(s, g) = d, \qquad d^\star(s', g) = d - \Delta, \qquad d > \Delta > 0. \tag{17}$$

### B.1 Proof of Proposition 1

*Proof.* The high-level advantage of the candidate subgoal $s'$ relative to the current state $s$ is

$$\widetilde{A}_k(s, s', g) = \widetilde{V}_k(s', g) - \widetilde{V}_k(s, g). \tag{18}$$

Applying Equation (16) to both $s$ and $s'$, with distance given by Equation (17), we obtain

$$\begin{aligned} \widetilde{A}_k(s, s', g) &= -\frac{1 - \gamma^{(d-\Delta)/k}}{1 - \gamma} + \frac{1 - \gamma^{d/k}}{1 - \gamma} \\ &= \frac{\gamma^{(d-\Delta)/k} - \gamma^{d/k}}{1 - \gamma} \\ &= \gamma^{(d-\Delta)/k} \frac{1 - \gamma^{\Delta/k}}{1 - \gamma}. \end{aligned} \tag{19}$$

Therefore,

$$\widetilde{A}_k(s, s', g) = \underbrace{\gamma^{(d-\Delta)/k}}_{\text{reachability}} \underbrace{\frac{1 - \gamma^{\Delta/k}}{1 - \gamma}}_{\text{resolution}} \tag{20}$$

□

The *reachability factor* measures whether the residual distance $d - \Delta$ falls within a range where discounting carries signal, while the *resolution factor* measures how sharply the value function separates $s$ from a subgoal $\Delta$ primitive steps closer.

### B.2 Proof of the Vanishing-Advantage Corollary

We prove the two limits in Equation (6) separately. Throughout, let $\lambda := -\ln \gamma > 0$, so that $\gamma^x = e^{-\lambda x}$ for any exponent $x$.

**Limit as $k \to 0^+$.**

*Proof.* Since $d - \Delta > 0$ is fixed, $(d - \Delta)/k \to \infty$ as $k \to 0^+$, so the reachability factor vanishes:

$$\lim_{k \to 0^+} \gamma^{(d-\Delta)/k} = \lim_{k \to 0^+} \exp\left(-\lambda \frac{d - \Delta}{k}\right) = 0. \tag{21}$$

Meanwhile the resolution factor stays bounded for all $k$, since $0 < \gamma^{\Delta/k} < 1$ implies

$$0 < \frac{1 - \gamma^{\Delta/k}}{1 - \gamma} < \frac{1}{1 - \gamma}. \tag{22}$$

Combining Equation (21) and (22) in Equation (19) gives

$$0 < \widetilde{A}_k(s, s', g) < \frac{1}{1 - \gamma} \exp\left(-\lambda \frac{d - \Delta}{k}\right) \xrightarrow[k \to 0^+]{} 0, \tag{23}$$

so by the squeeze theorem,

$$\lim_{k \to 0^+} \widetilde{A}_k(s, s', g) = 0. \tag{24}$$

□

Although $k$ is discrete in the algorithm, this limit concerns its continuous relaxation. Equivalently, the same vanishing regime is obtained for any fixed admissible $k$ as $d \to \infty$, since it is governed by the ratio $(d - \Delta)/k$.

**Limit as $k \to \infty$.**

*Proof.* Here the reachability factor tends to 1 while the resolution factor's numerator $1-\gamma^{\Delta/k}$ tends to 0. Using $e^{-x} = 1 - x + \mathcal{O}(x^2)$ as $x \to 0$ with $x = \lambda\Delta/k$ and $x = \lambda(d-\Delta)/k$ respectively,

$$1 - \gamma^{\Delta/k} = \frac{\lambda\Delta}{k} + \mathcal{O}(k^{-2}), \qquad \gamma^{(d-\Delta)/k} = 1 + \mathcal{O}(k^{-1}). \tag{25}$$

Substituting Equation (25) into Equation (19), all cross terms are $\mathcal{O}(k^{-2})$ or smaller, leaving

$$\widetilde{A}_k(s, s', g) = \frac{\lambda\Delta}{(1-\gamma)k} + \mathcal{O}(k^{-2}). \tag{26}$$

Since the leading term is $\Theta(1/k)$,

$$\lim_{k\to\infty} \widetilde{A}_k(s, s', g) = 0 \tag{27}$$

□

Together, Equations (24) and (27) prove the vanishing-advantage corollary. Therefore, a coarser abstraction allows the advantage to better discriminate distant states when considering distant goals, because the *reachability factor* increases toward 1 as $k$ grows. However, the same coarser abstraction blurs the distinction between nearby states, since the *resolution factor* decreases toward 0 as $k$ grows. No $k$ simultaneously maximizes both.

### B.3 Proof of Corollary 1.2

*Proof.* From Equation (19),

$$\widetilde{A}_k(d, \Delta) = \frac{\gamma^{(d-\Delta)/k} - \gamma^{d/k}}{1-\gamma}. \tag{28}$$

Differentiating with respect to $k$ gives

$$\frac{\partial \widetilde{A}_k(d,\Delta)}{\partial k} = \frac{-\log\gamma}{(1-\gamma)k^2}\left[(d-\Delta)\gamma^{(d-\Delta)/k} - d\gamma^{d/k}\right]. \tag{29}$$

Since $-\log\gamma > 0$, $1-\gamma > 0$, and $k^2 > 0$, the sign of the derivative is governed entirely by the terms in brackets, and a stationary point satisfies

$$(d-\Delta)\gamma^{(d-\Delta)/k} = d\gamma^{d/k}. \tag{30}$$

Dividing both sides by $\gamma^{(d-\Delta)/k} > 0$ and using $\gamma^{d/k} = \gamma^{(d-\Delta)/k}\gamma^{\Delta/k}$,

$$d - \Delta = d\gamma^{\Delta/k} \qquad \Longrightarrow \qquad \gamma^{\Delta/k} = \frac{d-\Delta}{d}. \tag{31}$$

