ADAPTIVE VALUE ANCHORING FOR LONG-HORIZON STITCHING IN OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

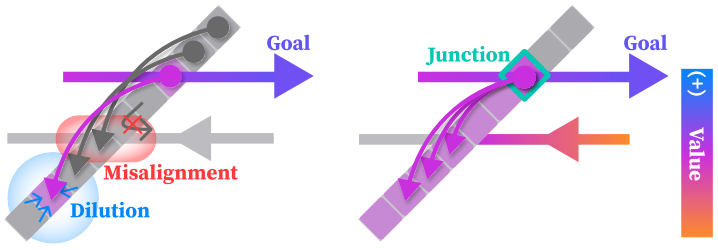
ABSTRACT

An important capability of offline goal-conditioned reinforcement learning is stitching, which can improve policies by connecting suboptimal trajectory segments and enable agents to solve long-horizon tasks by combining shorter trajectories. Stitching commonly relies on temporal difference (TD) learning, whose repeated bootstrapping propagates reward signals between trajectories but can also lead to bias accumulation in value estimation. Although longer backups can mitigate bias accumulation by reducing bootstrapping depth, we show that they can hinder stitching. Specifically, longer backups can skip junction states where reward signals propagate between trajectories. Therefore, we propose Adaptive Value Anchoring (AVA), which adapts backup lengths to bootstrap from junction states. AVA can thereby mitigate bias accumulation while preserving reward propagation between trajectories. We evaluate AVA on long-horizon OGBench tasks that require stitching. AVA achieves the highest average success rate among the compared methods. Moreover, it achieves the highest mean success rate in every subsampled setting, where reduced trajectory overlap makes stitching harder.

1 INTRODUCTION

Offline goal-conditioned reinforcement learning (GCRL) (Levine et al., 2020; Liu et al., 2022) trains a policy for reaching diverse goals using an offline dataset without additional environment interaction. Because GCRL allows the use of an unsupervised reward formulation, RL can be scaled to large, unlabeled datasets (Park et al., 2025a). An important challenge in learning from offline datasets is enabling trajectory stitching. Stitching can not only improve policies by connecting suboptimal trajectories (Kumar et al., 2022; Ghugare et al., 2024) but also enable more diverse goal-reaching behaviors by combining optimal trajectories (Park et al., 2025a).

Stitching in offline GCRL commonly relies on temporal difference (TD) learning (Sutton, 1988; Sutton & Barto, 2018). In single-step TD learning, each backup updates the value estimate at the current state using the immediate reward and the discounted next-state value (Bellman, 1954). In this way, repeated backups propagate sparse goal-conditioned reward signals across the state space. However, each backup inherits estimation error from the bootstrap value (Kearns & Singh, 2000), and repeated backups can accumulate estimation bias (Park et al., 2025b). Such bias accumulation can limit performance on long-horizon tasks and prevent further improvement as the dataset grows (Park et al., 2023; 2025b). To mitigate bias accumulation, horizon reduction techniques (Park



(a) Fixed horizon reduction **(b) Adaptive Value Anchoring**

Figure 1: (a) Fixed-horizon reduction can hinder stitching by skipping junction states. (b) We propose *Adaptive Value Anchoring* to preserve stitching under horizon reduction by anchoring TD backups to junction states.

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.

et al., 2025b; Ahn et al., 2025; Li et al., 2025) have been proposed. Horizon reduction methods use longer backups to reduce backup depth and potentially mitigate bias accumulation.

However, in this paper, we show that horizon reduction can hinder the stitching ability of offline GCRL. Stitching relies on bootstrapping from the value of a *junction state* shared by multiple trajectories. Bootstrapping from junction states allows the reward signal to propagate between trajectories. However, a fixed-horizon backup bootstraps only from the value at the segment endpoint. Therefore, the reward signal from another trajectory propagates directly only to states whose backups end at a junction (Figure 1a). The value increases induced by the sparsely propagated reward signal can be diluted by lower values at nearby states (dilution). Moreover, propagation to the next trajectory can fail when the next junction is misaligned with the states that receive the value increase (misalignment). In this way, fixed-horizon reduction can hinder stitching.

A natural solution is to end each backup at a junction state and bootstrap from the junction value. We therefore propose Adaptive Value Anchoring (AVA), which adapts backup lengths to anchor backups to junction states (Figure 1b). However, because each backup ends upon reaching a junction, designating too many states as junctions makes AVA approach single-step TD. We propose a junction score that measures how useful each state is as a junction. We evaluate AVA on long-horizon locomotion and manipulation tasks from OGBench (Park et al., 2025a). We also evaluate AVA on LIBERO manipulation tasks (Liu et al., 2023). AVA achieves the highest average success rate among the compared methods on both benchmarks and in every subsampled OGBench setting.

2 Related Work

Offline Goal-Conditioned RL. Offline GCRL combines offline RL (Levine et al., 2020) with goal-conditioned RL (Schaul et al., 2015; Andrychowicz et al., 2017; Liu et al., 2022). One line of work formulates goal reaching as supervised learning by imitating actions from offline datasets (Ghosh et al., 2021; Emmons et al., 2022; Brandfonbrener et al., 2022; Lynch et al., 2020). However, such methods typically have limited stitching ability (Ghugare et al., 2024; Zhou et al., 2024). TD-based methods (Kostrikov et al., 2022; Park et al., 2023; 2024b; Kang et al., 2026) instead enable trajectory stitching through dynamic programming (Ghugare et al., 2024). However, TD-based methods can accumulate estimation bias over long horizons (Park et al., 2025b). On the other hand, there are TD-free methods, including quasimetric learning (Wang et al., 2023; Myers et al., 2025), contrastive learning (Eysenbach et al., 2022; Liu et al., 2025; Ma et al., 2023), and transitive RL (Park et al., 2026). While such methods can also perform stitching, TD-based methods have been observed to achieve better stitching performance than TD-free methods (Bortkiewicz et al., 2026; Sobal et al., 2025). Moreover, scaling TD-free methods to long-horizon tasks remains an open challenge (Park et al., 2025a) (see the locomotion results in Table 1).

Horizon Reduction in RL. Several approaches (Dayan & Hinton, 1992; Levy et al., 2019; Park et al., 2025b; Ahn et al., 2025; Li et al., 2025; 2026; Jawaid, 2026) reduce backup depth using n-step targets or hierarchical structure, typically with fixed horizons or abstraction levels. However, fixed-horizon backups can hinder stitching. Several methods instead adapt the backup length to address the limitations of fixed horizons (Shin et al., 2026; Chen et al., 2026; Zheng et al., 2026). In particular, concurrent work by Zheng et al. (2026) shows in a tabular example that fixed horizons can prevent optimal stitching. The failure occurs when subsequent negative rewards suppress positive reward signals on the same trajectory. We show that stitching can fail under function approximation even when all non-goal states have the same reward. Specifically, unlike Zheng et al. (2026), we analyze dilution under function approximation and misalignment across successive stitches. Moreover, such methods (Shin et al., 2026; Chen et al., 2026; Zheng et al., 2026) select backup lengths by maximizing learned value estimates. However, maximizing inaccurate value estimates can favor overestimated states (Thrun & Schwartz, 1993; Hasselt, 2010). AVA instead selects bootstrap endpoints using precomputed junction scores.

Trajectory Stitching Algorithms. One line of research augments offline datasets with stitched trajectories (Ghugare et al., 2024; Char et al., 2021; Hepburn & Montana, 2022; Zhou et al., 2024; Yuan et al., 2025; Lee & Choi, 2025). Another approach generates stitched trajectories at decision time with generative planners (Wang et al., 2024; Luo et al., 2025; Kim et al., 2024). AVA instead focuses on value learning in offline GCRL and can also be used with augmented datasets.

3 BACKGROUND

Offline Goal-Conditioned RL. We consider offline GCRL in a Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \mu, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces. The transition dynamics is $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and $\mu \in \Delta(\mathcal{S})$ is the initial state distribution.[1] For a goal $g \in \mathcal{S}$, the sparse reward $r(s, g)$ is 0 when the goal is reached and -1 otherwise. The discount factor is $\gamma \in [0, 1)$, and the effective horizon is $1/(1-\gamma)$. Offline GCRL aims to learn a goal-conditioned policy $\pi(a|s, g)$ from a fixed dataset $\mathcal{D}$ to maximize the expected return $\mathbb{E}_\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, g)\big]$. Each trajectory in $\mathcal{D}$ is collected by a behavior policy π_β, with $a_t \sim \pi_\beta(\cdot|s_t)$ and $s_{t+1} \sim p(\cdot|s_t, a_t)$.

Implicit Q-Learning. A goal-conditioned Q-function estimates the expected return given a state, action, and goal. Standard Q-learning updates the Q-function using the maximum Q-value over actions at the next state (Watkins & Dayan, 1992). However, in offline RL, maximizing over actions can involve evaluating out-of-sample actions with overestimated values (Levine et al., 2020). Implicit Q-Learning (IQL) (Kostrikov et al., 2022) avoids evaluating out-of-sample actions by approximating in-sample maximization using expectile regression. The expectile loss is defined as follows: $L_2^\tau(x) = |\tau - \mathbb{I}\{x < 0\}|x^2$, where x is the regression residual. For $\tau \in (0.5, 1)$, the loss assigns greater weight to positive residuals. As τ approaches 1, the fitted expectile approaches the maximum over in-sample values. Under deterministic dynamics, the value loss of action-free IQL (Ghosh et al., 2023; Xu et al., 2022; Park et al., 2023) is defined as follows:

$$\mathcal{L}_V(\phi) = \mathbb{E}_{(s,s',g)\sim\mathcal{D}}\big[(L_2^\tau(r(s, g) + \gamma V_{\hat{\phi}}(s', g) - V_\phi(s, g)))\big], \tag{1}$$

where $V_\phi(s, g)$ is a goal-conditioned value function, $V_{\hat{\phi}}$ is the target network (Mnih et al., 2015), and the backup target is $y = r(s, g) + \gamma V_{\hat{\phi}}(s', g)$. An n-step backup replaces the single-step target with $\sum_{i=0}^{n-1} \gamma^i r(s_{t+i}, g) + \gamma^n V_{\hat{\phi}}(s_{t+n}, g)$. The learned values can guide flat (Peng et al., 2019; Nair et al., 2020; Fujimoto & Gu, 2021) or hierarchical policy extraction (Park et al., 2023).

