Temporal Distance-aware Subgoal Generation for Offline Hierarchical Reinforcement Learning

Taegeon Park*
Sungkyunkwan University
Suwon, Republic of Korea
tg.park@g.skku.edu

Seungho Baek
Sungkyunkwan University
Suwon, Republic of Korea
qortmdgh4141@g.skku.edu

Jongchan Park
Sungkyunkwan University
Suwon, Republic of Korea
marin0625@skku.edu

Seungjun Oh
Sungkyunkwan University
Suwon, Republic of Korea
depthfirst@g.skku.edu

Yusung Kim†
Sungkyunkwan University
Suwon, Republic of Korea
yskim525@skku.edu

Abstract

Efficient subgoal generation is essential in offline Hierarchical Reinforcement Learning (HRL) for tackling long-horizon and sparse-reward tasks. Existing approaches often struggle with redundant and inefficient subgoal candidates and fail to maintain meaningful temporal relationships due to fixed-step subgoal sampling. To address these issues, we propose Temporal Distance-Aware Subgoal Generation (TDSG), a novel framework leveraging pre-trained Temporal Distance (TD) representations. TDSG identifies a compact set of anchor states in the TD representation space. These states, evenly spaced at consistent temporal distance intervals and collectively covering all states in the dataset while comprising less than 1% of the entire dataset, serve as the training targets for subgoal generation. This ensures efficient and temporally consistent high-level policy learning. Furthermore, the low-level policy leverages intrinsic rewards derived from the alignment between current states and subgoals in the TD representation space, enabling effective learning even under sparse-reward conditions. Experimental results demonstrate that TDSG achieves consistent performance improvement over prior offline HRL methods across numeric and visual environments. Our code is available at https://github.com/Ptaegeon/TDSG.git

CCS Concepts

• **Theory of computation → Reinforcement learning**.

Keywords

Offline Hierarchical Reinforcement learning; Subgoal Generation;

ACM Reference Format:
Taegeon Park, Seungho Baek, Jongchan Park, Seungjun Oh, and Yusung Kim. 2025. Temporal Distance-aware Subgoal Generation for Offline Hierarchical Reinforcement Learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746252.3761326

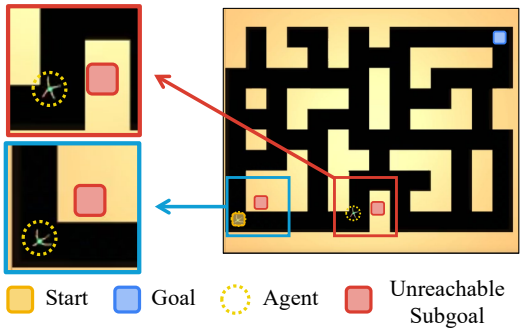

Figure 1: Existing offline HRL methods often generate subgoals that the low-level policy cannot reach.

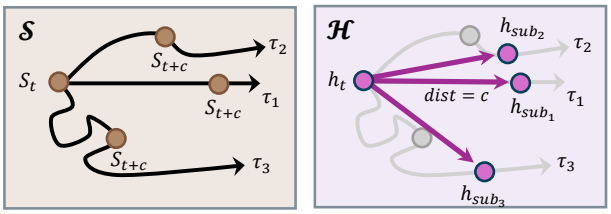

Figure 2: Fixed environment steps are often used to sample subgoal candidates from an offline dataset. However, this approach can result in significant variations in temporal distances across trajectories, leading to inconsistencies in subgoal spacing and reduced effectiveness in policy training.

1 Introduction

Reinforcement learning (RL) has traditionally relied on online interactions with environments to iteratively improve a behavior policy [28]. This paradigm has achieved remarkable success in controlled (simulation) settings, such as board games, Atari video games, autonomous driving, and robotic locomotion and manipulation [10, 29, 38, 42]. However, the dependence on online data collection poses significant challenges in scenarios where acquiring such data is impractical or prohibitively expensive. To address

*First author.
†Corresponding author.

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761326

this issue, Offline RL has emerged as a promising paradigm that leverages pre-existing datasets to eliminate the need for online interactions [20, 22].

Despite its potential, training policies for tasks characterized by long horizons and sparse rewards, where success rewards are received only upon achieving a distant goal, remain a difficult challenge [9, 13]. Hierarchical Reinforcement Learning (HRL) has gained attention as an effective approach to address these challenges by decomposing complex tasks into smaller, manageable subtasks [34, 39]. HRL employs a high-level policy to generate subgoals and a low-level policy to achieve these subgoals, enabling efficient learning even in long-horizon and sparse-reward settings. Although recent advances have explored the use of offline datasets for HRL, existing methods still suffer from several limitations.

First, existing approaches often treat all states in the offline dataset as potential subgoals. This unnecessarily includes redundant subgoal candidates, degrading training efficiency and the quality of subgoal generation as shown in Figure 1. Second, all the methods simply sample states located at fixed environment steps behind within the same trajectory (as illustrated in Figure 2) and use them as subgoal candidates for training. However, in offline datasets where behavior quality is not consistently optimal, states sampled at fixed steps behind may fail to maintain meaningful temporal relationships. For instance, in a quadruped robot's maze navigation task, trajectories may include zigzag movements, collisions with walls, or repeated stalling at the same location. Using such states as subgoals can lead to suboptimal subgoal quality, hindering high-level policy training and ultimately limiting the overall performance of HRL.

Recent studies have demonstrated that it is possible to learn the optimal temporal distance (TD), defined as the minimum number of time steps required for the transition between any two states, using unsupervised learning on offline datasets [35]. TD representations can serve as a fundamental component for various algorithms in planning, control, and goal-reaching tasks. Moreover, pre-trained TD representations have shown the potential for zero-shot policy learning in new downstream tasks. However, these studies yet fail to integrate TD representations into hierarchical learning frameworks, leaving long-horizon and sparse-reward problems largely unresolved.

To address these challenges, we propose a novel framework, Temporal Distance-Aware Subgoal Generation (TDSG). The main contributions of our work are as follows.

- Our method addresses the challenges of long-horizon and sparse-reward tasks in offline HRL by leveraging temporal distance representations.
- TDSG identifies the compact set of anchor states, comprising less than 1% of the dataset, as subgoal candidates, reducing the redundancy and ensuring meaningful temporal relationships in subgoal sampling for high-level policy training.
- By employing intrinsic rewards based on the alignment between current states and generated subgoals in the TD representation space, TDSG enhances the performance of offline HRL.
- Extensive experiments on long-horizon navigation and multi-task robotic manipulation tasks demonstrate that TDSG achieves 13.5% improvement over prior offline HRL methods, across numeric and visual environments.

