Extendable Planning via Multiscale Diffusion

Chang Chen[*2], **Hany Hamed**[*1], **Doojin Baek**[1], **Taegu Kang**[1], **Samyeul Noh**[3], **Yoshua Bengio**[4], **Sungjin Ahn**[1,5]

[1]KAIST
[2]Rutgers University
[3]ETRI
[4]Mila
[5]NYU

Abstract

Long-horizon planning is crucial in complex environments, but diffusion-based planners like Diffuser are limited by the trajectory lengths observed during training. This creates a dilemma: long trajectories are needed for effective planning, yet they degrade model performance. In this paper, we introduce this *extendable long-horizon planning* challenge and propose a two-phase solution. First, Progressive Trajectory Extension incrementally constructs longer trajectories through multi-round compositional stitching. Second, the Hierarchical Multiscale Diffuser enables efficient training and inference over long horizons by reasoning across temporal scales. To avoid the need for multiple separate models, we propose Adaptive Plan Pondering and the Recursive HM-Diffuser, which unify hierarchical planning within a single model. Experiments show our approach yields strong performance gains, advancing scalable and efficient decision-making over long-horizons.

Introduction

The ability to plan over long horizons via a learned world model (Hamrick et al. 2020; Mattar and Lengyel 2022) enables agents to pursue long-term goals, even in environments with sparse rewards (Silver et al. 2016; Hafner et al. 2019; Hansen, Wang, and Su 2022). However, constructing effective world models (Ha and Schmidhuber 2018) for such planning remains a challenge. Traditional planning approaches based on autoregressive forward models are particularly susceptible to compounding errors (Lambert, Pister, and Calandra 2022; Bachmann and Nagarajan 2024). In offline reinforcement learning, the Diffuser framework (Janner et al. 2022; Ajay et al. 2022) offers a promising alternative by leveraging diffusion models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020) to bypass explicit forward dynamics. Instead of predicting states sequentially, Diffuser generates entire trajectories holistically thereby mitigating compounding errors and improving planning accuracy, particularly for long-horizon planning.

A fundamental but underappreciated limitation of applying diffusion models to long-horizon planning is that they cannot plan beyond the trajectory lengths seen during training. As a result, modeling trajectories longer than those seen during training becomes difficult. However, many real-world applications require the ability to plan beyond the observed sequence lengths. In contrast, forward model-based planning does not suffer from this problem as it can extend to previously unseen horizons by rolling out longer sequences. Therefore, it is crucial for diffusion-based planner to address this issue.

While one solution is to collect longer trajectories, this quickly becomes impractical. For example, enabling a robot to plan over week- or month-long horizons would require uninterrupted trajectories of that length—an increasingly infeasible task as the horizon grows. Even if such data were available, training a Diffuser model on trajectories of that scale remains challenging. In fact, planning performance has been shown to degrade on extended sequences in existing diffusion-based approaches (Chen et al. 2024b). This presents a fundamental dilemma: *long trajectories are needed for effective planning, yet they undermine the performance of the Diffuser itself.*

In this paper, we propose a simple two-phase solution to this fundamental dilemma, which we term the *extendable long-horizon planning* challenge. To address this problem, we propose both a mechanism to generate long trajectories from short ones and a model capable of learning effectively from them.

First, we introduce Progressive Trajectory Extension (PTE)—a method that incrementally extends short training trajectories into significantly longer ones. While trajectory stitching is not a new concept, PTE differs from traditional approaches by not merely augmenting data, but by actively extending trajectories through multiple rounds of compositional stitching of previously extended sequences. This multi-round process enables the construction of extremely long trajectories.

Second, we present the Hierarchical Multiscale Diffuser (HM-Diffuser), which enables diffusion models to be trained effectively on these long trajectories by introducing multiple temporal scales. In particular, to overcome limitations of existing hierarchical Diffuser frameworks—which require separate models for each hierarchy level—we further propose the Adaptive Plan Pondering and the Recursive HM-

[*]These authors contributed equally.

Diffuser in the HM-Diffuser framework. These techniques consolidate multiple hierarchical layers into a single model capable of recursively reasoning across temporal scales.

Our experiments demonstrate that the integration of PTE and HM-Diffuser yields non-trivial benefits. The combined approach significantly enhances performance across a range of planning tasks, underscoring its potential to advance scalable and efficient long-horizon decision-making.

The key contributions of this paper are as follows. First and foremost, this paper identifies and formulates a previously underexplored challenge: extendable long-horizon planning in Diffuser-based models. Second, as a solution to this novel challenge, we propose a new unified two-phase framework that integrates a process for generating extended trajectories from short demonstrations, and a learning framework that can leverage them effectively. While each component draws from prior work on trajectory stitching and hierarchical diffusion, their integration to tackle this specific challenge is novel. Third, we present PTE, a multi-round stitching method that generates explicitly longer trajectories through iterative compositional extension. Fourth, we introduce the Recursive Hierarchical Diffuser, a model that enables efficient long-horizon planning through multiscale hierarchical reasoning. Finally, we introduce the Plan Extendable Trajectory Suite (PETS) benchmark.