4 HORIZON REDUCTION CAN HINDER STITCHING

Horizon reduction alleviates bias accumulation but can hinder stitching. We first analyze how horizon reduction can hinder stitching through dilution and misalignment (Section 4.1). We then illustrate the analysis with a didactic example (Section 4.2).

4.1 DILUTION AND MISALIGNMENT

In a continuous state space, two trajectories rarely share the same state. Therefore, stitching relies on value function generalization between nearby states on different trajectories (Bortkiewicz et al., 2026). Consider two nearby states from different trajectories. Suppose the reward signal has increased state values along the first trajectory but has not yet propagated along the second trajectory. Value function generalization can transfer the value increase from the first trajectory to the nearby state on the second trajectory. Then, earlier states on the second trajectory can receive the reward signal by bootstrapping from the increased value.

Let s_t denote the state on the second trajectory whose value has increased. For simplicity, assume that the initial increase is confined to s_t. Under a fixed n-step backup, only s_{t-n} among the earlier states $s_{t-1}, \dots, s_{t-n}$ on the second trajectory bootstraps from s_t. Therefore, only s_{t-n} receives a higher backup target from the increased value. In that case, stitching can be hindered in two ways.

The first problem is *dilution*. Because the value function tends to assign similar values to nearby states, competition with lower targets can dilute the value increase at s_{t-n}. A potential objection is that expectile regression should prevent dilution by giving greater weight to higher targets.

Lemma 1. *Suppose a set of nearby states is fitted to a single value. A fraction $\alpha \in (0, 1)$ of the backup targets equal y^+, and the rest equal y^-, with $y^+ > y^-$. Under the expectile loss of Equation 1 with $\tau \in (0, 1)$, the fitted value is $v^\star = y^- + \rho(y^+ - y^-)$, where $\rho = \frac{\tau\alpha}{\tau\alpha+(1-\tau)(1-\alpha)}$ is the recovery fraction of the target gap. For a desired recovery fraction $q \in (0, 1)$, $\rho \geq q$ holds if and only if $\tau \geq \frac{q(1-\alpha)}{(1-q)\alpha+q(1-\alpha)}$.*

[1] $\Delta(\mathcal{X})$ denotes the set of probability distributions on $\mathcal{X}$.

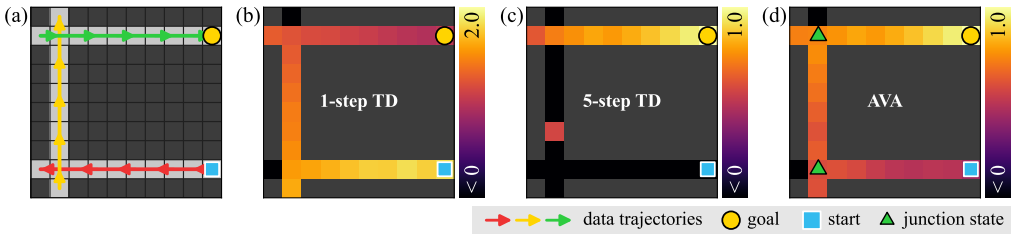


Figure 2: **Didactic example.** (a) Three dataset trajectories require two stitches to reach the goal. Value estimates from (b) single-step TD, (c) 5-step TD, and (d) AVA with known junctions (green triangles).

Proof. See Appendix A.1.

Lemma 1 shows that maintaining the same recovery fraction at smaller α requires a larger τ. However, a larger τ can make regression harder to optimize (Kostrikov et al., 2022).

The second problem is *misalignment* across successive stitches. After the first stitch, repeated n-step backups directly propagate the junction value increase along the trajectory only to $s_{t-n}, s_{t-2n}, \ldots$. Expectile regression reweights residuals but does not change the bootstrap endpoints. Therefore, states that do not directly receive the value increase can retain low values even without dilution. In this case, junction states needed for subsequent stitches can be misaligned with the states whose values have increased. Such misalignment can prevent the reward signal from propagating to the next trajectory. A larger n makes the propagation positions sparser and increases the risk of misalignment. Moreover, each subsequent stitch depends on signal propagation through earlier junctions. Therefore, misalignment compounds in longer tasks that require more successive stitches.

The initial value increase was assumed to be confined to s_t, but generalization can raise nearby state values and alleviate sparse propagation in practice. However, the values of nearby states are influenced by lower targets as well as the increased junction value. Therefore, dilution and misalignment may be less severe than in the simplified setting but can still hinder stitching.

4.2 Didactic Example

We illustrate the analysis using the 10×10 gridworld in Figure 2a.[2] The agent moves one cell horizontally or vertically at each step. The dataset consists of three trajectories, and none of them reaches the goal (yellow circle) from the initial state (blue square). Reaching it therefore requires two stitches. This setup resembles the `stitch` datasets of OGBench (Park et al., 2025a). We set $\gamma = 0.95$, which corresponds to a 20-step effective horizon. To emulate the error in value estimation, we scale each estimate by a factor drawn uniformly from $[0.9, 1.1]$. We use in-sample maximization instead of expectile regression because the tabular setting allows us to compute the maximum directly. Therefore, the example isolates the effect of misalignment without dilution.

In this example, single-step TD propagates values across all three trajectories (Figure 2b). However, repeated backups can accumulate overestimation from maximization over noisy values and even reverse the estimated value ordering. Therefore, a greedy policy would move away from the goal. The 5-step update reduces accumulated error but propagates the stitched value only to every fifth state (Figure 2c). Therefore, misalignment prevents the stitched value from reaching the second junction. The failure illustrates that misalignment can arise without dilution.

The didactic example raises the following question. *Can we benefit from horizon reduction while preserving stitching ability?* When junctions are known, a natural solution is to end each backup at a junction within the backup horizon. Therefore, we propose AVA, which anchors backups to junction states. In Figure 2d, AVA completes both stitches with less error than single-step TD. The remaining challenge is to identify junctions in continuous state spaces.

5 Adaptive Value Anchoring

As discussed in Section 4, anchoring backups to junction states can restore stitching under horizon reduction. The remaining challenge is to identify useful junction states.

[2]For visual clarity, the didactic example uses rewards of $+1$ at the goal and 0 otherwise.

5.1 JUNCTION SELECTION

If we designate all states as junctions, AVA approaches single-step TD because each backup ends upon reaching a junction. Therefore, junctions should be assigned selectively. Anchoring at a junction can stitch trajectories that overlap at the junction. Additionally, a junction that enables broader state coverage through stitching can support more tasks because every state is a potential goal in GCRL. Therefore, we assign each state a *junction score* based on the state coverage of trajectories that would be stitched at that state.

Feature Space for Junction Scoring. Identifying trajectory overlaps and measuring state coverage require a notion of distance. However, raw states can include dimensions less informative about proximity between states, such as oscillating joint positions in locomotion tasks. Therefore, we project raw states into a feature space. Notably, the feature space is used only for junction scoring, and the value and policy networks receive the raw state.

By default, AVA uses Euclidean distances between selected state coordinates. We select the 2D root position for locomotion tasks and the 3D gripper position for manipulation tasks. Notably, junction selection aims to avoid missing states needed for reward propagation between trajectories. Therefore, we consider gripper proximity sufficient for identifying potential junctions because gripper proximity is a prerequisite for proximity in the full state space.

To avoid manually choosing state coordinates for junction scoring, we also consider a feature space based on learned temporal distance. We use the latent space learned by HILP (Park et al., 2024b) to approximate temporal distances[3] for junction scoring. We call this variant **AVA w/ HILP**. Notably, learning temporal distance is essentially equivalent to value learning because temporal distances from states to goals directly determine the corresponding values under sparse goal-conditioned rewards (Park et al., 2024b; Myers et al., 2024). Therefore, temporal distance learning can face the same value learning challenges addressed in this work. However, our junction scoring uses learned temporal distances only to identify local overlaps and measure coverage at the scale of a single trajectory. Therefore, our junction scores can be less sensitive to errors in long-horizon distance estimates. Notably, AVA w/ HILP performs comparably to AVA (Table 1).

Junction Scoring. The state coverage of overlapping trajectories depends only on the offline dataset and can be precomputed before training. However, a naive search for overlapping trajectories requires comparing all pairs of states in the dataset, and quantifying coverage in continuous state spaces is not straightforward. Therefore, we discretize the feature space for junction scoring into a grid of cells. In this way, we can identify overlapping trajectories by looking up states in the same cell without comparing all pairs of states. Additionally, we can quantify coverage simply by counting distinct cells visited by overlapping trajectories.