Taking logarithms and solving for $k$ gives the exact continuous optimum

$$k^\star_{\text{opt}}(d, \Delta) = \frac{-\Delta\log\gamma}{\log\left(\frac{d}{d-\Delta}\right)}. \tag{32}$$

Since $\gamma^{\Delta/k}$ increases strictly and continuously from 0 to 1 as $k$ ranges over $(0, \infty)$, and $(d-\Delta)/d \in (0, 1)$, Equation (31) has a unique solution. From Equation (29), the derivative is positive before this solution and negative after it. Therefore, $k^\star_{\text{opt}}(d, \Delta)$ is the unique global maximizer of the relaxed advantage. □

**Dependence on state-goal distance.** For fixed $\Delta$ and $\gamma$, write $k^\star_{\text{opt}}(d, \Delta) = C/L(d)$ with $C = -\Delta\log\gamma > 0$ and $L(d) = \log\left(\frac{d}{d-\Delta}\right)$. Since

$$L'(d) = \frac{1}{d} - \frac{1}{d-\Delta} = -\frac{\Delta}{d(d-\Delta)} < 0, \tag{33}$$

the quotient rule gives

$$\frac{\partial k^\star_{\text{opt}}}{\partial d} = \frac{(-\Delta\log\gamma)\Delta}{d(d-\Delta)\left[\log\left(\frac{d}{d-\Delta}\right)\right]^2} > 0. \tag{34}$$

Thus, the advantage-maximizing abstraction factor increases with state-goal distance. This monotonicity result holds for fixed $\Delta$ and $\gamma$.

**Local-progress approximation.** For the regime considered in Corollary 1.2, let $\epsilon = \frac{\Delta}{d} \ll 1$. Expanding the denominator of Equation (32),

$$\log\left(\frac{d}{d-\Delta}\right) = -\log(1-\epsilon) = \epsilon + \frac{\epsilon^2}{2} + \mathcal{O}(\epsilon^3). \tag{35}$$

Substituting $\Delta = d\epsilon$ into Equation (32) and dividing through by $\epsilon$,

$$k^\star_{\text{opt}}(d,\Delta) = \frac{-d\epsilon\log\gamma}{\epsilon\left[1+\epsilon/2+\mathcal{O}(\epsilon^2)\right]} = \frac{-d\log\gamma}{1+\epsilon/2+\mathcal{O}(\epsilon^2)}. \tag{36}$$

Applying the reciprocal expansion $1/(1+x) = 1 - x + \mathcal{O}(x^2)$ as $x \to 0$, with $x = \epsilon/2$, gives

$$k^\star_{\text{opt}}(d,\Delta) = -d\log\gamma\left[1 - \frac{\epsilon}{2} + \mathcal{O}(\epsilon^2)\right]. \tag{37}$$

Therefore,

$$k^\star_{\text{opt}}(d,\Delta) \approx -d\log\gamma, \qquad \Delta \ll d \tag{38}$$

which proves Equation (7).

**Effective discount horizon.** Dividing $d - \Delta = d(1-\epsilon)$ by Equation (37) and re-expanding to first order in $\epsilon$ gives

$$\frac{d-\Delta}{k^\star_{\text{opt}}(d,\Delta)} = -\frac{1}{\log\gamma}\left[1 - \frac{\epsilon}{2} + \mathcal{O}(\epsilon^2)\right], \tag{39}$$

so that, for all $\Delta \ll d$,

$$\frac{d-\Delta}{k^\star_{\text{opt}}(d,\Delta)} \approx -\frac{1}{\log\gamma}. \tag{40}$$

Since $-1/\log\gamma \sim 1/(1-\gamma)$ as $\gamma \to 1$, combining the two approximations yields

$$\frac{d-\Delta}{k^\star_{\text{opt}}(d,\Delta)} \approx -\frac{1}{\log\gamma} \approx \frac{1}{1-\gamma}, \qquad \Delta \ll d, \quad \gamma \to 1, \tag{41}$$

which proves Equation (8).

Together, these results show that no fixed abstraction factor can preserve informative advantages across all state-goal distances: small $k$ loses signal for distant goals, while large $k$ sacrifices local resolution, and the best scale therefore grows with distance.

## C Fixed-Scale Trade-off Experiment Details

This section provides additional details on the experiment used to evaluate how the quality of the high-level advantage varies with the abstraction factor and the state-goal distance.

**Geodesic distance.** For each environment, we sample $(s, s', g)$ triples using the same sampling procedure used to construct training examples for the high-level policy. We then compute reference distances using the known geometry of the OGBench mazes. Each maze is discretized into a fine grid, where navigable cells form the nodes of a four-connected graph and edges connect adjacent cells without crossing walls. The geodesic distance between two states is defined as the shortest-path distance between their corresponding cells and is computed using Dijkstra's algorithm.

For each triple, we compute $d_{\text{geo}}(s, g)$ and $d_{\text{geo}}(s', g)$ and define the geodesic progress as

$$\Delta d = d_{\text{geo}}(s, g) - d_{\text{geo}}(s', g). \tag{42}$$

The triples are grouped into equal-width intervals. Geodesic distance is used only as a proxy for task progress and need not equal expert-policy temporal distance.

**Advantage–Progress Correlation.** For each abstraction factor $k$, we compute the scale-conditioned high-level advantage

$$A_k^h(s, s', g) = V_k^h(s', g) - V_k^h(s, g). \tag{43}$$

To measure how well this advantage ranks subgoals according to their progress toward the goal, we compute the Spearman rank correlation between $A_k^h(s, s', g)$ and the corresponding geodesic progress $\Delta d$. We first group training triples $(s, s', g)$ according to their geodesic state-goal distance $d_{\text{geo}}(s, g)$. Each distance bin $b$ corresponds to an interval of state-goal distances, and $\mathcal{I}_b$ denotes the set of triples whose $d_{\text{geo}}(s, g)$ falls within that interval. For each abstraction factor $k$ and distance bin $b$, we define the Advantage-Progress Correlation (APC) as

$$\text{APC}_{k,b} = \rho_{\text{S}} \left( \left\{ A_{k,i}^h \right\}_{i \in \mathcal{I}_b}, \{\Delta d_i\}_{i \in \mathcal{I}_b} \right). \tag{44}$$

Values close to 1 indicate that the learned advantage preserves the ordering induced by geodesic progress, values near 0 indicate little monotonic agreement, and negative values indicate an inverted ordering. We compute APC independently for each training seed and report the mean and one standard deviation across four seeds.