Preliminaries

Diffusion models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020), inspired by the modeling of diffusion processes in statistical physics, are latent variable models with the following generative process

$$p_\theta(\mathbf{x}_0) = \int p(\mathbf{x}_M) \prod_{m=1}^{M} p_\theta(\mathbf{x}_{m-1} \mid \mathbf{x}_m)\, \mathrm{d}\mathbf{x}_{1:M}. \quad (1)$$

Here, $\mathbf{x}_0$ is a datapoint and $\mathbf{x}_{1:M}$ are latent variables of the same dimensionality as $\mathbf{x}_0$. A diffusion model consists of two core processes: the reverse process and the forward process. The reverse process is defined as

$$p_\theta(\mathbf{x}_{m-1}|\mathbf{x}_m) := \mathcal{N}(\mathbf{x}_{m-1}|\boldsymbol{\mu}_\theta(\mathbf{x}_m, m), \sigma_m \mathbf{I})\ . \quad (2)$$

This process transforms a noise sample $\mathbf{x}_M \sim p(\mathbf{x}_M) = \mathcal{N}(0, \mathbf{I})$ into an observation $\mathbf{x}_0$ via a sequence of denoising steps $p_\theta(\mathbf{x}_{m-1}|\mathbf{x}_m)$ for $m = M, \ldots, 1$. In the reverse direction, the forward process defines the approximate posterior $q(\mathbf{x}_{1:M}|\mathbf{x}_0) = \prod_{m=0}^{M-1} q(\mathbf{x}_{m+1}|\mathbf{x}_m)$ via the forward transitions:

$$q(\mathbf{x}_{m+1}|\mathbf{x}_m) := \mathcal{N}(\mathbf{x}_{m+1}; \sqrt{\alpha_m}\mathbf{x}_m, (1-\alpha_m)\mathbf{I})\ . \quad (3)$$

The forward process iteratively applies this transition from $m = 0, ..., M-1$ according to a predefined variance schedule $\alpha_1, \ldots, \alpha_M$ and gradually transforms the observation $\mathbf{x}_0$ into noise $\mathcal{N}(0, \mathbf{I})$ as $m \to M$ for a sufficiently large M. Unlike the reverse process involving learnable model parameters θ, the forward process is predefined. Learning the parameter θ of the reverse process is done by optimizing the variational lower bound on the log likelihood $\log p_\theta(\mathbf{x}_0)$. Ho, Jain, and Abbeel (2020) demonstrated that this can be achieved by minimizing the following simple denoising objective: $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0, m, \boldsymbol{\epsilon}} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_m, m)\|^2 \right]$.

Planning with Diffusion. A prominent approach to planning via diffusion has recently gained increasing attention, with Diffuser (Janner et al. 2022) being one of the most well-known methods. Diffuser formulates planning as a trajectory generation problem using diffusion models, first training a diffusion model $p_\theta(\boldsymbol{\tau})$ on offline trajectory data $\boldsymbol{\tau} = (s_0, a_0, \ldots, s_T, a_T)$ and then guiding sampling towrad high-return trajectories using a classifier guided approach (Dhariwal and Nichol 2021). This guidance model, $p_\phi(\mathbf{y}|\boldsymbol{\tau}) \propto \exp(G_\phi(\mathbf{x}))$, predicts trajectory returns, modifying the sampling distribution to $\tilde{p}_\theta(\boldsymbol{\tau}) \propto p_\theta(\boldsymbol{\tau}) \exp(\mathcal{J}_\phi(\boldsymbol{\tau}))$, which biases the denoising process toward optimal trajectories at test time.

Proposed Method

Our goal is to enable planning far beyond the horizon lengths available in the original dataset. To this end, we propose a two-phase approach. First, we extend the collected trajectories to significantly longer horizons via a Progressive Trajectory Extension (PTE) procedure. Second, we train a hierarchical diffusion-based planner on these extended trajectories, which we call the Hierarchical Multiscale Diffuser (HM-Diffuser). This planner is specifically designed to handle multiple temporal scales efficiently, significantly improving long-horizon planning capabilities.

Progressive Trajectory Extension

Progressive Trajectory Extension (PTE) is an iterative process that stitches together shorter trajectories into longer ones through successive rounds. Before running PTE, we train a few key components needed for trajectory stitching: (1) a diffusion model called the "stitcher" for bridging trajectories, (2) an inverse dynamics model for inferring actions, and (3) a reward model for estimating rewards. The stitcher $p_\theta^{\text{stitcher}}(\boldsymbol{\tau})$ is trained on the base dataset D_0, composed of short trajectories collected from the environment, to generate trajectory segments from a given start state. The inverse dynamics model f_a predicts an action a_t from consecutive states (s_t, s_{t+1}), and the reward model f_r estimates the reward r_t for a state-action pair (s_t, a_t). These components lay the groundwork for extending trajectories.