Experimental results demonstrate that TDSG consistently outperforms recent offline RL methods across both numeric and visual settings. Furthermore, extensive ablation studies highlight the importance of each design component in achieving the overall performance improvements of the proposed framework.

2 Related works

2.1 Offline Reinforcement Learning

Offline Reinforcement Learning (RL) leverages pre-collected datasets to train RL models in scenarios where real-time data acquisition may be infeasible, prohibitively expensive, or even risky [22, 24]. However, a critical problem in offline RL arises from the distributional shift [25] between the behaviors observed in the offline datasets and the behaviors of the newly learned policy, making it challenging to generalize effectively. Accordingly, prior work has focused on aligning the learned policy with offline data constraints, including explicit regularization [19, 43], conservative Q learning [21], and importance sampling approaches [12]. In addition, some research employs an advantage-weighted supervised learning framework, which imposes implicit constraints on the offline dataset while avoiding excessively conservative updates [19, 31]. On the other hand, model-based offline RL methods address distributional shifts by estimating model uncertainty [46]. Another line of work investigates unsupervised (self-supervised) offline pretraining of state representations to improve generalization and transfer in downstream decision-making tasks [44]. However, such flat policy approaches often face challenges in long-horizon tasks.

2.2 Hierarchical Policy Learning

Hierarchical Reinforcement Learning (HRL) tackles long-horizon tasks by decomposing them into subtasks with multi-level policies [3, 30]. Goal-conditioned HRL, where high-level policies set subgoals, has proven effective in sparse-reward tasks [5]. Recent online HRL studies, such as HIGL [17] and NGTE [33], guide high-level exploration by constructing graphs over promising states (e.g., landmarks or outposts) and generating subgoals from their nodes, but they typically require costly online interactions.

Recent works extend HRL to offline settings by discovering skills or subgoals from datasets. For instance, OPAL [1] discovers skills offline through a high-level policy, while a low-level policy executes them via behavior cloning. Subgoal Diffuser [14] and HDMI [26] generate subgoal trajectories via diffusion-based imitation learning. Although effective with high-quality demonstrations, these methods are sensitive to data quality. While these imitation-based methods show promising results in settings with high-quality demonstrations, they are inherently sensitive to offline data quality. To address such limitations, offline hierarchical RL methods have been proposed. For instance, Guider [39] addresses this issue but still depends on hand-crafted goal spaces.

HIQL [34] mitigates this by using latent embeddings as goals, enabling HRL in high-dimensional environments. However, it overlooks temporal distance, often leading to inconsistent subgoal transitions. In our concurrent work, we propose constructing a graph with TD representations, where nodes serve as subgoals [2].

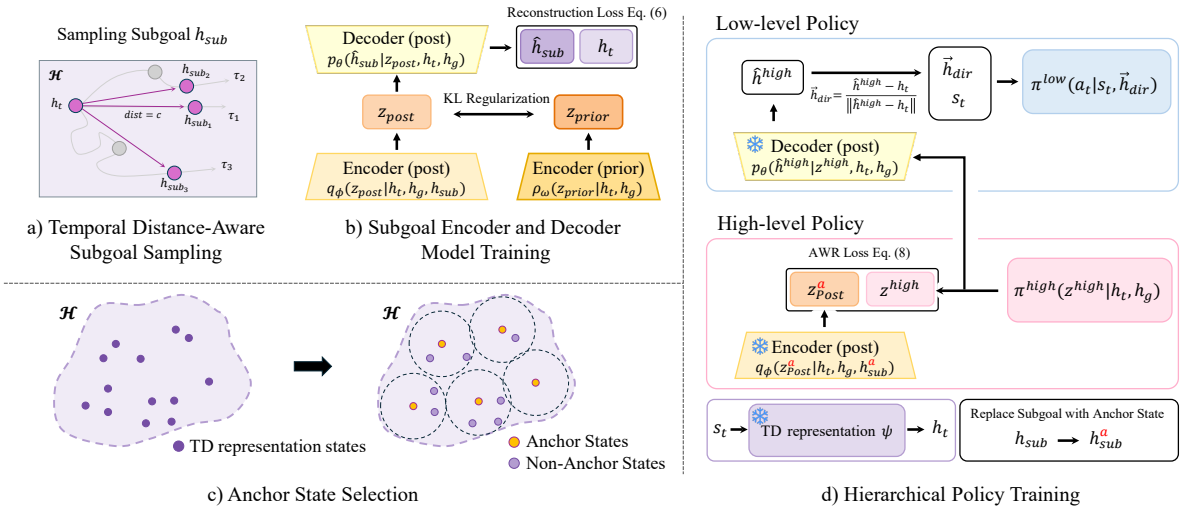

Figure 3: The overall framework of TDSG. It highlights the major components: TD-aware subgoal sampling, subgoal encoder-decoder pre-training, anchor state selection, and hierarchical policy training.

3 Preliminaries

3.1 Notation

We consider continuous control problems within the framework of goal-conditioned Markov Decision Processes (MDPs). The problem is defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ denotes the action space, $p(s'|s,a)$ represents the transition probability, $r(s,g)$ is a goal-conditioned reward function, and γ is the discount factor. The objective is to optimize the policy $\pi(a|s,g)$ to maximize the expected cumulative return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r(s_t, g)]$. In goal-conditioned and sparse reward environments, the goal-conditioned reward function assigning a value of 1 if the agent reaches the predefined target goal or successfully completes a designated subtask. Otherwise, the reward is set to 0.

3.2 Offline Hierarchical Reinforcement Learning

The high-level policy π^h generates subgoals that guide the agent toward the final objective, while the low-level policy π^l learns actions needed to reach the subgoals specified by the high-level policy. This approach decomposes the overall problem into smaller subproblems, thus making the learning more tractable. Both the high-level and low-level policies are trained to maximize the following objective functions [34], employing Advantage Weighted Regression (AWR) [36] in the process:

$$J_{\pi^h} = \mathbb{E}\Big[\exp\big(\beta \cdot A^h\big)\ \log \pi^h\big(s_{t+c} \mid s_t, s_g\big)\Big] \tag{1}$$

$$J_{\pi^\ell} = \mathbb{E}\Big[\exp\big(\beta \cdot A^\ell\big)\ \log \pi^\ell\big(a_t \mid s_t, s_{t+c}\big)\Big] \tag{2}$$

By optimizing these objectives, the high-level policy is encouraged to behave similarly to the original data-collecting policy. Specifically, the probability of generating a subgoal s_{t+c} is weighted by $\exp\big(\beta \cdot A\big)$, where β is an inverse temperature parameter and A denotes the advantage estimated from the value function. Advantage measures how much better an action is compared to the policy's baseline expectation.