# D Environments

This section provides additional details on the environments and offline datasets used in our experiments. We evaluate GITA on locomotion and manipulation tasks from OGBench (Park et al., 2025). Our locomotion experiments use PointMaze, AntMaze, and HumanoidMaze, which combine maze navigation with increasingly complex control, while our manipulation experiments use Cube and Scene. We consider state-based observations for all five environments and pixel-based variants of AntMaze, Cube, and Scene.

## D.1 OGBench Environments

**Maze navigation.** PointMaze controls a 2-D point mass directly through planar actions, AntMaze controls an 8-DoF quadruped, and HumanoidMaze controls a 21-DoF humanoid. The task in each case is to reach a target location in the maze, with AntMaze and HumanoidMaze additionally requiring the agent to learn the underlying locomotion dynamics. State-based observations contain the full low-dimensional state, including the agent's $x$-$y$ position. In Visual-AntMaze, the low-dimensional observation is replaced by a $64 \times 64 \times 3$ RGB image rendered from a third-person camera, without additional proprioceptive state information.

**Maze layouts.** OGBench provides the `medium`, `large`, and `giant` layouts. The `medium` and `large` layouts follow the corresponding D4RL (Fu et al., 2020) maze structures, while `giant` is approximately twice the size of `large` and contains substantially longer paths. These layouts provide increasingly long state-goal distances and therefore directly stress long-horizon value propagation. The main benchmark focuses primarily on the `large` and `giant` layouts.

**Manipulation.** Cube and Scene use a UR5e robot arm controlled through a 5-dimensional end-effector action space. Cube requires arranging blocks into target configurations and provides variants with different numbers of cubes; our experiments use the `single` and `double` variants for state observations. Scene contains a cube, a drawer, a window, and two locking buttons, and its evaluation goals require composing multiple manipulation behaviors into a desired object configuration. Both environments also support $64 \times 64 \times 3$ RGB observations; in the visual variants, the arm is rendered transparently to reduce visual occlusion.

## D.2 Offline Dataset Variants

The maze environments use three offline dataset variants with different trajectory structure and behavioral quality. The `navigate` datasets are collected by a noisy expert that repeatedly moves toward randomly sampled goals, producing relatively long and coherent goal-reaching trajectories. The `stitch` datasets instead consist of short local trajectory segments, so solving evaluation tasks requires combining information across multiple trajectories. For AntMaze, we additionally consider

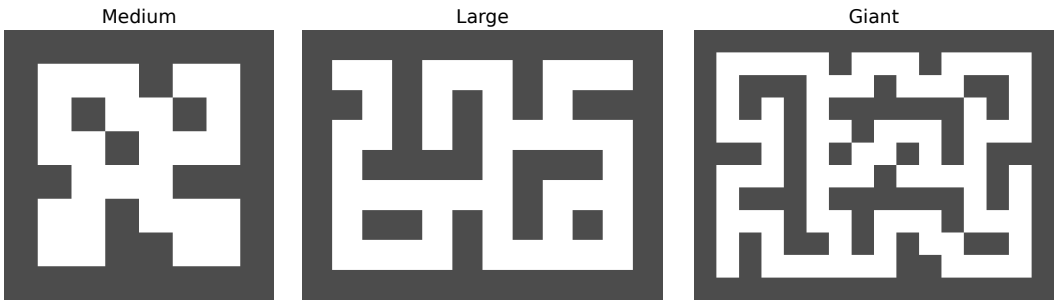


Figure 5: **Maze layouts.** OGBench provides the `medium`, `large`, and `giant` layouts considered in our experiments, with progressively longer state-goal distances.

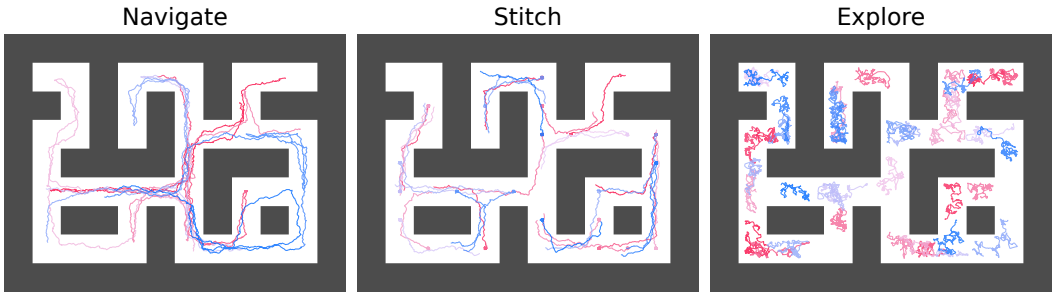


Figure 6: **OGBench datasets exhibit distinct trajectory structures.** Representative trajectories from the OGBench offline dataset variants in the `large` maze. The `navigate` dataset contains long goal-directed trajectories, `stitch` consists of shorter local trajectory segments that must be combined to solve long-horizon tasks, and `explore` contains high-coverage but highly suboptimal behavior.

the `explore` dataset, which contains high-coverage but suboptimal behavior generated from randomly changing movement directions with substantial action noise. State- and pixel-based AntMaze variants share the same underlying trajectories and differ only in their observation modality. For manipulation, OGBench provides `play` and `noisy` datasets: `play` is generated by non-Markovian scripted policies with temporally correlated noise, whereas `noisy` uses Markovian scripted policies with uncorrelated Gaussian noise and higher state coverage. Our state-based Cube and Scene experiments use `play`, while Visual-Cube and Visual-Scene use `noisy`. Figure 6 illustrates the characteristic trajectory structure of the maze dataset variants.