For each round of PTE, the algorithm takes as input a source set $\mathcal{S}$ of trajectories to extend and a target set $\mathcal{T}$ of trajectories providing potential continuations. How $\mathcal{S}$ and $\mathcal{T}$ are constructed plays a crucial role in determining the extension behavior of the resulting trajectories. In Section , we describe several effective strategies for organizing these sets to guide trajectory growth. As illustrated in Figure 1, each PTE round performs a series of stitching iterations as follows:

(i) Sampling a source trajectory, a bridge, and target candidates. We begin by randomly selecting a **source** trajectory $\boldsymbol{\tau}_s \in \mathcal{S}$ for extension. Next, we generate a **bridge** by conditioning the stitcher model on the last state of the source, $\boldsymbol{\tau}_b \sim p_\theta^{\text{stitcher}}(\boldsymbol{\tau}|s_0 = s_{\boldsymbol{\tau}_s}^{\text{last}})$. We use the reward model f_r and the inverse dynamics model f_a to infer rewards and actions for the bridge. Finally, we sample a batch of c target **candidate** trajectories $(\boldsymbol{\tau}_t^{(1)}, \boldsymbol{\tau}_t^{(2)}, \ldots, \boldsymbol{\tau}_t^{(c)}) \subset \mathcal{T}$. These

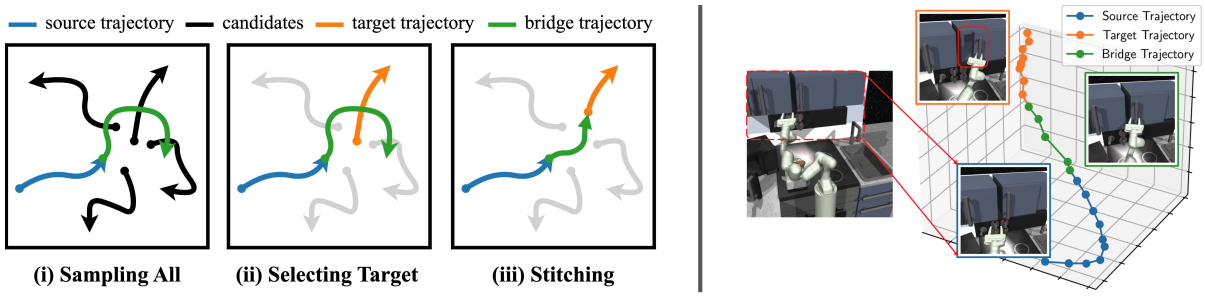


Figure 1: Progressive Trajectory Extension process. **(Left)** PTE overview: (i) A source trajectory (blue), a bridge (green), and target candidates (black) are sampled. (ii) Candidates are filtered by proximity to the bridge, and a valid target (orange) is selected. (iii) The extended trajectory is formed by bridging the source to the chosen target. **(Right)** Visualization of this process in the Franka Kitchen environment.

three components—the source, the bridge, and the candidates—form the basis for the stitching process in each round of PTE.

(ii) Selecting a target trajectory. Note that not all candidate targets will be reachable by the sampled bridge. We therefore compute the minimum distance (e.g., Euclidean or cosine similarity) between the bridge trajectory $\boldsymbol{\tau}_b$ and each candidate $\boldsymbol{\tau}_t^{(i)}$. We then select one candidate whose distance to the bridge is below a threshold, for example by choosing uniformly at random among the valid ones.

(iii) Stitching the source and the target. After selecting a target based on the initial bridge, we refine the connection to improve quality. In the target trajectory $\boldsymbol{\tau}_t$, there exists a state that is closest to the bridge $\boldsymbol{\tau}_b$; suppose this is the k-th state. Then, we sample a new bridge $\boldsymbol{\tau}_b'$ from the stitcher, conditioned on both the last state of $\boldsymbol{\tau}_s$ and the first k states of $\boldsymbol{\tau}_t$, to provide a smoother and more coherent connection. Finally, we concatenate the source, the refined bridge, and the target to form the extended trajectory: $\boldsymbol{\tau}^{\text{new}} = \boldsymbol{\tau}_s \parallel \boldsymbol{\tau}_b' \parallel \boldsymbol{\tau}_t$. The full procedure is described in Algorithm E.1.

Trajectory extension strategies While PTE is a general framework with flexible instantiations, its behavior largely depends on how the source and target sets are constructed in each round. Since our goal is to extend the planning horizon, we design the following two strategies to progressively increase trajectory length.

Linear Extension is the default strategy for steadily increasing trajectory length over successive PTE rounds. In this approach, the source set consists of trajectories extended in the previous round, while the target set remains fixed as the original base dataset: $\mathcal{S} = \mathcal{D}_{r-1}$, $\mathcal{T} = \mathcal{D}_0$. This setup ensures that each round yields slightly longer trajectories than the last, enabling stable and incremental growth.

Exponential Extension accelerates trajectory growth by using the output of the previous round as both the source and target sets: $\mathcal{S}_r = \mathcal{T}_r = \mathcal{D}_{r-1}$. By stitching together already-extended trajectories, this strategy enables faster horizon expansion, which is beneficial in large or complex environments. However, it may result in a sparser distribution of intermediate lengths compared to the linear case. We analyze this trade-off in more detail in Section .