When training the high-level policy for subgoal generation, a tuple (s_t, s_{t+c}, s_g) is sampled from the offline dataset, and the policy is trained to produce s_{t+c} as the subgoal. Existing approaches frequently set c to a fixed number of steps or draw c from a geometric distribution [8], but these methods rely on a sequence of states in trajectory gathered by a suboptimal policy. Consequently, the optimal temporal distance between s_t and s_{t+c} can vary widely across trajectories, causing inconsistencies in subgoal generation and hindering high-level policy learning.

3.3 Temporal Distance Representation

Recent work [35] demonstrates that a temporal distance representation ψ can be learned from offline datasets for reinforcement learning. The core idea is to embed states into a latent space $\mathcal{H}$, where the Euclidean distance between any two embedded points approximates the optimal *temporal distance*, i.e. the minimal number of time steps required to reach one state from the other, between the corresponding states in the original state space $\mathcal{S}$. To learn the mapping $\psi : \mathcal{S} \to \mathcal{H}$, one approach is to utilize a goal-conditioned value learning scheme such as IQL [19]. Concretely, define

$$V(s,g) = -\big\|\psi(s) - \psi(g)\big\|_2 \tag{3}$$

By interpreting $-\|\psi(s) - \psi(g)\|$ as a value estimation, this approach minimizes the following objective via the expectile loss ℓ^2_τ:

$$\mathcal{L} = \mathbb{E}_{(s,s',g)\sim\mathcal{D}}\Big[\ell^2_\tau\Big(-\mathbf{1}\{s \neq g\} + \gamma \overline{V}(s',g) - V(s,g)\Big)\Big] \tag{4}$$

where $\ell^2_\tau(x) = |\tau - \mathbf{1}(x < 0)|x^2$ denotes the expectile loss [32], $\overline{V}$ denotes target value function [28].

4 Method

TDSG improves offline hierarchical reinforcement learning (HRL) by leveraging pre-trained Temporal Distance (TD) representations. TDSG enables the sampling of states located at specific temporal distances from the current state as subgoal candidates and compactly learns the consistent temporal relationship between the current state and TD-aware subgoals. Moreover, TDSG significantly reduces the training targets of the high-level policy by using only a small number (1% of the entire dataset) of anchor states that are spaced apart at consistent TD intervals while covering the entire dataset in the TD representation space. This approach eliminates redundant learning and enhances the quality of subgoal generation. Finally, TDSG improves the efficiency of low-level policy learning by utilizing an intrinsic reward based on how well the current state aligns with the direction toward the generated subgoal.

4.1 Temporal Distance-Aware Subgoal Sampling

We first pre-train a temporal distance (TD) representation ψ from the offline dataset by following Equation 4 introduced in [35]. The TD representation space $\mathcal{H}$ provides the temporal relationships between states, where the Euclidean distance represents the optimal temporal steps required to transition between states. Leveraging this property, we address the limitations of fixed c-step subgoal sampling, which often results in inconsistent temporal distances and suboptimal subgoal quality due to variations in trajectory quality, as illustrated in Figure 2. In the TD representation space, the temporal distance between two states s_i and s_j can be obtained by calculating their Euclidean distance as follows:

$$\text{dist}(s_i, s_j) = \|\psi(s_i) - \psi(s_j)\|_2 \tag{5}$$

As shown in Algorithm 1, using these computed distances, we can sample subgoal candidates s_{sub} located at the expected temporal distance from a given state s_t, even with suboptimal behavior trajectories. This TD-aware subgoal sampling can be utilized to train the high-level policy, providing more meaningful temporal relationships compared to fixed c-step sampling methods.

Algorithm 1 TD-aware Subgoal Sampling

1: **Input:** Offline dataset $\mathcal{D}$, pre-trained TD representation ψ, temporal-distance range (d_{min}, d_{max})
2: Sample a trajectory $\tau \in \mathcal{D}$
3: Sample a state $s_i \in \tau$
4: Sample a state $s_j \in \tau$ such that $i < j$, where $d = \|\psi(s_i) - \psi(s_j)\|_2$ and $d_{min} < d < d_{max}$
5: Set $s_t \leftarrow s_i$ and $s_{sub} \leftarrow s_j$

4.2 Subgoal Encoder and Decoder

While our subgoal encoder-decoder architecture follows a similar structure to Guider [39], there are several key differences. Guider operates on raw numerical states and uses handcrafted goal information, whereas our method works in a pre-trained temporal distance (TD) representation space, where both current and goal observations are embedded. This allows our model to learn compact subgoal embeddings z that not only encode goal-directed behavior but also respect the temporal structure of the task.

First, the Posterior Encoder q_ψ takes $(h_t, h_{\text{sub}}, h_g)$ as input and predicts the Gaussian distribution with mean and variance of the latent vector $\mathbf{z}$, where the current state h_t, the subgoal h_{sub} sampled in a TD-aware manner, and the final goal state h_g in the TD representation space. The Decoder p_θ, given $\mathbf{z}$, h_t, and h_g as inputs, is trained to reconstruct h_{sub}.

The training process follows the principles of a Variational Autoencoder (VAE) [18], where we maximize the Evidence Lower Bound (ELBO). To enhance training stability, we introduce a Prior Encoder p_ω, which takes only (h_t, h_g) as input and predicts the Gaussian distribution (mean and variance) of $\mathbf{z}$. We impose a KL regularization term against a prior distribution, using KL-Balance [11]. By minimizing the KL divergence between the distributions predicted by the Posterior Encoder q_ψ and the Prior Encoder p_ω, the training process becomes more stable. Through this structure, the Posterior Encoder q_ψ can generate a compact latent vector $\mathbf{z}$ that best represents h_{sub} when $(h_t, h_{\text{sub}}, h_g)$ are provided. Once trained, the latent space of z can be utilized effectively as a high-level action space to learn the high-level policy.

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim q_\psi}\big[\log p_\theta(h_{\text{sub}} \mid \mathbf{z}, h_t, h_g)\big] \\ - \beta\, D_{\text{KL}}\big(q_\psi(\mathbf{z} \mid h_t, h_{\text{sub}}, h_g) \,\|\, \rho_\omega(\mathbf{z} \mid h_t, h_g)\big)\end{aligned} \tag{6}$$

$$\begin{aligned}D_{\text{KL}}(q_\psi \,\|\, \rho_\omega) = \tau\, D_{\text{KL}}\big(\text{sg}(q_\psi) \,\|\, \rho_\omega\big) \\ + (1-\tau)\, D_{\text{KL}}\big(q_\psi \,\|\, \text{sg}(\rho_\omega)\big)\end{aligned} \tag{7}$$

$\text{sg}(\cdot)$ denotes the stop-gradient operation, β is a weighting parameter for the regularization, and τ is a KL balancing weight.