The junction score of s counts additional cells covered by downstream segments of trajectories overlapping at s. We exclude cells already covered by the trajectory containing s because stitching adds no coverage in those cells. Only downstream segments contribute because stitching provides access to subsequent states along the overlapping trajectories, not to preceding states. Let $\mathcal{M}(s)$ denote the states in the same cell as s and $\mathcal{C}(\tau)$ the cells visited by the trajectory τ. Let τ_s be the trajectory containing s and $\tau_{>m}$ the downstream segment after a state $m \in \mathcal{M}(s)$ on the trajectory containing m. Then, the junction score is computed as follows:

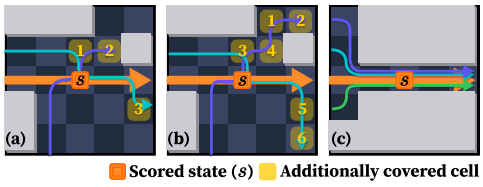


Figure 3: (a) Three additional cells give a junction score of 3. (b) The junction score is 6. (c) No additional coverage gives a junction score of 0.

$$\text{score}(s) = \left| \bigcup_{m \in \mathcal{M}(s)} \mathcal{C}(\tau_{>m}) \setminus \mathcal{C}(\tau_s) \right|. \tag{2}$$

We precompute the scores once and designate the top fraction η of dataset states as junctions. Without discretization, searching the entire dataset for each of the N states costs $O(N^2)$ overall. Discretization replaces each search with an expected $O(1)$ hash lookup.

Figure 3 illustrates three junction scoring examples. The orange rounded square marks the state s being scored, the orange arrow represents τ_s, and the remaining arrows represent trajectories over-

[3]Temporal distance is the minimum number of steps needed to reach one state from another.

lapping at s. Cells contributing to the junction score are highlighted in yellow and numbered. In Figure 3b, the overlapping trajectories cover more cells than in Figure 3a, so s receives a higher score. Additionally, high trajectory density does not necessarily imply additional coverage. For example, in Figure 3c, the cell containing s has the highest trajectory density. However, the overlapping trajectories add no coverage beyond τ_s, so the junction score is zero. We evaluate the number of states per cell as an alternative scoring criterion (Table 2 and Figure 7).

5.2 Implementation of Adaptive Value Anchoring

Value Learning. We compute the adapted horizon k as follows: $k = \min(\{i \in \{1, \dots, n\} : s_{t+i} \in \mathcal{J}\} \cup \{n\})$, where $\mathcal{J}$ contains the selected junction states and n is the maximum horizon length. We then replace the single-step target in Equation 1 with a k-step target as follows:

$$\mathcal{L}_V(\phi) = \mathbb{E}_{(s_t,g)\sim\mathcal{D}}\Big[L_2^\tau\Big(\sum_{i=0}^{k-1}\gamma^i r(s_{t+i},g) + \gamma^k V_{\hat{\phi}}(s_{t+k},g) - V_\phi(s_t,g)\Big)\Big]. \tag{3}$$

If the goal is reached within the n-step window, we end the backup at the goal and omit the bootstrap term. We discuss off-policy bias in multi-step value learning in Appendix C.1.

Policy Extraction. Because we focus on long-horizon tasks, we follow the hierarchical policy structure of HIQL (Park et al., 2023) to reduce the horizon of policy learning. The high-level policy predicts a subgoal, and the low-level policy selects an action given the subgoal. The subgoal horizon is fixed independently of the adapted backup horizon k. As in HIQL, we train the high-level policy with AWR (Peng et al., 2019). AWR assigns greater weight to subgoals with larger estimated value gains relative to the current state. However, we train the low-level policy with DDPG+BC (Fujimoto & Gu, 2021) instead of the AWR used in HIQL. Unlike AWR, which reweights dataset actions, DDPG+BC maximizes the normalized Q value through action gradients. Such action gradients allow extrapolation beyond dataset actions, while behavioral cloning discourages large deviations from the behavior distribution (Park et al., 2024a). Reaching a stitched subgoal on another trajectory may require some deviation from the behavior distribution. To obtain action gradients, we additionally train a single-step Q function that bootstraps from the value function learned by AVA.

However, the hierarchical policy structure of HIQL can be less effective on state-based manipulation tasks (Park et al., 2025a), as discussed in Appendix C.3. Therefore, we use a flat DDPG+BC policy for manipulation tasks to prevent policy extraction failures from obscuring improvements in value learning. Notably, policy extraction is matched across methods within each ablation comparison. Moreover, AVA achieves higher success than a variant with a fixed backup horizon when both methods use the same policy extraction setup as HIQL (Appendix D.2).

6 Experiment Setup

We evaluate the effectiveness of AVA on long-horizon tasks that require stitching. We also examine the contributions of the adaptive horizon and junction score.

Tasks and Datasets. We evaluate AVA on OGBench (Park et al., 2025a). For locomotion, we use `antmaze-giant` and `humanoidmaze-giant`, with maximum episode lengths of 1000 and 4000 steps, respectively. The corresponding `stitch` datasets have trajectory lengths of only 200 and 400 steps, respectively. Therefore, reaching distant goals requires stitching multiple trajectories. The methods evaluated by Park et al. (2025a) achieve success rates of at most 3% on these `stitch` datasets. We also subsample trajectories from the `stitch` datasets to evaluate stitching at lower data densities. For manipulation, we use the `puzzle` environments, where a robot arm presses buttons in an array to reach a goal configuration. Because the configuration space is large (2^{20} configurations for `puzzle-4×5`), reaching a goal can require stitching trajectory segments not observed together. We abbreviate locomotion datasets by the initials of the robot, maze size, and dataset type. For example, `antmaze-giant-stitch` becomes `AGS`, and `humanoidmaze-giant-navigate` becomes `HGN` . Puzzle datasets are abbreviated by size, such as `P33` for `puzzle-3×3`.

We also evaluate AVA on `KITCHEN-SCENE1` and `KITCHEN-SCENE3` from LIBERO-90 (Liu et al., 2023) to examine whether AVA remains effective when near-optimal trajectories are available.

The dataset for each scene includes trajectories for different tasks with the same set of objects. To focus on horizon reduction and stitching, we convert the dataset to use state-based observations. Notably, we conduct the LIBERO experiments without pretraining. We consider an episode successful when all task-specific goal conditions are satisfied simultaneously. Figure 5 shows examples of the OGBench and LIBERO environments used in our experiments.

Baselines and Ablations. We compare AVA with baselines in the following classes.

- **Single-step TD**: methods that learn values using single-step TD. We include IQL (Kostrikov et al., 2022), HIQL (Park et al., 2023), and LAVL (Kang et al., 2026)
- **Fixed-horizon TD**: methods that learn values with fixed-horizon backups. We include SHARSA (Park et al., 2025b), which uses n-step returns, and OTA (Ahn et al., 2025), which ends each option after a fixed number of steps.
- **Hybrid TD**: methods that combine single-step and multi-step backups. We include CGQ (Song et al., 2026), which regularizes a single-step critic toward a chunked critic, and the quasimetric method MQE (Zheng et al., 2025).
- **TD-free**: methods that learn values without temporal difference updates. We include CRL (Eysenbach et al., 2022), QRL (Wang et al., 2023), and TRL (Park et al., 2026).

For LIBERO, we additionally include goal-conditioned behavioral cloning (BC) as a baseline. We use the official implementations of all baselines.

We assess the adaptive horizon and junction score using four variants. ***n*-step VA** fixes the backup horizon at n. **Random VA** uniformly samples a bootstrap state within the horizon. **Density-based AVA** uses the number of states per cell as the junction score. **Value-based AVA** bootstraps from the state with the highest estimated value within the horizon, similar to prior adaptive-horizon methods (Shin et al., 2026; Chen et al., 2026; Zheng et al., 2026).

Hyperparameters. AVA uses a maximum horizon length of $n = 25$ and a top ratio of $\eta = 0.1$ across all tasks. We use cell sizes of 0.1 for locomotion, 0.02 for manipulation, and 5.0 for AVA w/ HILP. The cell sizes account for the different scales of the junction-scoring feature spaces. We set the expectile parameter τ to 0.7 for AVA and its variants, following the OGBench default for HIQL (Park et al., 2025a). Figure 4b, Table 12, and Appendix D.4 report sweeps of n, η, and the cell size, respectively. Appendix B describes the baseline hyperparameters and evaluation protocol.

7 Experimental Results

Evaluation on OGBench. AVA achieves the highest average success rate among the compared methods (Table 1). The average success rate of AVA is approximately 1.7 times that of LAVL, the strongest baseline. Moreover, AVA achieves higher mean success than n-step VA across all seven datasets. On `AGS`, AVA achieves approximately twice the mean success rate of n-step VA. The gains support the benefit of adapting backup horizons to junction states. Additionally, AVA w/ HILP consistently improves over n-step VA on all seven datasets. Therefore, the gains over n-step VA do not rely solely on manual coordinate selection. On `P44`, AVA achieves the highest mean success rate and exceeds CGQ by 16.3 percentage points. Although MQE and OTA perform similarly to

Table 1: **Results on long-horizon OGBench datasets.** We report the average and standard deviation of the success rate (%) over 5 seeds (see Appendix B.1). Following Park et al. (2025a), numbers at or above 95% of the best value in each row are highlighted in bold.