After R rounds of PTE, we obtain an extended dataset $\mathcal{D}_R$ with trajectories significantly longer than those in $\mathcal{D}_0$. This dataset is then used to train our hierarchical planner, designed for effective long-horizon planning.

Hierarchical Multiscale Diffuser

A straightforward way to train a planner on the extended dataset D_R is to learn a single diffusion model to generate entire long trajectories using the standard Diffuser (Janner et al. 2022; Ajay et al. 2022). However, scaling Diffuser to very long horizons poses serious challenges: the trajectory length directly affects the model's output dimensionality and the complexity of denoising, leading to much higher computational cost. Moreover, prior studies (Chen et al. 2024b) has shown that Diffuser's performance degrades as the planning horizon grows, due to the difficulty of modeling extremely long sequences.

To address these issues, we adopt the Hierarchical Diffuser (HD) framework (Chen et al. 2024b), which improves efficiency and scalability through structured planning. Our planner consists of a hierarchy of L levels, where each level $\ell \in 1, \cdots, L$ employs a diffusion model $p_{\theta_\ell}(\boldsymbol{\tau})$. The effective planning horizon at level ℓ is determined by a jump length j_ℓ and a jump count k_ℓ, giving $H_\ell = j_\ell \times k_\ell$. For example, if $j_\ell = 10$, meaning the planner outputs one state every 10 steps, and $k_\ell = 5$, generating five such states, the resulting trajectory spans 50 steps in total.

This temporally sparse structure enables efficient planning, as each level generates only every j_ℓ-th state over k_ℓ steps, resulting in an effective horizon of H_ℓ with reduced output dimensionality. The resulting states, denoted as $g_\ell^{(1)}, \ldots, g_\ell^{(k_\ell)}$, serve as **subgoals** for lower levels.

The core idea of hierarchical planning in HD is that each level ℓ generates a jumpy trajectory conditioned on two consecutive subgoals from the higher level $\ell + 1$, with one as the start and the other as the end. Given subgoals $g_{\ell+1}^{(t)}$ and $g_{\ell+1}^{(t+1)}$, the level-ℓ planner generates a sequence as follows:

$$(g_\ell^{(1)}, \ldots, g_\ell^{(k_\ell - 1)}, g_\ell^{(k_\ell)}) \sim p_{\theta_\ell}(\boldsymbol{\tau} | g_\ell^{(1)} = g_{\ell+1}^{(t)}, g_\ell^{(k_\ell)} = g_{\ell+1}^{(t+1)})$$

This hierarchical structure ensures that lower-level plans are consistent with higher-level subgoals. To maintain alignment across levels, we set $H_\ell = j_{\ell+1}$, so that each jump at level $\ell + 1$ is decomposed into k_ℓ subgoals at level ℓ. At the lowest level, we set $j_1 = 1$, producing a fine-grained, dense plan.

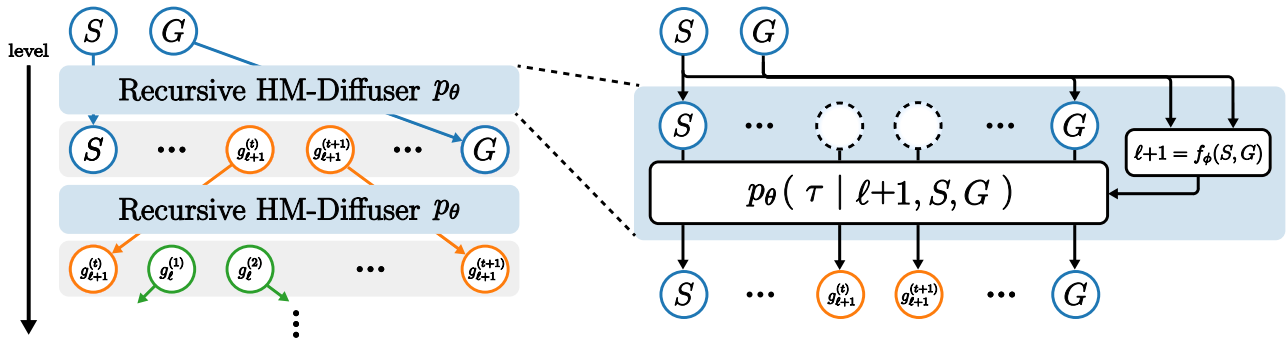


Figure 2: Recursive Hierarchical Multiscale Diffuser (HM-Diffuser). Given start S and goal G, APP selects an initial level $\ell{+}1$.A level-conditioned diffuser generates subgoals that are recursively refined across lower levels, progressively forming a trajectory from coarse to fine.