Table 3 summarizes the observation and action spaces and evaluation horizons of the environment variants used in our experiments. Table 4 reports the corresponding dataset statistics. Note that OGBench dataset episode lengths need not coincide with the maximum evaluation episode length.

### D.3 Evaluation Protocol

We follow the OGBench evaluation protocol (Park et al., 2025), using five predefined state–goal evaluation tasks for each benchmark environment. At each evaluation checkpoint, we run 50 episodes for each of the five tasks, yielding 250 evaluation episodes per checkpoint and seed. We evaluate state-based experiments over 8 random seeds and pixel-based experiments over 4 random seeds, and report the mean success rate and standard deviation across seeds.

The results in our main table are averaged over the final three evaluation checkpoints: for state-based tasks, models are trained for $10^6$ gradient steps and evaluated at 800K, 900K, and 1M steps. For pixel-based tasks, models are trained for 500K gradient steps and evaluated at 300K, 400K,

Table 3: **Environment specifications.** Specifications for the OGBench domains considered in our experiments.

| Environment | Variant | Observation | Action Dim. | Max. Episode Length |
|---|---|---|---|---|
| PointMaze | `large` | state (2) | 2 | 1000 |
| | `giant` | state (2) | 2 | 1000 |
| AntMaze | `large` | state (29) / pixels ($64 \times 64 \times 3$) | 8 | 1000 |
| | `giant` | state (29) / pixels ($64 \times 64 \times 3$) | 8 | 1000 |
| HumanoidMaze | `large` | state (69) | 21 | 2000 |
| | `giant` | state (69) | 21 | 4000 |
| Cube | `single` | state (28) / pixels ($64 \times 64 \times 3$) | 5 | 200 |
| | `double` | state (37) / pixels ($64 \times 64 \times 3$) | 5 | 500 |
| Scene | – | state (40) / pixels ($64 \times 64 \times 3$) | 5 | 750 |

Table 4: **Offline dataset specifications.** Specifications of the OGBench offline datasets considered in our experiments. Visual-AntMaze uses the same trajectory datasets and statistics as the corresponding AntMaze configuration.

| Environment | Dataset | Variant | # Transitions | # Episodes | Data Episode Length |
|---|---|---|---|---|---|
| PointMaze | `navigate` | `large` | 1M | 1000 | 1000 |
| | `navigate` | `giant` | 1M | 500 | 2000 |
| | `stitch` | `large` | 1M | 5000 | 200 |
| | `stitch` | `giant` | 1M | 5000 | 200 |
| AntMaze | `navigate` | `large` | 1M | 1000 | 1000 |
| | `navigate` | `giant` | 1M | 500 | 2000 |
| | `stitch` | `large` | 1M | 5000 | 200 |
| | `stitch` | `giant` | 1M | 5000 | 200 |
| | `explore` | `large` | 5M | 10000 | 500 |
| HumanoidMaze | `navigate` | `large` | 2M | 1000 | 2000 |
| | `navigate` | `giant` | 4M | 1000 | 4000 |
| | `stitch` | `large` | 2M | 5000 | 400 |
| | `stitch` | `giant` | 4M | 10000 | 400 |
| Cube | `play` | `single` | 1M | 1000 | 1000 |
| | `play` | `double` | 1M | 1000 | 1000 |
| Scene | `play` | – | 1M | 1000 | 1000 |
| Visual-Cube | `noisy` | `single` | 1M | 1000 | 1000 |
| | `noisy` | `double` | 1M | 1000 | 1000 |
| Visual-Scene | `noisy` | – | 1M | 1000 | 1000 |

and 500K steps. Thus, each reported per-seed score averages performance over $3 \times 5 \times 50 = 750$ evaluation episodes before results are aggregated across seeds.

For the ablation studies, we instead report performance using only the final training checkpoint (1M steps for state-based tasks).

## E Implementation Details

We implement GITA on top of the HIQL implementation provided by OGBench, which facilitates reproducibility and enables direct comparison with baselines evaluated under the same implemen-

tation framework. Unless explicitly stated, we retain the HIQL network architectures, optimization settings, goal-sampling distributions, and hyperparameters. The main architectural change is restricted to the high-level value function, which is additionally conditioned on the temporal abstraction factor $k$. The high-level and low-level policies retain the HIQL architectures.

**Network architecture.** Following the OGBench implementation of HIQL, the value and policy networks use three hidden layers of size $(512, 512, 512)$ with Gaussian Error Linear Unit (GELU) activations (Hendrycks & Gimpel, 2016) and Layer Normalization (Ba et al., 2016). For state-based tasks, the goal representation network maps the concatenated state-goal input to a 10-dimensional length-normalized representation using the same hidden dimensions. The low-level policy predicts primitive actions conditioned on the current state and the learned subgoal representation, while the high-level policy predicts a 10-dimensional subgoal representation.

For pixel-based tasks, we encode images with the `impala_small` encoder, based on the IMPALA architecture (Espeholt et al., 2018), before the corresponding value, policy, and goal-representation networks. This encoder uses three residual stacks with 16, 32, and 32 channels, one residual block per stack, followed by a 512-dimensional MLP. We use image augmentation with probability $0.5$ for visual manipulation tasks and no augmentation for the visual maze tasks.

The value function consists of two independently parameterized scalar heads, $V_1$ and $V_2$, together with corresponding target networks. The minimum of the two next-state target values is used when constructing the conservative bootstrap signal, while policy advantages are computed from the mean of the two online value estimates.

**Abstraction-factor conditioning.** Only the high-level value function receives the abstraction factor as input. We provide $k$ to the network on a logarithmic scale and compute a 32-dimensional abstraction embedding

$$e_k = f_{\text{abs}}(\log k), \tag{45}$$

where $f_{\text{abs}}$ is a one-layer MLP with GELU activation. We use $\log k$ because the abstraction factors span multiplicative temporal scales; the logarithm converts multiplicative changes in $k$ into approximately additive changes in the conditioning input and reduces the dynamic range seen by the embedding network.