		Single-step TD			Fixed-horizon TD		Hybrid TD		TD-free			Ablations	Ours	
Task type	Dataset	IQL	HIQL	LAVL	SHARSA	OTA	CGQ	MQE	CRL	QRL	TRL	n-step VA	**AVA w/ HILP**	**AVA**
Locomotion	`AGS`	0.0 ±0.0	1.5 ±0.9	35.7 ±5.5	0.0 ±0.0	16.4 ±3.4	0.0 ±0.0	45.7 ±4.5	0.0 ±0.0	3.3 ±5.8	0.0 ±0.0	24.3 ±7.5	40.4 ±11.5	**48.7 ±5.8**
	`HGS`	0.1 ±0.2	7.6 ±1.8	56.1 ±7.5	0.2 ±0.2	**75.6 ±4.4**	0.0 ±0.0	34.2 ±19.8	0.1 ±0.2	0.2 ±0.2	0.2 ±0.1	69.5 ±4.6	**73.4 ±3.8**	**75.8 ±2.5**
	`AGN`	0.3 ±0.1	64.3 ±3.6	75.9 ±7.1	17.0 ±4.4	58.7 ±4.0	0.0 ±0.0	17.0 ±3.6	17.4 ±5.3	13.3 ±3.7	5.6 ±2.2	**88.8 ±1.6**	**90.2 ±2.0**	**90.3 ±1.2**
	`HGN`	0.5 ±0.5	23.9 ±16.2	84.1 ±2.0	1.4 ±0.3	82.0 ±1.9	0.0 ±0.0	59.8 ±2.9	7.5 ±1.3	0.8 ±0.3	2.5 ±1.2	90.1 ±2.5	**95.8 ±0.5**	**94.7 ±1.1**
Manipulation	`P33`	**94.6 ±2.7**	11.7 ±2.3	14.1 ±1.7	49.5 ±2.8	16.5 ±1.3	75.2 ±42.4	7.9 ±1.9	4.1 ±1.0	0.7 ±0.1	70.0 ±6.6	67.5 ±5.2	72.8 ±9.1	74.1 ±5.4
	`P44`	27.1 ±0.9	7.5 ±2.6	11.7 ±5.3	52.1 ±6.6	3.0 ±0.9	61.4 ±7.7	11.5 ±1.8	0.5 ±0.2	0.1 ±0.1	38.0 ±1.2	63.0 ±7.7	68.7 ±5.6	**77.7 ±2.8**
	`P45`	13.4 ±1.2	3.2 ±0.2	5.8 ±1.8	**18.5 ±1.3**	1.9 ±0.8	13.0 ±1.6	2.2 ±0.8	0.8 ±0.5	0.2 ±0.3	8.9 ±1.2	15.4 ±1.4	16.5 ±1.0	**18.5 ±0.8**
Average		19.4	17.1	40.5	19.8	36.3	21.4	25.5	4.3	2.6	17.9	59.8	**65.4**	**68.5**

Table 2: **Results on full and subsampled OGBench datasets.** Success rates (%) on the full `stitch` datasets and subsets retaining 50% or 25% of the trajectories. The reporting and boldface conventions follow Table 1.

		Baselines		Ablations				Ours	
Dataset	Fraction	OTA	MQE	n-step VA	Value-based AVA	Random VA	Density-based AVA	**AVA w/ HILP**	**AVA**
AGS	full	16.4 ±3.4	45.7 ±4.5	24.3 ±7.5	0.1 ±0.1	36.7 ±8.3	37.8 ±6.0	40.4 ±11.5	**48.7 ±5.8**
	50%	4.2 ±8.7	20.1 ±1.5	17.3 ±4.6	0.7 ±0.6	30.0 ±2.7	22.4 ±4.6	24.9 ±8.2	**38.1 ±8.5**
	25%	0.1 ±0.1	6.6 ±1.8	6.5 ±2.2	0.7 ±0.8	9.2 ±4.0	7.2 ±2.9	8.8 ±2.1	**14.1 ±1.3**
HGS	full	**75.6 ±4.4**	34.2 ±19.8	69.5 ±4.6	0.9 ±0.2	68.0 ±3.1	63.4 ±3.5	**73.4 ±3.8**	**75.8 ±2.5**
	50%	66.0 ±1.1	33.5 ±2.2	64.9 ±3.1	0.6 ±0.3	63.2 ±5.1	57.5 ±4.4	66.2 ±3.9	**71.5 ±3.1**
	25%	27.7 ±6.3	24.3 ±3.1	41.6 ±5.4	1.3 ±0.3	42.5 ±5.3	40.6 ±3.5	45.2 ±3.1	**48.5 ±2.1**

AVA on `AGS` and `HGS`, respectively, AVA performs better than both methods by larger margins after subsampling (Table 2).

Evaluation on Subsampled OGBench Datasets. Table 2 reports results with 50% and 25% of the trajectories retained from the `stitch` datasets. Subsampling can reduce trajectory overlap and thereby make stitching more difficult. AVA achieves the highest mean success across all four subsampled settings. MQE and OTA are competitive with AVA on the full datasets but fall behind after subsampling. At 25% data, AVA achieves more than twice the mean success rate of MQE on `AGS`. On `HGS`, AVA achieves approximately 1.8 times the mean success rate of OTA.

Additionally, the ordering n-step VA $<$ Random VA $<$ AVA holds in all three `AGS` settings. Random VA varies backup lengths and can alleviate sparse propagation of value increases. Therefore, varying backup lengths could indirectly alleviate misalignment (Section 4.1) and explain the improvement over n-step VA. Nevertheless, AVA explicitly anchors backups to selected junction states and achieves a higher mean success rate than Random VA across all six settings. AVA also achieves higher mean success than Density-based AVA in all six settings. This improvement is consistent with the broader downstream coverage at AVA junctions illustrated in Appendix D.3. Finally, Value-based AVA remains below 2% in all six settings, possibly because value maximization favors overestimated states (Hasselt, 2010).

Value Estimation Accuracy. We examine whether the higher success rates of AVA are accompanied by more accurate value estimates. Because discounting compresses value differences at long temporal distances, we convert value estimates to temporal distances as follows: $d(s,g) = \log(1+(1-\gamma)V(s,g))/\log\gamma$. We measure the relative error $|d-d^*|/d^*$ against the true temporal distance d^*.[4] We average the error along each optimal trajectory and use the same aggregation as for success rates. AVA attains the lowest mean error in all six `stitch` settings (Table 3). On `AGS` with 25% data, AVA reduces the error by 35.9% relative to n-step VA. However, n-step VA attains lower error on both `navigate` datasets. The error of n-step VA on `AGS` is more than three times the error on `AGN`. Because stitching is more important on `AGS` than on `AGN`, the larger error on `AGS` may reflect the stitching limitations of n-step VA. IQL has the highest error on `AGN` and all `humanoidmaze` datasets. The high error might be due to bias accumulation over long horizons in single-step TD learning (Park et al., 2025b).

Table 3: **Value estimation error.** Relative temporal-distance error $|d-d^*|/d^*$. The lowest error in each row is bold.

Dataset	Fraction	IQL	n-step VA	**AVA**
AGS	full	0.495 ±0.004	0.582 ±0.037	**0.411 ±0.022**
	50%	0.494 ±0.002	0.641 ±0.028	**0.409 ±0.013**
	25%	0.512 ±0.003	0.749 ±0.037	**0.480 ±0.018**
AGN	full	0.535 ±0.002	**0.173 ±0.013**	0.206 ±0.004
HGS	full	0.697 ±0.001	0.172 ±0.008	**0.160 ±0.003**
	50%	0.697 ±0.002	0.178 ±0.006	**0.166 ±0.005**
	25%	0.693 ±0.002	0.196 ±0.010	**0.171 ±0.005**
HGN	full	0.705 ±0.002	**0.186 ±0.003**	0.212 ±0.006

Horizon Sweeps. We examine whether the gains of AVA on `AGS` result solely from the incidental decrease in average backup length. We compare AVA with n-step VA across different fixed backup horizons (Figure 4a). AVA achieves higher mean success than n-step VA at every tested horizon. Therefore, the horizon sweep suggests that the gains of AVA do not result solely from shorter average backup lengths. Moreover, at $n=1$, n-step VA uses the same value update as HIQL but differs in low-level policy training. The success rate of 1-step VA remains near zero, similar to the success rate of HIQL on `AGS` (Table 1). Therefore, the change in low-level policy training alone does not explain the performance of AVA. We also vary the maximum horizon of AVA (Figure 4b). Reducing n to 10 lowers success, consistent with less horizon reduction. Increasing n from 25 to 50 or 100

[4]The optimal trajectory used to compute d^* is obtained from Ahn et al. (2025).

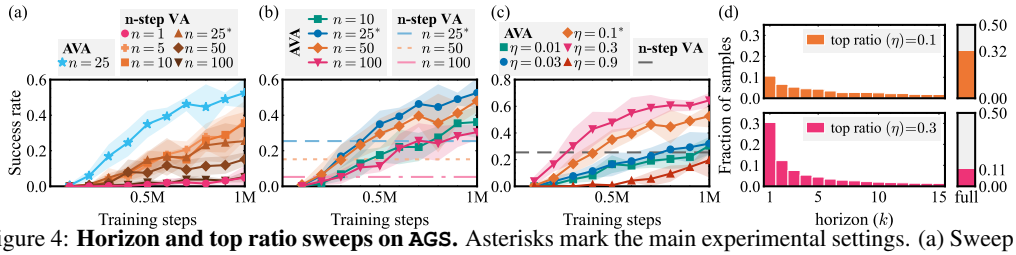


Figure 4: **Horizon and top ratio sweeps on AGS.** Asterisks mark the main experimental settings. (a) Sweep of n for n-step VA. (b) Sweep of n for AVA. (c) Sweep of the top ratio η for AVA. (d) Distributions of the adapted horizon k at $\eta = 0.1$ and 0.3, with $16 \leq k < 25$ omitted for visual clarity. Separate bars show full-length backup fractions ($k = n$).

also reduces the success rate of AVA. However, the relative decrease is smaller for AVA than for n-step VA. The smaller decrease may arise because anchoring at selected junctions limits the potential increase in misalignment at larger maximum horizons.