Adaptive Plan Pondering A limitation of the above hierarchical planner is that it always starts from the top level L, regardless of the distance to the goal. This can lead to inefficiencies when the goal is nearby, as it produces unnecessarily indirect plans. To address this, we introduce **Adaptive Plan Pondering (APP)**, which dynamically adjusts the starting level based on the goal proximity. Specifically, we train a depth predictor f_ϕ that takes the start and goal states (s_0, s_g) and outputs the most suitable planning level $\bar{\ell} = f_\phi(s_0, s_g)$. Since the target level for each trajectory in the training dataset $\mathcal{D}_R$ can be readily determined (e.g., from its length), the predictor can be trained in a straightforward supervised manner.

At test time, APP enables the planner to skip unnecessary higher levels and begin planning from a lower level when appropriate. This not only reduces computational overhead but also improves overall plan quality by avoiding unnecessary detours.

Recursive HM-Diffuser Another key limitation of the hierarchical approach is the need to maintain separate diffuser models for each level—$p_{\theta_1}, p_{\theta_2}, \ldots, p_{\theta_L}$—each with its own set of parameters. This raises an important question: can a single diffusion model effectively handle all levels of the hierarchy? While sharing parameters may not necessarily improve performance compared to non-shared models with larger capacity, it significantly simplifies model management and reduces computational complexity, making it a desirable approach if performance can be maintained.

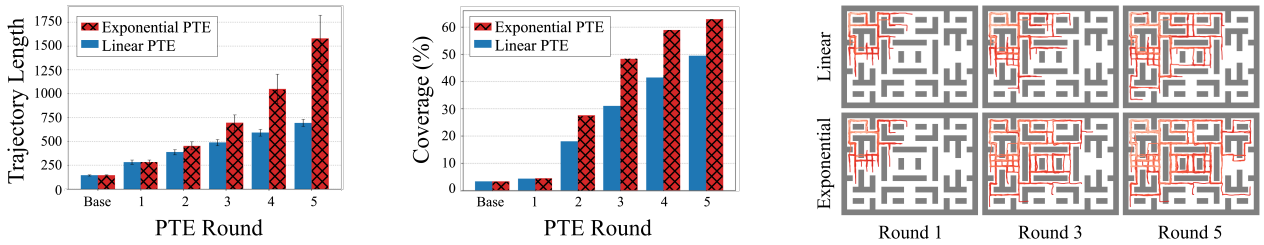
To that end, we incorporate Recursiveness into HM-Diffuser, enabling a single diffusion model to handle the entire hierarchy, as illustrated in Figure 2. Instead of maintaining separate diffusers for each level, we replace them with a single level-conditioned diffusion model $p_\theta(\boldsymbol{\tau}|\ell)$. During training, the shared diffuser is trained to generalize across different levels by uniformly sampling $\ell \sim \text{Uniform}(1, \ldots, L)$, ensuring exposure to all levels and enabling it to generate trajectories conditioned on any level.

For planning, the process starts at the appropriate level predicted by APP. Once a high-level subgoal sequence is generated, each pair of consecutive subgoals $(g_{\ell+1}^{(t)}, g_{\ell+1}^{(t+1)})$ is recursively fed back into the same diffuser, with the level indicator explicitly decreased by one from $\ell + 1$ to ℓ. The model then generates a refined trajectory at the lower level, conditioned on a start and end state defined by a pair of consecutive states from the higher level: $p_\theta(\boldsymbol{\tau}|\ell, g_\ell^{(1)} = g_{\ell+1}^{(t)}, g_\ell^{(k_\ell)} = g_{\ell+1}^{(t+1)})$. This process continues until the lowest level is reached, resulting in a fully specified dense trajectory.

Related Works

Diffusion-based planners in offline RL. Diffusion models are powerful generative models that frame data generation as an iterative denoising process (Ho, Jain, and Abbeel 2020; Song, Meng, and Ermon 2020). They were first introduced in reinforcement learning as planners by (Janner et al. 2022), utilizing their sequence modeling capabilities. Subsequent work (Ajay et al. 2022; Liang et al. 2023; Rigter, Yamada, and Posner 2023) has shown promising results in offline-RL tasks. Diffusion models have also been explored as policy networks to model multi-modal behavior policies (Wang, Hunt, and Zhou 2023; Kang et al. 2024). Recent advancements have extended these models to hierarchical architectures (Wenhao Li and Zha 2023; Chen et al. 2024b; Dong et al. 2024; Chen et al. 2024a), proving effective for long-horizon planning. Our method builds on this by not only using diffusion models for extremely long planning horizons but also exploring the stitching of very short trajectories with them. Additional related works on hierarchical planning are provided in Appendix B.

Data augmentation in RL has been a crucial strategy for improving generalization in offline RL. Previous work used dynamic models to stitch nearby states (Char et al. 2022), generate new transitions (Hepburn and Montana 2022), or create trajectories from sampled initial states (Zhou et al. 2023; Lyu, Li, and Lu 2022; Wang et al. 2021; Zhang et al. 2023). Recently, diffusion models have been applied for augmentation (Zhu et al. 2023b). (Lu et al. 2023) used diffusion models to capture the joint distribution of transition tuples, while (He et al. 2024) extended it to multi-task settings. (Li et al. 2024) used diffusion to connect trajectories with inpainting.