4.3 Anchor State Identification

Using all states in the offline dataset as subgoal candidates is inefficient. This approach includes unnecessary redundant states and low-quality states that are not suitable as subgoals, thereby reducing the learning efficiency of the high-level policy. To address this issue, we propose selecting anchor states that are evenly spaced in the TD representation space. This method allows the high-level policy to be trained more efficiently with a significantly smaller set of subgoal candidates.

Our Anchor State Selection Algorithm, as shown in Algorithm 2, ensures that anchor states in the TD representation space $\mathcal{H}$ are separated by a consistent temporal distance threshold δ. The algorithm proceeds as follows: 1) The first TD state h_1 from the offline dataset is selected as the first anchor state a_1. 2) For each subsequent state h_i in the dataset, the algorithm computes its Euclidean distance $\|h_i - a_j\|_2$ to the nearest anchor state a_j that has already been selected. 3) If the distance $\|h_i - a_j\|_2$ is within the threshold δ, h_i is assigned to the cluster of the nearest anchor state. If the distance exceeds δ, h_i is designated as a new anchor state. By repeating this process for all states in the dataset, the algorithm constructs a set of anchor states that cover the entire dataset while maintaining consistent spacing between them.

To further improve the representativeness of anchor states, we introduce a distance-based refinement step that considers the global distribution of TD embeddings. First, we compute the overall mean point h_{center} of all states in the TD space. For each cluster $\mathcal{A}_j$, we assign weights to its member states inversely proportional to their distance from h_{center}, emphasizing states closer to the global center. Using these weights, we compute a weighted mean μ_j for each

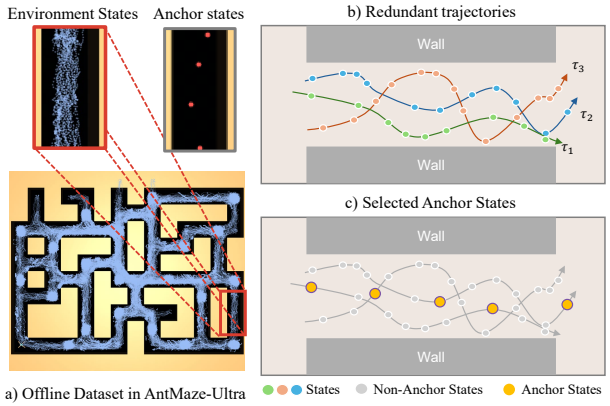

Figure 4: Visualization of the state distribution in the AntMaze-Ultra dataset and the selected anchor states. Choosing anchor states at regular TD intervals reduces redundancy and ensures a more consistent subgoal distribution.

cluster and select the state closest to μ_j as the final anchor state. This refinement ensures that anchor states are not only well-separated but also aligned with the global structure of the dataset. As a result, they serve as more generalizable and effective subgoal candidates in goal-conditioned tasks, ultimately enhancing the efficiency of high-level policy learning.

As depicted in Figure 3 c), each anchor state covers a local region within a distance threshold δ in TD representation space. By ensuring that the combined coverage of all anchor states spans the entire offline dataset, we mitigate subgoal redundancy while preserving overall coverage, thereby reducing the effective subgoal candidates to less than 1% of the dataset. Consequently, by combining them with the subgoal encoder–decoder, the high-level policy can learn from a substantially smaller set of potential subgoals without excessive redundancy, maintaining sufficient coverage and enhancing overall training efficiency.

4.4 Hierarchical RL with Anchor States

In this section, we describe how the previously described Subgoal Encoder-Decoder and the selected Anchor States are integrated into our hierarchical RL (HRL) framework, as illustrated in Figure 3(c).

4.4.1 High-Level Policy. The Posterior Encoder q_ϕ, introduced in Section 4.2, takes $(h_t, h_{\text{sub}}, h_g)$ as input and generates a high-level action z that can reconstruct the subgoal. Additionally, by leveraging the anchor state set $\mathcal{A}$ selected in Section 4.3, the high-level policy can be learned very efficiently. Instead of using the TD-aware sampled subgoal h_{sub} from the offline dataset, it is replaced with the nearest anchor state h^a_{sub}. Then, $(h_t, h^a_{\text{sub}}, h_g)$ is input into the Posterior Encoder q_ϕ, and the output latent vector z^a_{post} contains implicit information that can reconstruct h^a_{sub} via the subgoal decoder.

The high-level policy π^h, similar to the prior encoder, takes (h_t, h_g) as input and learns to model the distribution of z_{post} output by the Posterior Encoder. At the same time, it is optimized to generate the z_{post} with a higher advantage (leading to achieving the final goal and receiving a success reward) via Advantage Weighted Regression (AWR) [36]:

$$J_{\pi^h} = \mathbb{E}\Big[\exp\big(\beta \cdot A^h\big)\ \log \pi^h\big(z^a_{post} \mid h_t, h_g\big)\Big] \tag{8}$$

$$A^h = V(h^a_{\text{sub}}, h_g)\ -\ V(h_t, h_g) \tag{9}$$

where β is the inverse temperature parameter, and A^h denotes the advantage for high-level policy. By combining the anchor states with the subgoal encoder, we effectively reduce the high-level action space and achieve more efficient training. The high-level value function is optimized using the action-free variant [8] of IQL, analogous to Equation 4. The difference is that the high-level value function receives the TD state as input, while the reward term is still calculated in the original environment state. The loss is then defined as follows:

$$\mathcal{L}_{V^h} = \mathbb{E}\Big[\ell^2_\tau\Big(-\mathbf{1}\{s_t \neq s_g\}\ +\ \gamma\,\overline{V}^h(h_{t+1}, h_g)\ -\ V^h(h_t, h_g)\Big)\Big] \tag{10}$$

4.4.2 Low-Level Policy.

Intrinsic reward calcuation. To train the low-level policy π^ℓ, we utilize the concept of the intrinsic rewards[1] based on the TD representation:

$$r = \big\langle h_{t+1} - h_t,\ \vec{h}_{dir}\big\rangle,\ \vec{h}_{dir} = \frac{h_{\text{sub}} - h_t}{\|h_{\text{sub}} - h_t\|_2} \tag{11}$$

Algorithm 2 Anchor State Selection Algorithm

Input: TD states $H = \{h_1, h_2, \ldots, h_N\}$, distance threshold δ, Anchor set count $n \leftarrow 1$, Anchor cluster $\mathcal{A}_1 \leftarrow h_1$, Anchor set $\mathcal{A} \leftarrow h_1$.