We inject this embedding into the hidden layers of the high-level value function using Feature-wise Linear Modulation (FiLM; Perez et al., 2018). For a hidden representation $h_\ell$ at layer $\ell$, the abstraction embedding is projected into scale and shift parameters,

$$[\eta_\ell(k),\, \delta_\ell(k)] = W_\ell^{\text{FiLM}} e_k + b_\ell^{\text{FiLM}}, \tag{46}$$

and the hidden representation is modulated as

$$h_\ell \leftarrow (1 + \eta_\ell(k)) \odot h_\ell + \delta_\ell(k). \tag{47}$$

FiLM is applied after the activation and LayerNorm in the high-level value network. The FiLM projection layers are initialized to zero.

The state-goal representation is independent of the abstraction factor: $k$ does not modify the state or goal encoder and is introduced only through FiLM in the high-level value function. Since $k$ is introduced only through FiLM, the same state-goal representation can be reused across abstraction factors, allowing GITA to support multiple temporal scales with only modest additional training cost.

**Optimization and hyperparameters.** All networks are optimized with Adam (Kingma & Ba, 2015) using a learning rate of $3 \times 10^{-4}$. Following the OGBench protocol, state-based models use a minibatch size of $1024$ and are trained for $10^6$ gradient steps, while pixel-based models use a minibatch size of $256$ and are trained for $5 \times 10^5$ gradient steps. The target value networks are updated using Polyak averaging with coefficient $\tau = 0.005$, and the expectile parameter is fixed to $0.7$.

We retain the HIQL subgoal offset $m$, independently of the abstraction factor $k$. We use $m = 25$ for PointMaze and AntMaze, $m = 100$ for HumanoidMaze, and $m = 10$ for Cube and Scene, with the same offsets used for their corresponding visual variants. The subgoal representation dimension is 10.

Table 5: **Hyperparameters used for GITA.** Unless marked as GITA-specific, the values follow the HIQL reference implementation in OGBench.

| Hyperparameter | Value |
|---|---|
| *– Optimization and Architecture –* | |
| Learning rate | $3 \times 10^{-4}$ |
| Optimizer | Adam |
| Gradient steps | $10^6$ (state), $5 \times 10^5$ (pixels) |
| Minibatch size | 1024 (state), 256 (pixels) |
| Actor hidden dimensions | $(512, 512, 512)$ |
| Value hidden dimensions | $(512, 512, 512)$ |
| Nonlinearity | GELU |
| Value LayerNorm | True |
| Visual encoder | `impala_small` |
| Image augmentation probability | 0.5 (visual manipulation), 0 (otherwise) |
| Target smoothing coefficient $\tau$ | 0.005 |
| Expectile | 0.7 |
| Subgoal representation dimension | 10 |
| Subgoal offset $m$ | 25 (PointMaze, AntMaze), 100 (HumanoidMaze), 10 (Cube, Scene) |
| *– Goal Sampling and Policy Extraction –* | |
| Value current-goal probability | 0.2 |
| Value future-goal probability | 0.5 (geometric) |
| Value random-goal probability | 0.3 |
| High-level actor goals (`navigate`) | trajectory: 1.0 |
| High-level actor goals (`stitch`) | trajectory: 0.5, random: 0.5 |
| High-level actor goals (`explore`) | random: 1.0 |
| AWR inverse temperature | 3 (`navigate`/`stitch`), 10 (`explore`) |
| *– GITA-Specific Parameters –* | |
| Discount factor $\gamma$ | $\{0.97, 0.98, 0.99, 0.995\}$ (tuned per task) |
| Geometric sampling parameter $\alpha$ | $\alpha = \gamma$ |
| Candidate abstraction factors $\mathcal{K}$ | $\{3, 5, 8, 13, 21\}$ |
| Abstraction embedding dimension | 32 |

We preserve the environment-specific HIQL policy-extraction and goal-sampling hyperparameters from OGBench. In particular, the AWR inverse temperature is 3 for the standard `navigate` and `stitch` datasets and 10 for AntMaze `explore`. For value learning, goals are sampled as the current state, a geometrically sampled future state, or a random dataset state with probabilities 0.2, 0.5, and 0.3, respectively. For high-level actor training, `navigate` uses trajectory goals, `stitch` uses an equal mixture of trajectory and random goals, and `explore` uses random goals.

The discount factor $\gamma$ is the only HIQL hyperparameter that we re-tune for GITA. The OGBench reference implementation already uses environment-dependent discount factors for HIQL, with $\gamma = 0.995$ for the longest-horizon maze environments and $\gamma = 0.99$ otherwise. In GITA, we re-tune $\gamma$ because it plays an additional role beyond TD learning: we use the same value both as the discount factor in the temporal-difference updates and as the parameter controlling the geometric distribution over abstraction factors. We therefore sweep $\gamma \in \{0.98, 0.99, 0.995\}$ and tune it independently for each task. For simplicity, we chose $\alpha = \gamma$. For the manipulation tasks, we additionally consider $\gamma = 0.97$. All remaining HIQL hyperparameters are kept unchanged.

For the candidate abstraction factors, we use values from the Fibonacci sequence, starting from 3. This gives $\mathcal{K} = \{3, 5, 8, 13, 21\}$. The range was chosen to cover the abstraction factors used by the fixed-scale OTA baseline: the smallest candidate, 3, is below the smallest task-specific abstraction factor selected for OTA, while 21 extends beyond the largest fixed factor. For OTA, we use the results reported under the unified-hyperparameter setting, except for the abstraction factor, which is selected separately for each task.