(a) Mean Trajectory Length (b) Start-Goal Pair Coverage (c) Maze Coverage over PTE Rounds

Figure 3: Progressive Trajectory Extension (PTE) results. (a) Mean trajectory lengths for the XXLarge maze. Linear PTE increases trajectory length at a constant rate, while Exponential PTE grows exponentially across rounds. (b) Start–goal coverage increases over rounds, with Exponential PTE covering a larger area. (c) Maze coverage across PTE rounds, showing faster and broader expansion with Exponential PTE.

Experiments

We aim to investigate the following questions through our experiments: (1) Can PTE generate trajectories that are significantly longer than those originally collected from the environment? (2) Can HM-Diffuser leverage these extended trajectories to produce coherent long-horizon plans? (3) Is the overall approach effective in high-dimensional manipulation tasks? To facilitate our analysis, we introduce the Plan Extendable Trajectory Suite (PETS).

Plan Extendable Trajectory Suite

In practice, collecting long trajectories is often costly and impractical, whereas short-horizon data is far more accessible. However, existing benchmarks rarely evaluate the ability to perform long-horizon planning when only short-horizon data is available. To address this gap, we introduce the **Plan Extendable Trajectory Suite (PETS)**—a benchmark built by restructuring existing environments to evaluate this capability. PETS spans three domains—Maze2D, Franka Kitchen, and Gym-MuJoCo—each chosen to represent a distinct challenge in control and generalization.

Extendable Maze2D. Maze2D is a widely used environment for evaluating planning algorithms. Building on the existing Large Maze from D4RL and the Giant Maze from Park et al. (2024), we additionally introduce a new XXLarge Maze, four times larger than the Large Maze, to enable more rigorous evaluation of long-horizon planning. To assess extendable planning capability, training data consists only of short trajectories between 130 and 160 steps. However, at test time, tasks require reaching goals up to 1,000 steps, depending on the maze size. See Appendix D for details.

Extendable Franka Kitchen. FrankaKitchen is a widely used benchmark for complex, high-dimensional manipulation tasks. We modify the training data by splitting the D4RL offline trajectories into non-overlapping segments of length 10, exposing the model only to short-horizon trajectories during training. At test time, however, it must complete compound tasks that require much longer sequences involving multiple stages of manipulation. This setup enables evaluation of extendable long-horizon planning while preserving the full complexity of the original high-dimensional environment. See Appendix F for details.

Extendable Gym-MuJoCo. Gym-MuJoCo environments are widely used benchmarks for low-level motor control, but they are not originally designed for long-horizon planning. To adapt them, we split the D4RL offline trajectories from Hopper and Walker2D into non-overlapping segments of length 10, so that only short-horizon trajectories are available during training. At test time, however, models are required to plan over much longer horizons while maintaining temporally coherent control. This setup retains the dense reward structure of the original environments while enabling assessment of extendable long-horizon planning. See Appendix F for details.

We now present experimental results based on the PETS benchmark. Throughout this section, we use the -X suffix to denote baselines trained on the extended dataset generated by PTE, which reflects their ability to model long-horizon plans. In contrast, baselines without this suffix are trained solely on the short base dataset. Additional implementation details are provided in Appendix C.

Extendable Maze2D results

We first validate our proposed unified framework of PTE and HM-Diffuser in the Extendable Maze2D environment, focusing on whether PTE can generate trajectories that connect unseen start-goal pairs to improve maze coverage, and whether HM-Diffuser can utilize them to reach goals situated in remote areas of the maze.

PTE on Extendable Maze2D. As shown in Figure 3(a), linear and exponential variants of PTE progressively and substantially increase trajectory lengths over successive rounds. While both methods show similar gains in the early stages, their growth dynamics differ: Linear PTE increases length at a steady rate, reaching roughly 700 steps by round 5, whereas exponential PTE accelerates rapidly, exceeding 1,500 steps at the same round as trajectories from previous rounds are stitched together to form even longer ones.

This trajectory extension leads to broader maze coverage, as illustrated in Figure 3(b). Both variants significantly expand the diversity of reachable start-goal pairs compared to the base dataset, which covers less than 5%. By round 5, linear PTE increases coverage to over 65%, and exponential PTE further extends it beyond 80%. However, as detailed in Section , since exponential PTE requires more data than

Environment		Diffuser	Diffuser-X	HD-X	HMD-X
Maze2D	Large	45.8 ± 5.9	114.4 ± 4.7	144.3 ± 3.3	$\mathbf{166.9} \pm 4.6$
Maze2D	Giant	77.2 ± 11.7	114.6 ± 9.10	138.4 ± 9.6	$\mathbf{177.4} \pm 11.9$
Maze2D	XXLarge	14.1 ± 4.9	23.0 ± 3.4	56.4 ± 4.9	$\mathbf{82.1} \pm 8.5$
Single-task Average		45.7	84.0	113.0	**142.1**
Multi2D	Large	33.4 ± 5.9	130.3 ± 4.0	150.3 ± 3.0	$\mathbf{174.7} \pm 3.8$
Multi2D	Giant	77.8 ± 11.8	140.2 ± 8.9	168.8 ± 9.0	$\mathbf{246.7} \pm 10.6$
Multi2D	XXLarge	23.4 ± 6.3	63.2 ± 5.2	71.9 ± 5.5	$\mathbf{109.9} \pm 8.4$
Multi-task Average		44.9	111.2	130.4	**177.1**

Table 1: Extendable Maze2D performance. We compare baselines across Large, Giant, and XXLarge mazes. -X baselines are trained on data extended with 7 rounds of Linear PTE, while Diffuser uses only the base dataset. Results are averaged over 200 planning seeds. HMD-X consistently outperforms all baselines.