for $i = 2$ to N **do**
  Compute distances: $d_j = \|h_i - a_j\|_2, \forall a_j \in \mathcal{A}$
  Find closest anchor set: $k = \arg\min_k d$
  if $d_k > \delta$ **then**
    $n \leftarrow n + 1$
    $\mathcal{A}_n \leftarrow h_i$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{h_i\}$
  else
    $\mathcal{A}_k \leftarrow \mathcal{A}_k \cup \{h_i\}$
  end if
end for
//Distance-based Selection
Compute center point: $h_{\text{center}} = \frac{1}{N}\sum_{h\in H} h$
Compute weight: $W(h_i) \leftarrow \frac{1}{\|h_i - h_{\text{center}}\|_2}, \forall h_i \in H$
Initialize anchor set: $\mathcal{A} \leftarrow \varnothing$
for each anchor cluster $\mathcal{A}_j$ **do**
  Compute weighted mean: $\mu_j = \frac{\sum_{h_i\in\mathcal{A}_j} W(h_i)\cdot h_i}{\sum_{h_i\in\mathcal{A}_j}(h_i)}$
  Find closest state: $k = \arg\min_{h_i\in\mathcal{A}_j} \|h_i - \mu_j\|_2$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{h_k\}$
end for
Output: Anchor set $\mathcal{A}$

[1] In [35], the reward is defined as $r(s, s', \vec{h}) = \langle \psi(s') - \psi(s), \vec{h}\rangle$ with $\vec{h} \sim$ uniform and $\vec{h} \in \mathcal{H} : \|\vec{h}\|_2 = 1$.

Algorithm 3 Training RL algorithm with Anchor States

Input: offline dataset $\mathcal{D}$
Initialize: TD representation ψ, Subgoal Encoder & Decoder, high-level policy, low-level policy

// TD representation
for each gradient step **do**
  Sample $(s_t, s_{t+1}, s_g) \sim \mathcal{D}$
  Update Temporal distance representation ψ on (s_t, s_{t+1}, s_g) (Eq. 4)
end for

// Subgoal Encoder, Decoder
for each gradient step **do**
  Sample $(h_t, h_{sub}, h_g) \sim \psi(\mathcal{D})$
  Update Subgoal enc & dec on (h_t, h_{sub}, h_g) (Eq. 7)
end for

// Anchor state selection
Anchor States $\mathcal{A} \leftarrow$ clustering with TD states $\mathcal{H}$.

// RL algorithm
for each gradient step **do**
  Sample $(s, a, s') \sim \mathcal{D}, (h_t, h_{sub}, h_g) \sim \psi(\mathcal{D}), z \sim \pi^h$
  $h^a_{sub} \leftarrow h_{sub}$ // Replace subgoal with nearest anchor state
  Update High-level Policy on (h_t, h^a_{sub}, h_g) (Eq. 8)
  Update Low-level Policy on (s, a, s', z) (Eq. 13)
end for

where $(s_t, s_{t+1}, s_{\text{sub}}) \sim \mathcal{D}$, $h = \psi(s)$. The $\langle A, B \rangle$ operator means the inner product between A and B. This reward design encourages the agent to move reliably toward the subgoal direction, even in sparse-reward environments, by aligning states with the subgoal direction $\vec{h}_{dir}$ in the TD space. Unlike prior methods that rely solely on subgoals sampled from offline datasets, our approach leverages subgoals generated by the high-level policy, thereby mitigating distribution shift issues that arise when transitioning from training to test time.

Subgoal sampling strategy for the low-level policy. We introduce a direction vector $\vec{h}_{dir}$ between the current state h_t and the reconstructed subgoal $\hat{h}^{\text{high}}$. The reconstructed subgoal is defined as

$$h_{\text{sub}} \leftarrow \hat{h}^{\text{high}} \sim p_\theta\big(z^{\text{high}}, h_t, h_g\big), \quad z^{\text{high}} \sim \pi^h(\cdot) \tag{12}$$

where we use reconstructed subgoal $\hat{h}^{\text{high}}$ to calculate subgoal direction vector $\vec{h}_{dir}$. We then use this direction vector $\vec{h}_{dir}$ as the subgoal of the low-level policy. Specifically, for each transition (s, a, s'), if movement from $\psi(s)$ to $\psi(s')$ is better aligned with $\hat{h}^{\text{high}}$, then the policy is more likely to select action a. The low-level policy also follows AWR, maximizing:

$$J_{\pi^\ell} = \mathbb{E}\Big[\exp\big(\beta \cdot A^\ell\big)\ \log \pi^\ell\big(a_t \mid s_t, \vec{h}_{dir}\big)\Big] \tag{13}$$

$$A^\ell = Q\big(s_t, a, \vec{h}_{\text{dir}}\big) \ - \ V\big(s_t, \vec{h}_{\text{dir}}\big) \tag{14}$$

where β is the inverse temperature parameter, and A^ℓ denotes the advantage for the low-level policy. The low-level value and Q function are updated via the goal-conditioned IQL scheme [19] as follows:

$$L_{V^\ell} = \mathbb{E}\Big[L_2^\tau\big(Q^\ell(s_t, a_t, \vec{h}_{\text{dir}}) - V^\ell(s_t, \vec{h}_{\text{dir}})\big)\Big] \tag{15}$$

$$L_{Q^\ell} = \mathbb{E}\Big[\big(r(s_t, s_{t+1}, \vec{h}_{\text{dir}}) \ + \ \gamma\, V^\ell(s_{t+1}, \vec{h}_{\text{dir}}) - Q^\ell(s_t, a_t, \vec{h}_{\text{dir}})\big)^2\Big] \tag{16}$$

Both networks take the direction vector $\vec{h}_{dir}$ as subgoal, and The reward $r(s_t, s_{t+1}, \vec{h}_{\text{dir}})$ is computed as in Equation 11.

Figure 5: The benchmark environments for long-horizon, sparse rewards setting, goal-conditioned reinforcement learning. These include numeric tasks and visual tasks.

5 Experiments

We conduct a series of experiments to address two primary objectives. First, we evaluate the applicability and effectiveness of our proposed TDSG framework in both numeric and visual environments featuring long horizons and sparse rewards. Second, we examine how the core components of TDSG influence performance, thereby gaining insights into the method's key characteristics. Figure 5 depicts the environments used to evaluate our approach on long-horizon, goal-conditioned RL tasks.

5.1 Experimental Setup

Antmaze. We use the 'antmaze-large-diverse-v2' and 'antmaze-large-play-v2' datasets from D4RL [7]. For AntMaze-Ultra, we employ the 'antmaze-ultra-diverse-v0' and 'antmaze-ultra-play-v0' datasets introduced by Jiang et al. [15], which feature mazes roughly twice the size of those in the large environments. The objective is to operate an 8-DoF Ant robot to reach a designated goal position from its initial position [4, 7, 15, 40]. The observation space is 29-dimensional, consisting of 2D coordinates and 27 joint-related features. While the original AntMaze environment used only xy coordinates as the goal space, we follow the approach of [34] by concatenating xy coordinates with proprioceptive information to construct the goal observation. A reward of 1 is provided upon reaching the goal. **Visual-antmaze** The 2D coordinate components are replaced with top-down RGB images of size $(64 \times 64 \times 3)$, while the 27 joint features remain preserved [34].