Table 6: **Two-sided paired Wilcoxon signed-rank tests comparing GITA with the baselines in Table 1.** Each test is performed over the 23 tasks reported in the main results table. $p_{\mathrm{adj}}$ denotes the Holm–Bonferroni-adjusted $p$-value across the seven comparisons.

| **Comparison** | $W$ | **Raw** $p$ | $p_{\mathrm{adj}}$ |
|---|---|---|---|
| GITA vs. GCBC | 0.0 | $2.69 \times 10^{-5}$ | $1.88 \times 10^{-4}$ |
| GITA vs. GCIVL | 14.0 | $1.62 \times 10^{-4}$ | $8.08 \times 10^{-4}$ |
| GITA vs. GCIQL | 18.5 | $2.78 \times 10^{-4}$ | $8.08 \times 10^{-4}$ |
| GITA vs. QRL | 6.5 | $6.31 \times 10^{-5}$ | $3.79 \times 10^{-4}$ |
| GITA vs. CRL | 17.0 | $2.32 \times 10^{-4}$ | $8.08 \times 10^{-4}$ |
| GITA vs. HIQL | 7.0 | $1.62 \times 10^{-4}$ | $8.08 \times 10^{-4}$ |
| GITA vs. OTA | 16.0 | $5.36 \times 10^{-4}$ | $8.08 \times 10^{-4}$ |

Table 7: **Two-sided paired Wilcoxon signed-rank tests comparing GITA with the variants in Table 2.** $p_{\mathrm{adj}}$ denotes the Holm–Bonferroni-adjusted $p$-value across the four comparisons.

| **Comparison** | $W$ | **Raw** $p$ | $p_{\mathrm{adj}}$ |
|---|---|---|---|
| GITA vs. Unconditioned Value | 10.0 | 0.0100 | 0.0300 |
| GITA vs. Highest Advantage Factor | 5.0 | 0.00537 | 0.0215 |
| GITA vs. Mean Advantage Before Exponentiation | 30.5 | 0.873 | 0.873 |
| GITA vs. Geometric $k$ | 21.5 | 0.186 | 0.371 |

# F Statistical Testing

We assess differences between GITA and each baseline using a two-sided paired Wilcoxon signed-rank test. Each task constitutes one paired observation, using the mean success rate reported in the corresponding results table; thus, the main comparison uses 23 tasks and the ablation comparison uses 13 tasks. Because multiple baselines are compared against GITA, we correct the resulting $p$-values using the Holm–Bonferroni step-down procedure. Corrections are applied separately within each family of comparisons: the seven baselines in the main results and the four variants in the ablation study. We consider a difference statistically significant when the Holm-adjusted $p$-value is below $0.05$.

All comparisons remain significant after correction, supporting the aggregate improvement of GITA over each baseline in the main evaluation. And GITA significantly improves over removing scale conditioning and hard scale selection, while the two alternative aggregation/sampling variants are not significantly different after correction.

# G Baselines

This section provides additional details on the baseline methods considered in our experiments.

## G.1 OGBench Baselines

We compare against the six offline GCRL algorithms benchmarked in OGBench (Park et al., 2025). Goal-conditioned behavioral cloning (GCBC; Ghosh et al., 2021) trains a goal-conditioned policy by relabeling transitions with future states from the same trajectory as goals. Goal-conditioned implicit value learning and goal-conditioned implicit Q-learning (GCIVL and GCIQL; Kostrikov et al., 2022; Park et al., 2023) are goal-conditioned variants of Implicit Q-Learning (IQL). Both use expectile regression to estimate optimal goal-conditioned value functions. GCIQL learns both a state-value function $V(s, g)$ and an action-value function $Q(s, a, g)$, whereas GCIVL is the value-only variant introduced with HIQL and learns only $V(s, g)$. Quasimetric reinforcement learning (QRL; Wang et al., 2023) models goal-reaching distance with a learned quasimetric and exploits its triangle-inequality structure to learn goal-conditioned values. Contrastive reinforcement learning (CRL; Eysenbach et al., 2022) estimates a Monte Carlo goal-conditioned value function through contrastive

learning and performs one-step policy improvement. Hierarchical implicit Q-learning (HIQL; Park et al., 2023) extracts a two-level hierarchical policy from a single GCIVL value function. Its high-level policy predicts a latent representation of an intermediate $m$-step subgoal, while its low-level policy predicts primitive actions conditioned on that subgoal representation.

We additionally compare against option-aware temporally abstracted value learning (OTA; Ahn et al., 2025), which directly builds on HIQL and is the closest fixed-scale baseline to GITA. OTA replaces primitive one-step backups in the high-level value objective with temporally extended transitions using a fixed abstraction factor $k$, thereby shortening the effective value-propagation horizon. Unlike GITA, however, OTA uses a single abstraction factor for all state-goal pairs within a task and therefore requires task-specific selection of $k$. We use the task-specific abstraction factors and hyperparameters reported for OTA while otherwise following the corresponding HIQL/OGBench implementation.

## H Additional Ablation Details

We provide additional details on the ablations introduced in Section 5.2. Each variant changes a single component of GITA while leaving the remaining architecture, optimization procedure, goal sampling, and low-level policy unchanged. Unless stated otherwise, the abstraction-conditioned high-level value function is evaluated over the same candidate set $\mathcal{K}$ used by GITA. Recall that, for a training tuple $(s_t, s_{t+m}, g)$, GITA computes

$$A^h(s_t, s_{t+m}, g, k) = V_\theta^h(s_{t+m}, g, k) - V_\theta^h(s_t, g, k) \tag{48}$$

for every $k \in \mathcal{K}$ and uses the aggregated AWR weight

$$W_{\beta_h} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \exp\left(\beta_h A^h(s_t, s_{t+m}, g, k)\right) \tag{49}$$

to train the shared high-level policy. The following ablations separately modify the value conditioning, the scale aggregation rule, or the candidate abstraction factors.

**Unconditioned Value.** This variant removes the abstraction factor from the high-level value function, replacing $V_\theta^h(s, g, k)$ with a single unconditioned value function $V_\theta^h(s, g)$. Value training still uses temporally abstract transitions spanning different numbers of primitive steps, but the realized abstraction factor is not provided to the value network. The resulting value function must therefore fit supervision from multiple temporal scales within the same mapping. Since scale-specific values are no longer available, the high-level policy is trained from the single advantage

$$A^h(s_t, s_{t+m}, g) = V_\theta^h(s_{t+m}, g) - V_\theta^h(s_t, g), \tag{50}$$

using the standard AWR weight

$$\exp\left(\beta_h A^h(s_t, s_{t+m}, g)\right). \tag{51}$$

This ablation isolates whether training on multiple temporal scales is sufficient by itself, or whether explicitly representing the abstraction factor in the value function is necessary.