Linear one, we adopt linear PTE as the default setting in our experiments due to its stability and data efficiency. In the following experiments, we apply 7 rounds of Linear PTE to extend trajectory lengths. The resulting datasets are then aggregated into a single training set.

HM-Diffuser on Extendable Maze2D. Using the extended dataset, we train Diffuser, HD, and our proposed HM-Diffuser, denoting these models with the -X suffix. Following Diffuser (Janner et al. 2022), we evaluate performance under two settings. In the single-task setting (Extendable Maze2D), the goal position is fixed at the bottom-right corner of the maze while the start position is randomized. The multi-task setting (Extendable Multi2D) randomizes both start and goal positions to assess the model's generalization across a wider range of tasks. Evaluation details are provided in Appendix D.

Table 1 demonstrates that HM-Diffuser-X consistently outperforms all baselines across both settings. In the single-task setting, HM-Diffuser-X achieves the highest average score (142.1), highlighting the effectiveness of its hierarchical and multiscale planning capabilities. Meanwhile, Diffuser-X, despite being trained on the extended dataset, struggles in the XXLarge maze environment, achieving only 23.0 ± 3.4—a performance comparable to Diffuser trained solely on the base dataset. This suggests that Diffuser-X faces challenges in modeling long-horizon plans in complex environments. However, HD-X and HM-Diffuser-X manage to retain better performance, demonstrating the effectiveness of the hierarchical planning. In the multi-task setting, HM-Diffuser-X outperforms all baselines, achieving an average score of 177.1, demonstrating its ability to generalize across diverse tasks.

These results demonstrate that PTE effectively constructs longer and more diverse trajectories, enabling HM-Diffuser to plan over substantially longer horizons than those present in the original dataset. The combination of scalable data generation via PTE and efficient hierarchical planning via HM-Diffuser proves essential for solving complex, long-horizon tasks. Detailed results across different PTE rounds are presented in Figure D.8.

Extendable Franka Kitchen and Extendable Gym-MuJoCo results

We also evaluate our framework on Extendable Franka Kitchen and Extendable Gym-MuJoCo to assess its ability to generalize beyond the short training trajectories and produce coherent plans over extended horizons in both high-dimensional manipulation and low-level locomotion control.

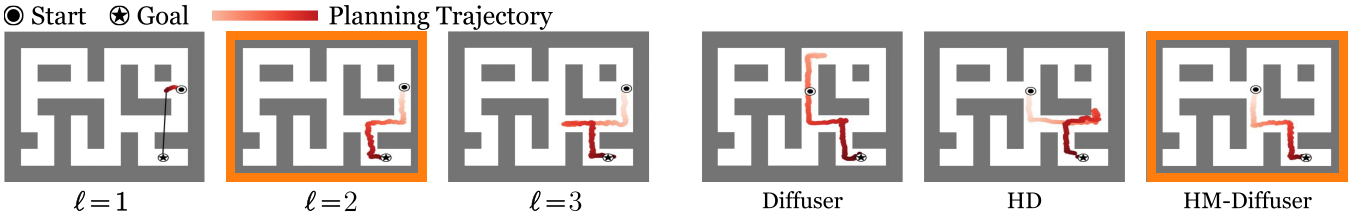
Environment		Diffuser	Diffuser-X	HD-X	HMD-X
Kitchen	Partial-v0	41.7 ± 3.2	43.3 ± 5.5	$\mathbf{56.7} \pm 5.8$	$\mathbf{56.7} \pm 5.3$
Kitchen	Mixed-v0	45.8 ± 3.1	48.3 ± 4.7	53.3 ± 3.1	$\mathbf{61.7} \pm 3.1$
Kitchen Average		43.8	45.8	55.0	**59.2**
Walker2d	MedReplay	22.8 ± 2.7	20.1 ± 4.3	$\mathbf{30.2} \pm 5.9$	29.6 ± 4.8
Walker2d	Medium	58.1 ± 5.6	62.6 ± 6.4	66.5 ± 4.3	$\mathbf{72.7} \pm 2.5$
Walker2d	MedExpert	$\mathbf{82.3} \pm 4.6$	80.3 ± 3.7	80.8 ± 2.9	79.3 ± 2.3
Walker2d Average		54.4	54.3	59.2	**60.5**
Hopper	MedReplay	18.7 ± 3.0	34.5 ± 6.2	22.5 ± 3.1	$\mathbf{37.3} \pm 4.8$
Hopper	Medium	$\mathbf{45.6} \pm 1.9$	44.3 ± 3.5	44.1 ± 2.8	44.9 ± 3.5
Hopper	MedExpert	61.4 ± 8.4	$\mathbf{74.9} \pm 8.0$	67.9 ± 7.7	74.3 ± 9.0
Hopper Average		41.9	51.2	44.8	**52.2**

Table 2: Performance on Extendable Franka Kitchen and Extendable Gym-MuJoCo. Training on extended data improves Diffuser-X, and a recursive hierarchical structure further enhances performance. Results are averaged over 15 planning seeds.