Kitchen. We employ the 'kitchen-partial-v0' and 'kitchen-mixed-v0' datasets from D4RL [7]. Manipulation domain where a 9-DoF Franka robot arm must complete four subtasks. In 'kitchen-partial-v0', the agent must complete four subtasks: opening the microwave, moving the kettle, turning on a light switch, and sliding a cabinet door. Only a small fraction of successful trajectories exists (approximately 3% share the same subtask order as in evaluation, and

Task	GC-IQL	HILP	HILP-Plan	HGCBC	HIQL	Ours (TDSG)
antmaze-large-diverse	57.5 ± 9.8	46.6 ± 4.7	73.3 ± 8.6	53.3 ± 18.0	**85.0** ± 8.3	**85.0** ± 6.9
antmaze-large-play	53.3 ± 18.1	46.6 ± 2.7	60.0 ± 14.4	61.2 ± 31.4	83.3 ± 7.2	**87.5** ± 7.3
antmaze-ultra-diverse	24.1 ± 18.5	20.0 ± 14.1	60.8 ± 8.3	33.3 ± 9.8	54.1 ± 9.9	**68.3** ± 7.9
antmaze-ultra-play	18.3 ± 23.9	15.8 ± 1.9	46.6 ± 17.6	42.4 ± 19.3	47.5 ± 18.3	**65.0** ± 5.7
kitchen-partial	35.8 ± 4.9	64.1 ± 2.9	59.7 ± 4.7	35.8 ± 17.0	57.9 ± 5.2	**68.3** ± 2.2
kitchen-mixed	45.4 ± 11.6	60.8 ± 5.2	60.4 ± 7.5	41.4 ± 11.0	**63.5** ± 9.7	**65.0** ± 3.9
visual-antmaze-large	76.6 ± 7.2	22.5 ± 14.2	56.6 ± 19.8	51.6 ± 2.3	78.8 ± 8.3	**84.1** ± 6.3
visual-kitchen-mixed	16.6 ± 11.4	38.5 ± 16.3	**46.0** ± 17.4	18.0 ± 11.4	25.8 ± 5.8	**45.6** ± 5.0
visual-kitchen-partial	47.6 ± 27.9	49.1 ± 15.9	54.9 ± 6.8	48.7 ± 8.4	54.5 ± 21.5	**57.4** ± 3.4
visual-fetch-reach	96.7 ± 3.2	96.6 ± 2.7	96.6 ± 4.7	76.6 ± 27.0	94.3 ± 4.0	**99.1** ± 1.7
visual-fetch-pick and place	37.8 ± 4.8	43.7 ± 18.7	**52.4** ± 28.0	35.8 ± 8.7	41.1 ± 8.0	**53.3** ± 8.1
Average Returns	46.3	45.8	60.6	45.2	62.3	**70.7**

Table 1: Performance comparison of TDSG against baseline offline goal-conditioned RL models across a range of benchmark environments (AntMaze, Kitchen, Fetch, and their visual versions). Baseline models include GC-IQL, HILP, HILP-Plan, HGCBC, and HIQL. Each method is evaluated using four random seeds, with average normalized returns and standard deviations reported.

around 1% fully solve the task). In 'kitchen-mixed-v0', the agent must accomplish four subtasks (microwave opening, kettle movement, light switch, and bottom left burner), but the dataset contains no trajectory achieving all four—at most three are completed. The observation space consists of 30 joint and object-related features. At test time, the agent receives a reward of 1 for each subtask completed. **Visual-kitchen** For pixel-based Kitchen experiments, we replace numeric joint and object observations with $64 \times 64 \times 3$ images [35]. We render from the same partial or mixed data used in the numeric Kitchen environment.

Visual-Fetch. The robot is a 7-DoF Fetch Mobile Manipulator with a two-fingered parallel gripper. The objective is to move either an end-effector (Fetch-Reach) or an object (Fetch-PickAndPlace) to a designated position, starting randomly [37]. For the visual variant (Visual-Fetch), we provide a $64 \times 64 \times 3$ front-view RGB image and remove any numeric object or joint features. The dataset is taken from the "expert" demonstrations of [45]. The entire trajectory data comprises 100k for Fetch-Reach and 2M for Fetch-PickAndPlace; however, we only use 1M for the latter in training.

5.2 Baselines

We compare TDSG against several established offline RL baselines. In the goal-conditioned offline RL category, we include "GC-IQL", IQL [19], and the hierarchical IQL (HIQL) [34], which generates subgoals in a latent space, along with hierarchical goal-conditioned behavioral cloning (HGCBC) [27], a method utilizing two-level policies. Additionally, TDSG is evaluated against HILP, which incorporates temporal-distance representation, and HILP-Plan [35], a variant that recursively computes midpoints to perform planning towards the goal. Each experiment is run with four different random seeds, and the main results are reported along with standard deviations for each algorithm's performance metric.

5.3 Main Results

We first summarize the overall experimental outcomes to evaluate TDSG across both numeric and visual environments presented in Figure 5 with other baseline models. Table 1 presents the main results, where the proposed method outperforms the baselines in terms of average returns, showing a performance improvement of over 13% compared to the existing algorithms. Notably, the AntMaze environment involves much longer task horizons compared to Kitchen or Fetch, and our approach demonstrates particular effectiveness in long-horizon scenarios. Additionally, for the visual environments, TDSG maintains an advantage over prior methods, highlighting its capability for high-dimensional image inputs. Overall, these results confirm that TDSG effectively leverages temporal distance representations and anchor states to generate high-quality subgoals, even when data quality is variable.

6 Ablation Studies

We further conduct an ablation study to gain insights into how each component of TDSG influences performance. Unless otherwise noted, the ablation experiments are carried out on two representative datasets: *AntMaze-ultra-diverse* and *Kitchen-mixed*.

6.1 Varying Temporal Distance Between Subgoals

Next, we vary the distance d between the current state and the next subgoal in our TD-aware subgoal selection. Table 2 reports the results. If d is too large, the low-level policy has to learn extended action sequences to reach the subgoal, increasing the difficulty of training. Conversely, a too-small d forces the high-level policy to create many subgoals before reaching the final goal, which also raises learning complexity as mentioned in previous HRL work [23]. Thus, an intermediate range of d tends to yield the best performance, aligning with the intuition that both edge cases are suboptimal.