Top Ratio η. The top ratio η controls the trade-off between junction anchoring and the degree of horizon reduction. On `AGS`, mean success peaks at $\eta = 0.3$ among the tested ratios (Figure 4c). At $\eta = 0.3$, more frequent junction anchoring and fewer full-length backups than at $\eta = 0.1$ accompany higher mean success (Figures 4c and 4d). However, designating too many states as junctions with an excessively large η shortens backups and can increase bias accumulation. For example, at $\eta = 0.9$, AVA approaches single-step TD and success declines substantially. On the other hand, at $\eta = 0.01$ and 0.03, backups are less likely to reach selected junctions, and mean success approaches the level of n-step VA. Appendix D.1 reports sweeps across all seven datasets.

Expectile and Top Ratio. Table 4 examines the interaction between τ and η. In both datasets, higher τ tends to favor lower η. On `AGS`, increasing τ from 0.7 to 0.9 raises the mean success of n-step VA from 24.3% to 54.8%. The improvement may reflect less dilution (Lemma 1). However, increasing τ reduces mean success across all configurations on `HGS`. On `AGS`, mean success also decreases across all configurations when τ increases from 0.9 to 0.95. Therefore, increasing τ does not consistently improve success, possibly because regression becomes harder to optimize (Kostrikov et al., 2022). Nevertheless, at each tested τ, the best AVA configuration achieves higher mean success than n-step VA on both datasets.

Table 4: **Success rates for different τ and η values.** The reporting and boldface conventions follow Table 1.

		n-step VA	AVA			
Dataset	τ	$\eta \to 0$	$\eta = 0.01$	$\eta = 0.03$	$\eta = 0.1$	$\eta = 0.3$
`AGS`	0.7	24.3 ±7.5	24.3 ±7.0	29.7 ±8.0	48.7 ±5.8	**61.9 ±3.0**
	0.9	**54.8 ±6.5**	**53.7 ±6.5**	**55.9 ±11.9**	**54.9 ±10.9**	27.5 ±9.0
	0.95	50.7 ±11.1	**52.5 ±11.4**	**55.0 ±9.5**	27.0 ±12.3	0.0 ±0.0
`HGS`	0.7	69.5 ±4.6	69.9 ±2.3	74.7 ±4.5	75.8 ±2.5	**82.0 ±3.3**
	0.9	60.4 ±4.1	**66.6 ±3.3**	**67.4 ±6.4**	63.3 ±6.4	4.6 ±2.4
	0.95	35.9 ±6.5	34.4 ±5.8	**43.7 ±5.1**	24.2 ±8.4	0.1 ±0.1

Evaluation on LIBERO. Table 5 reports results on LIBERO benchmarks. AVA achieves the highest mean success rate among the evaluated methods, including BC and n-step VA. The results suggest that AVA remains beneficial when near-optimal trajectories are available. Notably, in the LIBERO experiments, AVA and IQL both use flat DDPG+BC for policy extraction with identical hyperparameters. Therefore, the gain over IQL supports the benefit of adaptive horizon reduction in value learning.

Table 5: **Success rates (%) on LIBERO datasets.** The reporting and boldface conventions follow Table 1.

Dataset	BC	IQL	n-step VA	AVA
`KITCHEN-SCENE1`	10.9 ±2.8	11.3 ±2.3	14.9 ±3.8	**18.8 ±5.4**
`KITCHEN-SCENE3`	23.9 ±2.7	16.8 ±5.4	23.9 ±4.2	**29.6 ±5.0**

8 Conclusion and Limitations

We showed that horizon reduction can hinder stitching in offline GCRL. AVA anchors TD backups at junction states to preserve stitching under horizon reduction. We scored each state based on additional downstream coverage to identify useful junctions. AVA improves value estimates on the `stitch` datasets and achieves the highest average success rate across seven OGBench datasets. However, AVA achieves lower mean success than IQL and CGQ on `P33`, and success remains low on `P45`. Action chunking (Li et al., 2025) can improve manipulation performance but may limit reactivity (Li et al., 2026; Song et al., 2026; Chen et al., 2026). Future work could combine AVA with action chunking to improve manipulation performance while preserving reactivity.

AI USE STATEMENT

We drafted the manuscript and used generative AI tools to improve grammar, clarity, and conciseness. For the proofs, we provided the initial ideas and used generative AI tools to assist with the mathematical derivations. We also used generative AI tools to assist with the initial implementation of our methods. Additionally, we used generative AI tools to assist with writing Python code to plot our experimental results. We reviewed all AI-assisted contributions to the manuscript, mathematical derivations, and code. In particular, we verified the derivations and manually revised the code for correctness and consistency with the intended methods. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide dataset details, hyperparameters, and training and evaluation protocols in Appendix B. Additionally, we provide sweeps of the maximum horizon n, the top ratio η, and the cell size used in AVA in Figure 4b, Table 12, and Appendix D.4, respectively. We describe junction scoring and adaptive value learning in Section 5, with policy extraction details in Appendix C.2. The proof of Lemma 1 is provided in Appendix A.1. We will publicly release our code upon acceptance. The OGBench environments, datasets, and baseline implementations used in our experiments are publicly available. The optimal trajectories used to compute d^* are also available in the public repository of Ahn et al. (2025).

REFERENCES

Hongjoon Ahn, Heewoong Choi, Jisu Han, and Taesup Moon. Option-aware temporally abstracted value for offline goal-conditioned reinforcement learning. In *Adv. Neural Inf. Process. Syst.*, pp. 99833–99861, 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 *Adv. Neural Inf. Process. Syst.*, 2017.

Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

Michał Bortkiewicz, Władysław Pałucki, Mateusz Ostaszewski, and Benjamin Eysenbach. Is temporal-difference learning the only path to stitching in RL? In *Proc. ICML Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning*, 2026.

David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? In *Adv. Neural Inf. Process. Syst.*, pp. 1542–1553, 2022.

Ian Char, Viraj Mehta, Adam Villaflor, John M Dolan, and Jeff Schneider. BATS: Best action trajectory stitching. In *Proc. NeurIPS Workshop on Offline Reinforcement Learning*, 2021.

Qian Chen, Junqiao Zhao, Hongtu Zhou, Hang Yu, Yanping Zhao, Chen Ye, and Guang Chen. ACSAC: Adaptive chunk size actor-critic with causal transformer Q-network. *arXiv preprint arXiv:2605.11009*, 2026.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Adv. Neural Inf. Process. Syst.*, pp. 271–278, 1992.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. RvS: What is essential for offline RL via supervised learning? In *Proc. Int. Conf. Learn. Represent.*, 2022.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Adv. Neural Inf. Process. Syst.*, pp. 35603–35620, 2022.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Adv. Neural Inf. Process. Syst.*, pp. 20132–20145, 2021.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *Proc. Int. Conf. Learn. Represent.*, 2021.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *Proc. Int. Conf. Mach. Learn.*, pp. 11321–11339, 2023.

Raj Ghugare, Matthieu Geist, Glen Berseth, and Benjamin Eysenbach. Closing the gap between td learning and supervised learning-a generalisation point of view. In *Proc. Int. Conf. Learn. Represent.*, pp. 13774–13792, 2024.

Hado Hasselt. Double Q-learning. In *Adv. Neural Inf. Process. Syst.*, 2010.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Charles A Hepburn and Giovanni Montana. Model-based trajectory stitching for improved offline reinforcement learning. In *Proc. NeurIPS Workshop on Offline Reinforcement Learning*, 2022.

Ahad Jawaid. Offline RL with hierarchical action chunking. *arXiv preprint arXiv:2607.20834*, 2026.

Hyungkyu Kang, Byeongchan Kim, and Min-hwan Oh. Latent representation alignment for offline goal-conditioned reinforcement learning. In *Proc. Int. Conf. Mach. Learn.*, 2026.

Michael J Kearns and Satinder Singh. "Bias-Variance" error bounds for temporal difference updates. In *Proc. Conf. Learn. Theory*, pp. 142–147, 2000.

Sungyoon Kim, Yunseon Choi, Daiki E Matsunaga, and Kee-Eung Kim. Stitching sub-trajectories with conditional diffusion model for goal-conditioned offline RL. In *Proc. Conf. Adv. Artif. Intell.*, pp. 13160–13167, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Represent.*, 2015.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *Proc. Int. Conf. Learn. Represent.*, 2022.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? In *Proc. Int. Conf. Learn. Represent.*, 2022.

Kyowoon Lee and Jaesik Choi. State-covering trajectory stitching for diffusion planners. *Adv. Neural Inf. Process. Syst.*, pp. 57273–57303, 2025.

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

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *Proc. Int. Conf. Learn. Represent.*, 2019.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Adv. Neural Inf. Process. Syst.*, pp. 55518–55553, 2025.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled Q-chunking. In *Proc. Int. Conf. Learn. Represent.*, pp. 145315–145389, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Proc. NeurIPS Datasets and Benchmarks Track*, pp. 44776–44791, 2023.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals. In *Proc. Int. Conf. Learn. Represent.*, pp. 78599–78621, 2025.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In *Proc. Int. Joint Conf. Artif. Intell.*, 2022.

Yunhao Luo, Utkarsh Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. *Adv. Neural Inf. Process. Syst.*, pp. 37809–37843, 2025.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proc. PMLR Conf. Robot Learn.*, pp. 1113–1132, 2020.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *Proc. Int. Conf. Learn. Represent.*, 2023.