**Highest Advantage Factor.** This variant retains the abstraction-conditioned value function and computes $A^h(s_t, s_{t+m}, g, k)$ for every $k \in \mathcal{K}$ as in GITA, but replaces soft aggregation across scales with hard selection. For each training sample, we select

$$k^\star = \arg\max_{k \in \mathcal{K}} A^h(s_t, s_{t+m}, g, k) \tag{52}$$

and train the high-level policy using only

$$W_{\beta_h}^{\max} = \exp\left(\beta_h A^h(s_t, s_{t+m}, g, k^\star)\right). \tag{53}$$

All other factors are discarded for that sample. This ablation tests whether the multi-scale policy update benefits from combining supervision across several temporal resolutions, or whether selecting only the scale with the largest advantage is sufficient.

**Aggregate Advantage Before Exp.** This variant preserves all scale-specific advantages but changes the order of aggregation and the nonlinear AWR transformation. GITA first exponentiates each advantage and then averages the resulting weights,

$$W_{\beta_h} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \exp\left(\beta_h A^h(s_t, s_{t+m}, g, k)\right). \tag{54}$$

The ablation instead first averages the raw advantages,

$$\bar{A}^h(s_t, s_{t+m}, g) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} A^h(s_t, s_{t+m}, g, k), \tag{55}$$

and then applies the AWR exponential,

$$W^{\text{pre}}_{\beta_h} = \exp\left(\beta_h \bar{A}^h(s_t, s_{t+m}, g)\right). \tag{56}$$

This removes the scale-specific amplification induced by applying the exponential before aggregation. The comparison therefore isolates whether preserving differences in the magnitude of the individual scale-conditioned advantages through the nonlinear AWR weighting is important for policy supervision.

**GITA with Geometric $k$.** The default GITA policy update evaluates each training sample on a fixed candidate set $\mathcal{K}$. This ablation replaces that set with freshly sampled abstraction factors at every high-level policy update. We sample the same number $K = |\mathcal{K}|$ of factors independently according to

$$k_j \sim \text{Geom}(1-\zeta), \qquad p(k_j) = (1-\zeta)\zeta^{k_j - 1}, \qquad j = 1, \ldots, K, \tag{57}$$

with $\zeta = 0.9$, corresponding to a mean abstraction factor of 10. The parameter $\zeta$ is specific to this ablation and is distinct from the value-learning parameter $\alpha = \gamma$. The sampled set

$$\mathcal{K}_{\text{geo}} = \{k_1, \ldots, k_K\} \tag{58}$$

then replaces the fixed candidate set in the standard GITA aggregation,

$$W^{\text{geo}}_{\beta_h} = \frac{1}{K} \sum_{k \in \mathcal{K}_{\text{geo}}} \exp\left(\beta_h A^h(s_t, s_{t+m}, g; k)\right). \tag{59}$$

This change affects only the candidate factors used to construct the high-level policy supervision; the abstraction-conditioned value-learning procedure, including its sampling distribution parameterized by $\alpha$, is unchanged. The ablation tests whether GITA depends on a manually specified finite set of candidate scales or can obtain similar behavior from stochastic coverage of the temporal-scale space.

# I Further Results and Visualizations

## I.1 Scale-Dependent Policy Weights Across State–Goal Distances

To further visualize how different temporal scales contribute to high-level policy learning, we examine the scale-specific AWR weights as a function of state–goal distance. For each abstraction factor $k$, we compute

$$w_k = \min\left\{\exp\left(\beta_h A^h(s_t, s_{t+m}, g; k)\right), 100\right\}, \tag{60}$$

where $A^h(s_t, s_{t+m}, g; k)$ is the abstraction-conditioned high-level advantage from Equation (11). The clipping threshold of 100 is also applied during training, so the plotted quantity corresponds to the final AWR weight used in the high-level policy update, up to the particular set of $k$ values shown. This clipping convention is inherited from the official HIQL and OTA implementations. We group samples according to their geodesic state–goal distance and report the mean of $w_k$ within each distance bin for each abstraction factor. Curves are averaged across seeds, with shaded regions indicating one standard deviation.

Recall from Equation (12) that GITA averages the exponentiated advantages across $k$. Thus, scales assigning larger positive advantages to a training tuple contribute disproportionately to its policy