	Temporal Distance between Subgoals	Results
Antmaze-ultra-diverse	5	55.8 ± 19.1
	10	65.8 ± 8.7
	15	**68.3** ± 7.9
	20	51.6 ± 7.9
Kitchen-mixed	20	61.6 ± 6.5
	30	61.4 ± 4.1
	40	**65.0** ± 3.9
	50	50.2 ± 5.1

Table 2: Evaluation of various temporal distances d for TD-aware subgoal sampling. Moderate distances tend to yield the best results, balancing between efficiency and effectiveness.

6.2 Effectiveness of Proposed Components

To evaluate the impact of our proposed techniques, we examine different configurations that enable or disable (i) TD-aware subgoal selection, (ii) anchor state usage, and (iii) subgoal representation learning. As summarized in Table 3, the highest returns are obtained when all three methods are enabled. Removing any one component leads to a noticeable performance drop, confirming that (i) consistent temporal distances, (ii) compact anchor states, and (iii) subgoal embedding each provide synergistic benefits.

TD-aware Subgoal Sampling	Anchor State	Subgoal Enc&Dec	Antmaze Ultra diverse	Kitchen Mixed
✘	✘	✘	46.6 ± 7.2	45.7 ± 5.6
✔	✘	✘	52.5 ± 16.4	47.5 ± 7.5
✘	✔	✘	51.6 ± 16.6	48.5 ± 3.6
✘	✘	✔	47.5 ± 6.8	51.2 ± 6.4
✘	✔	✔	56.6 ± 16.7	52.5 ± 6.8
✔	✔	✘	59.1 ± 5.0	49.7 ± 3.2
✔	✘	✔	61.6 ± 12.6	51.4 ± 4.9
✔	✔	✔	**68.3** ± 7.9	**65.0** ± 3.9

Table 3: Analysis of ablation conditions for TDSG. The results demonstrate that combining 1) TD-aware subgoal sampling, 2) Anchor states, and 3) Subgoal representation yields the highest performance, whereas omitting any one component degrades performance.

6.3 Sampling Strategies for Direction Vector

This experiment investigates how variations in the sampling strategies for direction vectors impact the low-level policy's learning efficiency and generalization. Specifically, we compare three approaches to generating the direction vector $\vec{h}_{dir}$:

- **Uniform Sampling** Sample $\vec{h}_{dir}$ from a uniform distribution on the unit sphere, i.e.,
$$\vec{h}_{dir} \sim \text{Uniform} \quad \text{where} \quad \|\vec{h}_{dir}\|_2 = 1$$
- **Dataset Sampling** Sample (h_t, h_{sub}) from the offline dataset $\mathcal{D}$, then compute the direction vector as:
$$\vec{h}_{dir} = \frac{h_{sub} - h_t}{\| h_{sub} - h_t \|_2} \tag{17}$$
- **Policy π^{high} Sampling** Generate a subgoal z^{high} from the high-level policy, reconstruct $\hat{h}^{\text{high}} \sim p_\theta(z^{\text{high}})$, and compute:
$$\vec{h}_{dir} = \frac{\hat{h}^{\text{high}} - h_t}{\| \hat{h}^{\text{high}} - h_t \|_2} \tag{18}$$

Table 4 shows that using the policy π^h distribution leads to the best performance, by avoiding discrepancy between training and inference phases. In contrast, random or dataset-generated vectors degrade performance.

	Uniform Sampling	Dataset Sampling	Policy π^{high} (ours) Sampling
Antmaze ultra-diverse	54.1 ± 14.4	61.6 ± 8.3	**68.3** ± 7.9
Kitchen mixed	55.4 ± 2.5	36.6 ± 13.8	**65.0** ± 3.9

Table 4: Comparison of the effects of different sampling strategies for direction-vector on low-level policy training.

6.4 Comparison of Anchor State Selection Algorithms

We evaluate different anchor-state selection algorithms, including k-means [16] (1x and 2x centroids), VQ-VAE codebooks [41], and our proposed approach (TDSG). Each algorithm selects anchor states from the TD representation state $\psi(s)$ rather than the original environment state s. We keep the total number of anchor states consistent (e.g., 1500 for AntMaze-ultra-diverse and 700 for Kitchen-mixed) across all methods, except for the "2x" variant which doubles the centroids. Experimentally, as shown in Figure 6, our approach exhibits higher returns than these baselines, indicating that uniform spacing can be more effective than prior clustering-based methods.

7 Implementation Details

7.1 Goal Sampling

We train temporal distance representation, subgoal encoder, decoder, and RL models using different goal-sampling distributions. For the temporal distance representation, we sample s_g either from a geometric distribution over future states within the same trajectory (with probability 0.625) or uniformly from the entire dataset (with probability 0.375) [35]. For the low-level policy, we use the direction vector $\vec{h}_{\text{dir}}$ as the subgoal. After the high-level policy generates a subgoal embedding z^{high}, the subgoal decoder p_θ reconstructs $\hat{h}^{\text{high}}$ in accordance with Equation 12. Subsequently, $\vec{h}_{\text{dir}}$ is computed

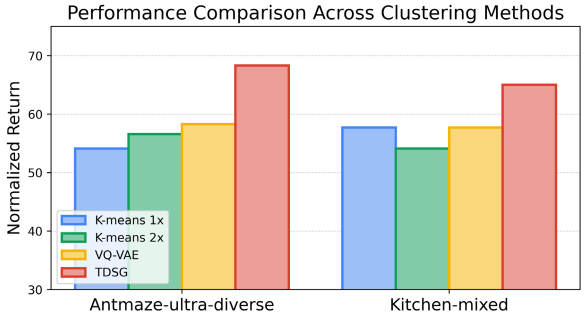

Figure 6: Normalized return comparison across anchor state selection methods (K-means 1× and 2× centroids, VQ-VAE codebooks, and our TDSG approach) in AntMaze-ultra-diverse and Kitchen-mixed.

following Equation 18. To enhance the generalization capability of the low-level value and Q functions across diverse directions within the offline dataset, we compute the direction vector $\vec{h}_{\text{dir}}$ by sampling s_g uniformly at random from dataset and replacing $\hat{h}^{high}$ with $\psi(s_g)$. For the high-level policy, we select s_g from either random states, future states, or the current state with probabilities 0.3, 0.5, and 0.2, respectively, following [8], and subsequently, we adopt TD state $\psi(s_g)$ as the goal h_g. Specifically, we use Geom$(1-\gamma)$ for sampling future states in a trajectory, and a uniform distribution over the offline dataset for choosing random states.