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

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 *Proc. Int. Conf. Mach. Learn.*, 2024.

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In *Adv. Neural Inf. Process. Syst.*, pp. 19654–19679, 2025.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. *Adv. Neural Inf. Process. Syst.*, pp. 34866–34891, 2023.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline RL? In *Adv. Neural Inf. Process. Syst.*, pp. 79029–79056, 2024a.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proc. Int. Conf. Mach. Learn.*, 2024b.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *Proc. Int. Conf. Learn. Represent.*, pp. 94937–94982, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes RL scalable. In *Adv. Neural Inf. Process. Syst.*, pp. 8350–8389, 2025b.

Seohong Park, Aditya Oberai, Pranav Atreya, and Sergey Levine. Transitive RL: Value learning via divide and conquer. In *Proc. Int. Conf. Learn. Represent.*, pp. 3217–3235, 2026.

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.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proc. Int. Conf. Mach. Learn.*, pp. 1312–1320, 2015.

Yongjae Shin, Jongseong Chae, Seongmin Kim, Jongeui Park, and Youngchul Sung. Adaptive action chunking via multi-chunk Q value estimation. *arXiv preprint arXiv:2605.10044*, 2026.

Uladzislau 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 *Adv. Neural Inf. Process. Syst.*, pp. 43905–43941, 2025.

Gwanwoo Song, Kwanyoung Park, and Youngwoon Lee. Chunk-guided Q-learning. In *Proc. Int. Conf. Mach. Learn.*, 2026.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proc. Connectionist Models Summer School*, 1993.

Mianchu Wang, Rui Yang, Xi Chen, Hao Sun, Meng Fang, and Giovanni Montana. GOPlan: Goal-conditioned offline reinforcement learning by planning with learned models. *Trans. Mach. Learn. Research*, 2024.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *Proc. Int. Conf. Mach. Learn.*, pp. 36411–36430, 2023.

Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. *Adv. Neural Inf. Process. Syst.*, pp. 4085–4098, 2022.

Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. Efficient multi-agent offline coordination via diffusion-based trajectory stitching. In *Proc. Int. Conf. Learn. Represent.*, pp. 25018–25045, 2025.

Bill Chunyuan Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Multistep quasimetric learning for scalable goal-conditioned reinforcement learning. *arXiv preprint arXiv:2511.07730*, 2025.

Kexin Zheng, Xianyuan Zhan, and Xintao Yan. Horizon adaptive offline policy learning via value stitching. *arXiv preprint arXiv:2606.21136*, 2026.

Zhaoyi Zhou, Chuning Zhu, Runlong Zhou, Qiwen Cui, Abhishek Gupta, and Simon Du. Free from bellman completeness: Trajectory stitching via model-based return-conditioned supervised learning. In *Proc. Int. Conf. Learn. Represent.*, pp. 3687–3711, 2024.

A Proofs and Derivations

A.1 Expectile Regression with Sparse Higher Targets

Proof of Lemma 1. Let v denote the single value fitted to the states. For $\alpha, \tau \in (0, 1)$, the regression objective is defined as follows:

$$F(v) = \alpha L_2^\tau(y^+ - v) + (1 - \alpha) L_2^\tau(y^- - v).$$

Because the minimizer lies in (y^-, y^+), the residual from y^+ is positive and the residual from y^- is negative. Then, the objective can be written as follows:

$$F(v) = \tau\alpha(y^+ - v)^2 + (1 - \tau)(1 - \alpha)(v - y^-)^2.$$

Because F is strictly convex on the interval, we find the minimizer $v^\star$ by setting the derivative to zero as follows:

$$\tau\alpha(v^\star - y^+) + (1 - \tau)(1 - \alpha)(v^\star - y^-) = 0.$$

Then, we obtain $v^\star$ as follows:

$$v^\star = y^- + \rho(y^+ - y^-), \qquad \rho = \frac{\tau\alpha}{\tau\alpha + (1 - \tau)(1 - \alpha)}. \tag{4}$$

Because $0 < \rho < 1$, $v^\star$ lies in (y^-, y^+).

For any $q \in (0, 1)$, we rewrite $\rho \geq q$ as follows:

$$\begin{aligned} \rho \geq q &\iff (1 - q)\tau\alpha \geq q(1 - \tau)(1 - \alpha) \\ &\iff \tau \geq \frac{q(1 - \alpha)}{(1 - q)\alpha + q(1 - \alpha)}. \end{aligned} \tag{5}$$

□

B Detailed Experimental Setup

B.1 Evaluation Protocol

We follow the evaluation protocol of OGBench (Park et al., 2025a). We train each method for 1M steps and evaluate at 0.8M, 0.9M, and 1.0M steps. For OGBench and LIBERO, we run 50 rollouts per goal at each evaluation. For OGBench, we use 5 fixed goals, so each seed yields 750 episodes across the three evaluations. For LIBERO, we use the tasks in each scene as evaluation goals (Table 7). In the tables, we report the mean success rate and standard deviation across 5 random seeds. In the learning curves, lines and markers indicate the mean across 5 random seeds. Shaded regions show ± 1 standard deviation around the mean.

B.2 Dataset Metadata

Figure 5 shows example environments from OGBench (Park et al., 2025a) and LIBERO (Liu et al., 2023). Tables 6 and 7 summarize dataset sizes, maximum episode lengths, data trajectory lengths, and state and action dimensions for OGBench and LIBERO, respectively.

The task descriptions for `KITCHEN-SCENE1` in LIBERO-90 are as follows:

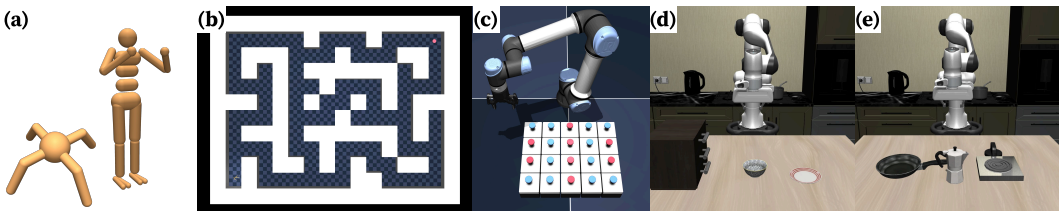


Figure 5: Example environments from OGBench and LIBERO. (a) Ant and humanoid agents. (b) The giant maze in OGBench. (c) The `P45` environment in OGBench. (d) `KITCHEN-SCENE1`, (e) `KITCHEN-SCENE3` from LIBERO-90.

- open_the_bottom_drawer_of_the_cabinet
- open_the_top_drawer_of_the_cabinet
- open_the_top_drawer_of_the_cabinet_and_put_the_bowl_in_it
- put_the_black_bowl_on_the_plate
- put_the_black_bowl_on_top_of_the_cabinet

The task descriptions for KITCHEN-SCENE3 are as follows:

- put_the_frying_pan_on_the_stove
- put_the_moka_pot_on_the_stove
- turn_on_the_stove
- turn_on_the_stove_and_put_the_frying_pan_on_it

Table 6: **OGBench metadata.** Specifications follow OGBench (Park et al., 2025a). Dataset sizes count transitions before subsampling.

Dataset	Dataset size (transitions)	Maximum episode length	Data trajectory length	State dimension	Action dimension
AGS	1M	1000	200	29	8
HGS	4M	4000	400	69	21
AGN	1M	1000	2000	29	8
HGN	4M	4000	4000	69	21
P33	1M	500	1000	55	5
P44	1M	500	1000	83	5
P45	3M	1000	1000	99	5

Table 7: **LIBERO metadata.** The action vector includes position and orientation components and a gripper open/close command.

Dataset	Dataset size (transitions)	Maximum episode length	Average data trajectory length	State dimension	Action dimension	Num. tasks
KITCHEN-SCENE1	31337	750	139.3	67	7	5
KITCHEN-SCENE3	32179	750	178.8	57	7	4

B.3 Common Hyperparameters

Table 8: **Common hyperparameters.**

Hyperparameter	Value
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2015)
# gradient steps	1M
Minibatch size	1024
MLP dimensions	$(512, 512, 512)$
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Target smoothing coefficient	0.005
Discount factor (γ)	0.998 (HG$\star$), 0.995 (AG$\star$), 0.99 (others)
Policy goal sampling ratio ($p_{\text{cur}}, p_{\text{traj}}, p_{\text{geom}}, p_{\text{rand}}$)	$(0, 0.5, 0, 0.5)$ (stitch), $(0, 0.5, 0, 0.5)$ (others)
Value goal sampling ratio ($p_{\text{cur}}, p_{\text{traj}}, p_{\text{geom}}, p_{\text{rand}}$)	$(0.2, 0, 0.5, 0.3)$

All methods share the settings in Table 8, most of which follow the OGBench defaults (Park et al., 2025a). One exception is the discount factor for HG$\star$. Although the maximum episode length for HG$\star$ is 4000 steps, the default discount factor of 0.995 leaves less than 10% of the reward signal after 500 steps. Therefore, we use $\gamma = 0.998$ for all methods on HG$\star$ to reduce reward decay. With $\gamma = 0.998$, approximately 37% of the reward signal remains after 500 steps.

B.4 AVA HYPERPARAMETERS

Table 9 lists the hyperparameters used for value learning in AVA in the main experiments. Notably, we apply the same criterion for value learning hyperparameters across OGBench and LIBERO datasets. For the policy extraction hyperparameters, please refer to Table 10.