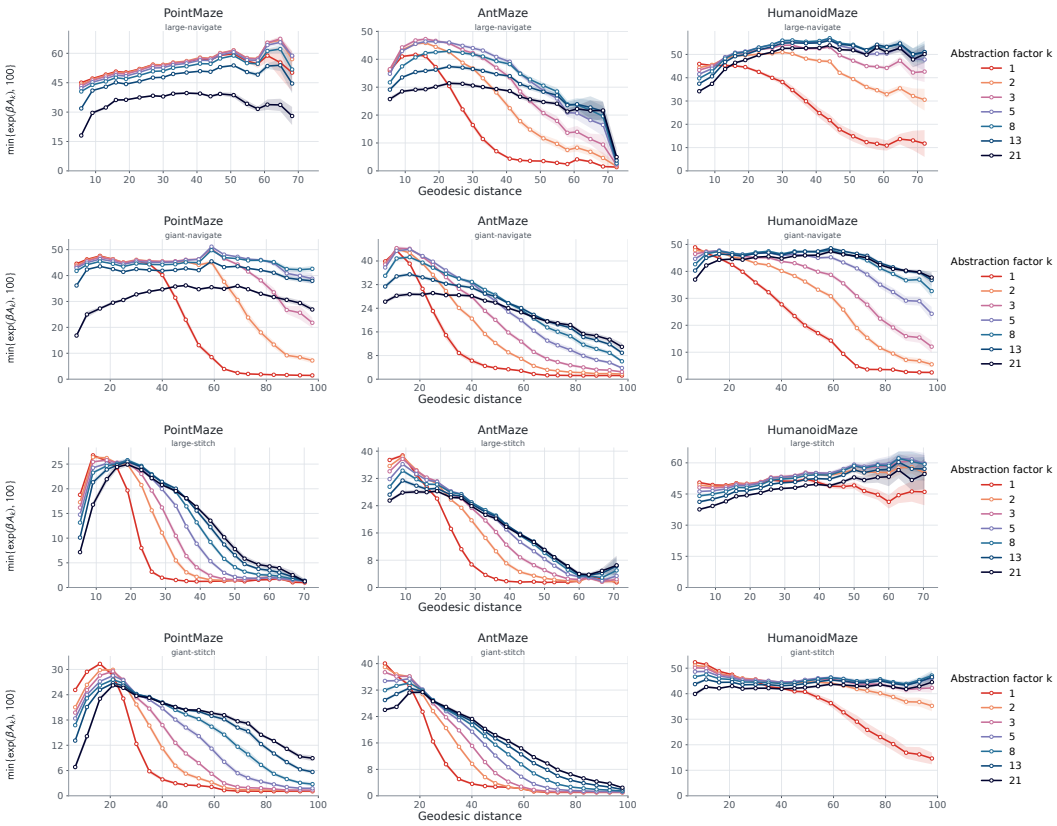


Figure 7: **Scale-specific AWR weights across navigation and stitching tasks.** Mean clipped exponentiated high-level advantage, $\min\{\exp(\beta_h A^h), 100\}$, as a function of geodesic state–goal distance for different abstraction factors $k$. From top to bottom: `large-navigate`, `giant-navigate`, `large-stitch`, and `giant-stitch`. Curves show the mean across seeds and shaded regions indicate one standard deviation. Each plot contains the corresponding maze morphologies.

weight, without requiring an explicit scale-selection rule. Figure 7 shows this behavior across PointMaze, AntMaze, and HumanoidMaze for the `navigate` and `stitch` datasets and the `large` and `giant` maze layouts.

Across the maze environments, the scale-specific AWR weights exhibit a clear dependence on state–goal distance. Smaller abstraction factors lose their advantage signal more rapidly as distance increases. Larger factors retain informative weights over longer ranges, with the transition occurring progressively later as $k$ increases. This pattern is especially clear in the `giant` layouts and is consistent with the reachability–resolution trade-off predicted by our analysis.

### I.2 Results Across Discount Factors

To characterize the sensitivity of GITA to the discount factor, we report performance for each value considered in our sweep, $\gamma \in \{0.98, 0.99, 0.995\}$, on the maze tasks in Table 8. Unlike the results reported in the main evaluation, this table uses only the final checkpoint after $10^6$ gradient steps. Results are reported as mean $\pm$ standard deviation over 8 seeds.

Because we set $\alpha = \gamma$, this sweep changes both the discount factor and the distribution of abstraction factors. In particular, $\gamma = \alpha = 0.995$ gives $\mathbb{E}[k] = 200$, which matches the data episode length of PointMaze and AntMaze `stitch`. As a result, many sampled abstraction factors are truncated by trajectory boundaries, so part of the performance drop at larger $\gamma$ may come from the induced geometric distribution rather than from discounting alone. This interaction further justifies re-tuning $\gamma$ for GITA.

Table 8: **Success rate (%) of GITA for each discount factor considered in our sweep.** Results use only the final checkpoint after $10^6$ gradient steps and are reported as mean $\pm$ standard deviation over 8 seeds. The *Best $\gamma$ per Task* column reports the best task-level result together with the corresponding discount factor in parentheses. The best global discount factor is $\gamma = 0.99$, selected by the average performance across all tasks; the last column reports the corresponding task-level performance.

| Environment | Type | Size | $\gamma = 0.98$ | $\gamma = 0.99$ | $\gamma = 0.995$ | Best $\gamma$ per Task | Best Global $\gamma = 0.99$ |
|---|---|---|---|---|---|---|---|
| AntMaze | navigate | large | 91.0±1.6 | 89.0±1.9 | 0.7±0.8 | 91.0±1.6 (0.98) | 89.0±1.9 |
| | navigate | giant | 78.8±2.7 | 65.8±2.6 | 2.2±3.0 | 78.8±2.7 (0.98) | 65.8±2.6 |
| | stitch | large | 85.6±2.9 | 60.2±27.8 | 0.1±0.4 | 85.6±2.9 (0.98) | 60.2±27.8 |
| | stitch | giant | 41.5±5.8 | 16.6±13.2 | 0.0±0.0 | 41.5±5.8 (0.98) | 16.6±13.2 |
| | explore | large | 65.9±12.5 | 87.0±8.8 | 50.8±21.9 | 87.0±8.8 (0.99) | 87.0±8.8 |
| HumanoidMaze | navigate | large | 50.8±7.1 | 72.2±4.6 | 68.2±2.8 | 72.2±4.6 (0.99) | 72.2±4.6 |
| | navigate | giant | 54.1±7.1 | 89.6±4.3 | 83.2±3.4 | 89.6±4.3 (0.99) | 89.6±4.3 |
| | stitch | large | 28.2±4.2 | 47.8±4.5 | 49.0±2.3 | 49.0±2.3 (0.995) | 47.8±4.5 |
| | stitch | giant | 10.0±1.3 | 61.2±6.5 | 67.5±2.9 | 67.5±2.9 (0.995) | 61.2±6.5 |
| PointMaze | navigate | large | 88.0±10.4 | 92.0±8.6 | 92.0±5.9 | 92.0±5.9 (0.995) | 92.0±8.6 |
| | navigate | giant | 74.9±12.7 | 82.4±4.7 | 81.5±7.7 | 82.4±4.7 (0.99) | 82.4±4.7 |
| | stitch | large | 72.7±16.6 | 61.5±17.0 | 20.6±8.5 | 72.7±16.6 (0.98) | 61.5±17.0 |
| | stitch | giant | 10.3±8.0 | 56.2±12.6 | 34.4±11.6 | 56.2±12.6 (0.99) | 56.2±12.6 |