7.2 Network Architecture

The high-level policy uses only a single value function, whereas the low-level policy uses both a value function and a Q function. The value function, Q function, policy networks, subgoal encoder-decoder, and temporal distance representation network all share a similar design: three fully connected layers of width 512, with layer normalization and GELU activation. For the value function, we also maintain a target network initialized identically and updated via a soft-update mechanism. For the visual setting, we adopt an Impala CNN architecture [6] as our image encoder. Baseline models are configured with the same fully connected layer dimensions and the same image encoder, ensuring consistency across all experiments.

7.3 Hyperparameters

Table 5 summarizes the hyperparameters used across experiments. We include the temporal distance representation expectile τ_{td}, high-level value function expectile τ_{high}, low-level value function expectile τ_{low}, high-level temperature β_{high}, and low-level temperature β_{low}. The values of these parameters (τ_{td}, τ_{high}, τ_{low}, β_{high}, β_{low}) for each environment are listed as follows. AntMaze-Large, AntMaze-Ultra, and Visual-AntMaze-Large use (0.9, 0.7, 0.9, 5, 10); Kitchen-partial and Kitchen-mixed use (0.95, 0.7, 0.9, 5, 10); Visual-Kitchen-partial and Visual-Kitchen-mixed use (0.9, 0.7, 0.7, 0.3, 15); and Visual-Fetch-Reach, Visual-Fetch-PickAndPlace adopt (0.95, 0.7, 0.7, 1, 10). All other hyperparameters remain consistent across tasks.

	Value
Batch size	1024 (numeric), 256 (visual)
Learning rate	3e-4
Target network soft & update coeff.	5e-3
Optimizer	Adam
Temporal distance repr. dim	32
Subgoal embedding dim	10
Subgoal weighting parameter β	0.1
Subgoal kl balancing parameter τ	0.8
Training steps	
Temporal distance repr.	200,000
Subgoal encoder & decoder	200,000
RL Algorithm	250,000 (kitchen)
	500,000 (others)

Table 5: Hyperparameters used for the experiments.

8 Conclusion

We presented Temporal Distance-Aware Subgoal Generation (TDSG), a framework leveraging temporal distance representations to enhance offline hierarchical reinforcement learning for long-horizon and sparse-reward tasks. By selecting a compact set of anchor states that uniformly cover the dataset in the temporal distance representation space, TDSG minimizes redundancy, eliminates low-quality subgoals, and maintains meaningful temporal relationships. This ensures both efficient high-level policy training and robust low-level policy alignment with generated subgoals. Experimental results show that TDSG achieves consistent performance improvements over prior methods across diverse environments.

Acknowledgments

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00348376) and the Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00438686, No. RS-2022-II221045, No. RS-2025-02218768, and No. RS-2019-II190421).

9 GenAI Usage Disclosure

In accordance with ACM's Authorship Policy, we declare that no generative AI tools were utilized in the research process.

References

[1] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. 2021. OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning. arXiv:2010.13611 [cs.LG] https://arxiv.org/abs/2010.13611

[2] Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. 2025. Graph-Assisted Stitching for Offline Hierarchical Reinforcement Learning. *arXiv preprint arXiv:2506.07744* (2025).

[3] Andrew G Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems* 13 (2003), 341–379.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[5] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. 2021. Goal-conditioned reinforcement learning with imagined subgoals. In *International conference on machine learning*. PMLR, 1430–1440.

[6] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. Impala:

Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*. PMLR, 1407–1416.

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

[8] Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. 2023. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning*. PMLR, 11321–11339.

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

[10] Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, G. Tucker, and Sergey Levine. 2018. Learning to Walk via Deep Reinforcement Learning. *ArXiv* abs/1812.11103 (2018). https://api.semanticscholar.org/CorpusID:57189150

[11] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. 2020. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193* (2020).

[12] Josiah P Hanna, Scott Niekum, and Peter Stone. 2021. Importance sampling in reinforcement learning with an estimated behavior policy. *Machine Learning* 110, 6 (2021), 1267–1317.

[13] Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. 2021. Successor Feature Landmarks for Long-Horizon Goal-Conditioned Reinforcement Learning. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:244345800

[14] Zixuan Huang, Yating Lin, Fan Yang, and Dmitry Berenson. 2024. Subgoal diffuser: Coarse-to-fine subgoal generation to guide model predictive control for robot manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 16489–16495.

[15] Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. 2022. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291* (2022).

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[17] Junsu Kim, Younggyo Seo, and Jinwoo Shin. 2021. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in neural information processing systems* 34 (2021), 28336–28349.

[18] Diederik P Kingma. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[19] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. 2021. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*. PMLR, 5774–5783.

[20] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

[21] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[22] Sascha Lange, Thomas Gabel, and Martin Riedmiller. 2012. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*. Springer, 45–73.

[23] Seungjae Lee, Jigang Kim, Inkyu Jang, and H. Jin Kim. 2022. DHRL: A Graph-Based Approach for Long-Horizon and Sparse Hierarchical Reinforcement Learning. *ArXiv* abs/2210.05150 (2022). https://api.semanticscholar.org/CorpusID:252815462

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

[25] Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *ArXiv* abs/2005.01643 (2020). https://api.semanticscholar.org/CorpusID:218486979

[26] Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. 2023. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*. PMLR, 20035–20064.

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

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs.LG]

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533. https://api.semanticscholar.org/CorpusID:205242740

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

[31] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2021. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets. arXiv:2006.09359 [cs.LG]

[32] Whitney K Newey and James L Powell. 1987. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society* (1987), 819–847.

[33] Jongchan Park, Seungjun Oh, and Yusung Kim. 2024. Novelty-aware Graph Traversal and Expansion for Hierarchical Reinforcement Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 1846–1855.

[34] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. 2023. Hiql: Offline goal-conditioned rl with latent states as actions. *arXiv preprint arXiv:2307.11949* (2023).

[35] Seohong Park, Tobias Kreiman, and Sergey Levine. 2024. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567* (2024).

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

[37] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. 2018. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. arXiv:arXiv:1802.09464

[38] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2019. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588 (2019), 604 – 609. https://api.semanticscholar.org/CorpusID:208158225

[39] Wonchul Shin and Yusung Kim. 2023. Guide to control: offline hierarchical reinforcement learning using subgoal generation for long-horizon and sparse-reward tasks. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 4217–4225.

[40] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[41] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2018. Neural Discrete Representation Learning. arXiv:1711.00937 [cs.LG] https://arxiv.org/abs/1711.00937

[42] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and P. Abbeel. 2022. DayDreamer: World Models for Physical Robot Learning. In *Conference on Robot Learning*. https://api.semanticscholar.org/CorpusID:250088882

[43] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[44] Mengjiao Yang and Ofir Nachum. 2021. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*. PMLR, 11784–11794.

[45] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. 2022. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv preprint arXiv:2202.04478* (2022).

[46] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems* 33 (2020), 14129–14142.