Table 9: **AVA value learning hyperparameters.**

Hyperparameter	Value
Value Learning Hyperparameters	
Maximum horizon (n)	25
Top ratio (η)	0.1
Expectile parameter (τ)	0.7
Cell size (AVA)	0.1 (locomotion), 0.02 (manipulation)
Cell size (AVA w/ HILP)	5.0
HILP Encoder Hyperparameters	
Latent dimension	16
Expectile parameter (τ)	0.9
Discount factor (γ)	0.995 (locomotion), 0.99 (manipulation)
Training steps	0.5M

Table 10: **AVA policy extraction hyperparameters.**

Benchmark	Dataset	h	β_h	β_l	Subgoal dim.
OGBench	`AG*`	25	3.0	0.1	10
	`HG*`	100	1.0	0.1	10
	`P*`	–	–	0.3	–
LIBERO		–	–	10.0	–

B.5 BASELINE HYPERPARAMETERS

For IQL, HIQL, CRL, and QRL, we use the environment-specific hyperparameters reported in OGBench (Park et al., 2025a), except for the common settings in Table 8. Notably, in the LIBERO experiments, both IQL and AVA use flat DDPG+BC with a behavioral cloning weight of $\beta_l = 10.0$. For the remaining baselines, we follow the original papers for method-specific hyperparameters. When settings for the evaluated tasks are unavailable, we transfer settings from related tasks according to the criteria described below.

LAVL. For LAVL, we follow the original hyperparameters for each `Locomotion` task (Kang et al., 2026). The original paper does not evaluate `puzzle` tasks, so we select the settings from `cube` tasks with matching maximum episode lengths (Park et al., 2025a). `P33`, `P44`, and `cube-double` share a maximum episode length of 500 steps. Therefore, we use the `cube-double` settings for `P33` and `P44`. Similarly, we use the `cube-triple` settings for `P45` because both tasks have a maximum episode length of 1000 steps.

SHARSA. For SHARSA, we set the horizon reduction factor to $n = 50$ for `Locomotion` tasks and $n = 25$ for `Manipulation` tasks. The values follow the settings reported for `humanoidmaze` and `puzzle`, respectively, in the original paper (Park et al., 2025b). For rejection sampling, we use 32 flow-based subgoals, as in the original paper.

OTA. For OTA, we follow the original hyperparameters for each `Locomotion` task (Ahn et al., 2025). For `puzzle` tasks, we transfer the original OTA hyperparameters from `cube` tasks using the same criterion as for LAVL. We use the `cube-double` settings for `P33` and `P44`, and the `cube-triple` settings for `P45`. Table 11 summarizes the settings.

CGQ. Because the official CGQ implementation is not goal-conditioned, we adapt CGQ for goal-conditioned RL. We add the goal to the policy and critic inputs. For CGQ, α and α_{step} denote the BC coefficients for the chunked and single-step policies, respectively. The remaining hyperparameters are the distillation coefficient β, distillation expectile τ, and chunk size h. For all `Locomotion`

Table 11: **OTA hyperparameters.**

Dataset	β_h	β_l	HIQL subgoal horizon	Value abstraction horizon
AGS	3.0	3.0	30	10
HGS	0.5	3.0	100	20
AGN	0.5	3.0	16	4
HGN	0.5	3.0	100	20
P33	3.0	3.0	20	4
P44	3.0	3.0	20	4
P45	3.0	3.0	25	4

tasks, we use $(\alpha, \alpha_{\text{step}}, \beta, \tau, h) = (300, 10, 0.01, 0.8, 10)$, as reported for `antmaze-giant` in the original paper (Song et al., 2026). For `HGS` and `HGN`, we additionally evaluate all four combinations of $\alpha \in \{100, 300\}$ and $\beta \in \{0.01, 0.5\}$. The candidate values follow the settings reported for `antmaze-giant` and `humanoidmaze-medium` in the original paper. However, all four configurations yield a success rate of 0% on both datasets. For `puzzle` tasks, we use $(100, 100, 0.01, 0.8, 20)$, as reported for `puzzle-3×3-sparse`.

MQE. Following the original paper (Zheng et al., 2025), we use 8 MRN components and a waypoint discount of $\lambda = 0.95$. We set $p = 0.2$ for sampling the next state s'_t as the waypoint. We also follow the environment-specific BC coefficients reported in the original paper. The BC coefficient is $\alpha = 0.03$ for `AGS` and $\alpha = 0.1$ for `AGN`. We use $\alpha = 0.01$ for `HGS` and `HGN`, and $\alpha = 1.0$ for all `puzzle` tasks.

TRL. For `HG⋆` and `P⋆`, we use the hyperparameters reported in the original TRL paper (Park et al., 2026). For `AG⋆`, we consider two BC coefficients reported for related locomotion tasks. The original paper uses $\alpha = 0.7$ for `antmaze-large` and $\alpha = 0.1$ for `humanoidmaze-giant`. We evaluate both coefficients on each `AG⋆` dataset and report the result with the higher success rate. The selected coefficients are $\alpha = 0.1$ for `AGS` and $\alpha = 0.7$ for `AGN`. In the original TRL paper, goals for value learning are sampled only from the same trajectory. Therefore, we set the value goal sampling ratios for TRL as follows: $(p_{\text{cur}}, p_{\text{traj}}, p_{\text{geom}}, p_{\text{rand}}) = (0, 0, 1, 0)$.

C Discussion of Design Choices

C.1 Off-Policy Bias in Multi-Step Value Learning

The multi-step targets can be biased when the behavior policy differs from the learned policy (Li et al., 2025; 2026). The difference can change the intermediate rewards and bootstrap state. However, every non-goal state yields the same reward in the sparse goal-conditioned setting. Therefore, for the same backup length with no intermediate goal visit, the accumulated reward contributes no off-policy bias. Alternatively, action chunking can remove multi-step off-policy bias because the critic evaluates the exact action sequence that generated the observed rewards (Li et al., 2025). However, long open-loop action chunks can limit the reactivity needed for trajectory stitching (Li et al., 2026; Song et al., 2026; Chen et al., 2026). Therefore, we use multi-step targets without action chunking because rewards are sparse and trajectory stitching is important in our setting.

C.2 Policy Extraction Details

By default, we follow the hierarchical structure of HIQL (Park et al., 2023). The hierarchy consists of a high-level policy $\pi^h_{\theta_h}$ and a low-level policy $\pi^l_{\theta_l}$. The high-level policy predicts a subgoal given the current state s_t and the goal g. During evaluation, the low-level policy selects an action conditioned on the current state and the subgoal provided by the high-level policy. Notably, the subgoal horizon h is fixed independently of the adapted backup horizon k used for value learning.

As in HIQL (Park et al., 2023), we use learned goal representations in the hierarchical structure. In the equations below, we omit goal encoding from the notation for simplicity. We train the high-level policy with AWR (Peng et al., 2019), as in HIQL. The loss function for the high-level policy is as

Table 12: **Per-task sweep of the top ratio η.** Success rates (%) averaged over 5 seeds, with the standard deviation across seeds. The asterisk marks the default value used in the main results. The reporting and bold conventions follow Table 1.

		AVA with top ratio η			
Dataset	n-step VA	0.01	0.03	0.1*	0.3
antmaze-giant-stitch	24.3 ±7.5	24.3 ±7.0	29.7 ±8.0	48.7 ±5.8	**61.9 ±3.0**
humanoidmaze-giant-stitch	69.5 ±4.6	69.9 ±2.3	74.7 ±4.5	75.8 ±2.5	**82.0 ±3.3**
antmaze-giant-navigate	**88.8 ±1.6**	**88.3 ±2.1**	**92.0 ±1.4**	**90.3 ±1.2**	**90.8 ±0.9**
humanoidmaze-giant-navigate	**90.1 ±2.5**	**90.8 ±1.9**	**92.7 ±0.7**	**94.7 ±1.1**	88.9 ±12.2
puzzle-3×3	67.5 ±5.2	62.1 ±8.6	63.9 ±7.5	74.1 ±5.4	**89.7 ±3.0**
puzzle-4×4	63.0 ±7.7	68.6 ±4.8	72.8 ±2.6	**77.7 ±2.8**	65.3 ±6.8
puzzle-4×5	15.4 ±1.4	15.6 ±1.4	17.9 ±1.1	**18.5 ±0.8**	**19.3 ±0.4**
Average	59.8	60.0	63.4	**68.5**	**71.1**

follows:

$$\mathcal{L}_{\pi^h}(\theta_h) = -\mathbb{E}_{(s_t, s_{t+h}, g)\sim\mathcal{D}}\left[\exp\big(\beta_h\big(V_\phi(s_{t+h}, g) - V_\phi(s_t, g)\big)\big)\log \pi^h_{\theta_h}(s_{t+h} \mid s_t, g)\right], \tag{6}$$

where β_h is the inverse temperature. The objective assigns greater weight to subgoals with higher estimated advantages.

We train the low-level policy with DDPG+BC (Fujimoto & Gu, 2021) instead of the AWR used in HIQL. To obtain action gradients, we additionally train a single-step Q function that bootstraps from the learned value function. Let $\mu^l_{\theta_l}(s, g^l)$ denote the deterministic action output of $\pi^l_{\theta_l}$, where g^l is the low-level goal. The loss functions for the Q function and the low-level policy are as follows:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{(s_t, a_t, s_{t+1}, g)\sim\mathcal{D}}\left[\big(Q_\psi(s_t, a_t, g) - r(s_t, g) - \gamma V_{\hat{\phi}}(s_{t+1}, g)\big)^2\right], \tag{7}$$

$$\mathcal{L}_{\pi^l}(\theta_l) = -\mathbb{E}_{(s_t, a_t, g^l)\sim\mathcal{D}}\left[\frac{Q_\psi(s_t, \mu^l_{\theta_l}(s_t, g^l), g^l)}{\frac{1}{B}\sum_{j=1}^{B}|Q_\psi(s_j, a_j, g^l_j)|} + \beta_l \log \pi^l_{\theta_l}(a_t \mid s_t, g^l)\right], \tag{8}$$

where β_l is the behavioral cloning weight and B is the batch size.

In the hierarchical policy, g^l is a dataset subgoal during training. During evaluation, g^l is provided by the high-level policy. As described in Section 5, we use a flat policy for manipulation tasks and set $g^l = g$.

C.3 Limitations of Hierarchical Policy Extraction

We discuss policy extraction limitations using results from OGBench (Park et al., 2025a), which evaluates multiple policy extraction methods across diverse tasks. The OGBench results suggest that HIQL can be less effective on state-based manipulation tasks. HIQL achieves strong performance on most locomotion tasks, including `antmaze` and `humanoidmaze`. However, HIQL achieves low success rates on state-based manipulation tasks such as `cube` and `puzzle` in OGBench. A possible explanation is the difficulty of learning useful subgoal representations for manipulation tasks (Park et al., 2025a). The subgoal representations must capture useful information about both the robot state and the object states.

D Additional Experimental Results

D.1 Top Ratio Sweeps Across Datasets

We examine the sensitivity of AVA to the top ratio by evaluating $\eta \in \{0.01, 0.03, 0.1, 0.3\}$ across all seven datasets (Table 12). AVA achieves higher mean success than n-step VA on all seven datasets at $\eta = 0.1$ and on six datasets at $\eta = 0.3$. Increasing η from 0.1 to 0.3 improves mean success on five datasets and raises average success from 68.5% to 71.1%. However, mean success decreases from 77.7% to 65.3% on `P44` and from 94.7% to 88.9% on `HGN`.

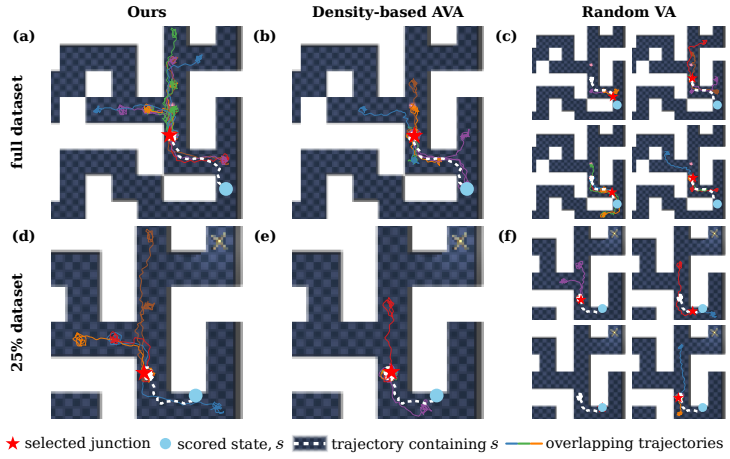


Figure 7: Comparison of selected junctions on the full AGS dataset and the 25% subset. In the illustrated examples, AVA junctions have broader downstream coverage than Density-based AVA and Random VA junctions.

D.2 Low-Level Policy Extraction with AWR

We examine whether the gains of AVA persist with the default policy extraction setup of HIQL. For both AVA and n-step VA, we replace DDPG+BC with AWR for low-level policy training (Figure 6). We set the inverse temperature to 3.0, as in HIQL. AVA still achieves a higher success rate than n-step VA. Therefore, the improvement is not limited to the low-level policy extraction method used in the main experiments.

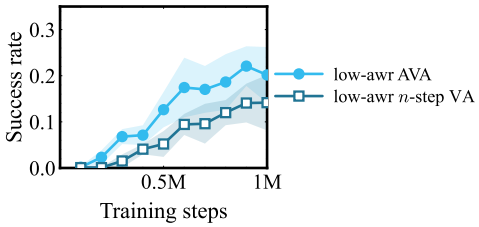


Figure 6: Comparison of AVA and n-step VA with AWR for low-level policy training on AGS.

D.3 Junction Selection Quality

Figure 7 compares the junctions selected under each criterion. For comparison, we select trajectories containing both AVA junctions and Density-based AVA junctions. For Random VA, we randomly sample four states from each selected trajectory to compare all three methods on the same trajectories. For each junction s in Figure 7, dashed white lines represent the trajectory τ_s containing s, and solid colored lines represent downstream segments $\tau_{>m}$ for $m \in \mathcal{M}(s)$. On both the full AGS dataset and the 25% subset, AVA junctions have broader downstream coverage than Density-based AVA and Random VA junctions. The broader downstream coverage is consistent with the higher success rates of AVA over Density-based AVA and Random VA (Table 2).

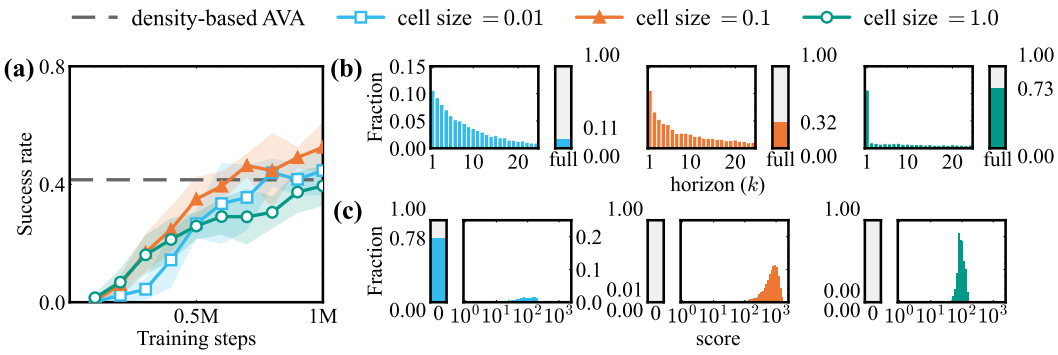
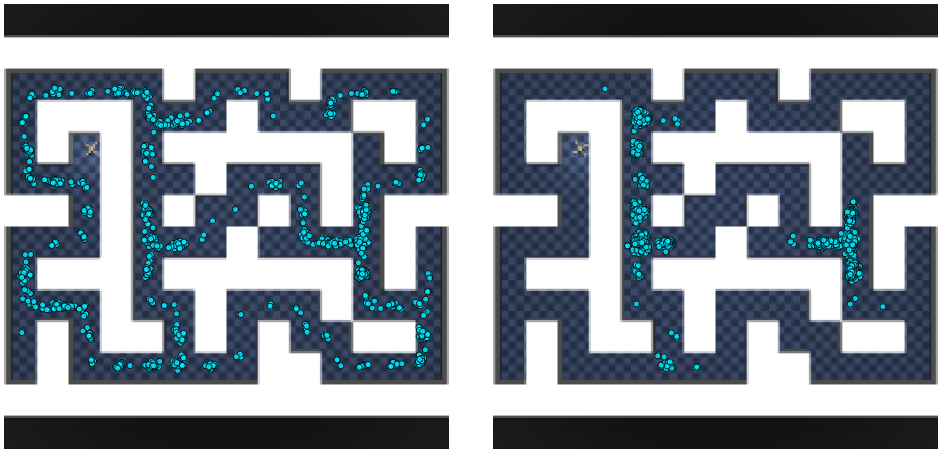


Figure 8: Effect of cell size on AVA on AGS. (a) Learning curves, with the dashed line showing Density-based AVA. (b) Distributions of adapted horizon lengths k. The bars to the right of each plot represent the fraction of full-length backups ($k = n$). (c) Distributions of junction scores. The bars to the left of each plot represent the fraction of states with zero junction scores.



(a) cell size = 0.1 **(b) cell size = 1.0**

Figure 9: Spatial distribution of selected junctions on AGS with cell sizes of (a) 0.1 and (b) 1.0. We subsample and display 1000 junctions for visual clarity.

D.4 Effect of Cell Size

Figure 8a compares success rates across three cell sizes for junction scoring. The default cell size of 0.1 achieves the highest mean success among the tested values. Larger cells may increase the imbalance in trajectory counts across cells. More overlapping trajectories tend to provide broader downstream coverage. Therefore, cells may receive high junction scores simply because they contain many trajectories. At a cell size of 1.0, the selected junctions cluster at a few locations (Figure 9b). The concentration can prevent many backups from finding a junction before the maximum horizon. With η fixed, 73% of backups reach the maximum horizon at a cell size of 1.0, compared with 32% at the default cell size of 0.1. Therefore, the value updates of AVA become more similar to the value updates of n-step VA.

Smaller cells are less likely to contain states from multiple trajectories. When $\mathcal{M}(s)$ contains only states from τ_s, the junction score in Equation 2 is zero by construction. At a cell size of 0.01, 78% of states receive a score of zero. In this case, even a single overlapping trajectory can give s a positive score. When an overlap alone can rank s above the many states with zero scores, junction selection may resemble Density-based AVA. The success rate is also close to that of Density-based AVA, as shown in Figure 8a. The backup-length distributions at cell sizes 0.01 and 0.1 show similar patterns for $k < n$. However, AVA achieves higher mean success with a cell size of 0.1. Therefore, similar patterns in backup lengths below n do not necessarily imply similar success rates.