PTE on Extendable Franka Kitchen and Extendable Gym-MuJoCo. To assess the effect of PTE in these complex environments, we analyze the distribution of normalized returns—defined as total return divided by trajectory length. This metric provides a conservative estimate of data quality: if longer trajectories are generated without meaningful improvement, the normalized return would decline. As shown in Figure 5, PTE shifts the distribution upward, increasing the density of high-return trajectories and indicating improved overall data quality. Despite a slight dip in Round 1 for Extendable Hopper-Medium-Replay, likely attributed to dataset characteristics, the normalized return improves in Round 2, demonstrating the effectiveness of PTE over time.

HM-Diffuser on Extendable Franka Kitchen. As shown in Table 2, training on the extended dataset improves the performance of Diffuser-X from 43.8 to 45.8, likely due to the longer planning horizon enabled by PTE, which addresses short-horizon limitations by connecting short trajectories into longer ones that complete the task (see Appendix G.15). Both HM-Diffuser-X and HD-X outperform Diffuser-X by leveraging hierarchical planning that reduces complexity—a trend also observed in Chen et al. (2024c). Between the two, HM-Diffuser-X scores higher than HD-X: 59.2 vs. 55.0, demonstrating the benefit of our efficient parameter sharing and adaptive planning design.

HM-Diffuser on Extendable Gym-MuJoCo. As shown in Table 2, training on the extended dataset improves the average performance of Diffuser-X on Extendable Hopper from 41.9 to 51.2, likely due to PTE alleviating limitations from short-horizon training data. On Extendable Walker2D, Diffuser-X performs similarly to Diffuser (54.3 vs. 54.4), showing no notable gain despite access to extended trajectories. However, prior work (Chen et al. 2024c) reports that standard Diffuser models often suffer performance degradation when planning over long horizons due to increased



(a) Multiscale Planning with HM-Diffuser (b) Long-Horizon Planning Comparison

Figure 4: Adaptive planning in HM-Diffuser. (a) Multiscale trajectories at levels $\ell = 1, 2, 3$. The highlighted level is selected by the depth predictor for efficient planning. (b) Long-horizon planning comparison: HM-Diffuser produces more direct trajectories than Diffuser and HD.

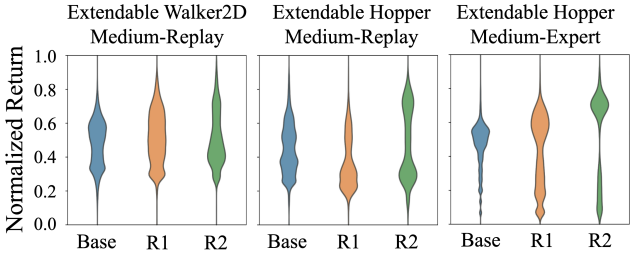


Figure 5: Trajectory quality. We evaluate normalized return (total return divided by length) to compare trajectories as stitching increases length. Despite penalizing longer trajectories, normalized return improves across PTE rounds, indicating progressively higher-quality data.

complexity. In contrast, our results show that Diffuser-X maintains stable performance, suggesting that PTE mitigates such degradation by generating higher-quality training data. Notably, HM-Diffuser-X outperforms both Diffuser-X and HD-X, achieving average scores of 60.5 on Extendable Walker2D and 52.2 on Extendable Hopper.

Ablation studies

To better understand the effectiveness of PTE and HM-Diffuser, we conduct ablations on key design choices, including the adaptive planning mechanism, the impact of different extension strategies, and comparisons with standard benchmarks and alternative data augmentation methods.

Linear PTE and Exponential PTE. We compare Linear and Exponential PTE on the Extendable Maze2d-XXLarge task and find that Exponential PTE is highly sensitive to dataset size. Linear PTE shows steady performance improvements as the dataset size grows from 1M to 4M, with scores rising from 82.1 ± 8.5 to 89.9 ± 8.2 using data generated from 7 PTE rounds. In contrast, Exponential PTE improves dramatically from 30.0 ± 5.5 to 120.5 ± 8.6 over the same range, highlighting its dependence on larger datasets. As our goal is to evaluate general applicability rather than maximize absolute performance, we adopt Linear PTE as the default. Detailed results are provided in Table D.4.

PTE outperforms other diffusion-based data augmentation methods. Next, we compare our proposed PTE with DiffStitch (Li et al. 2024), a diffusion-based data augmentation method, on the extendable planning problem. Table G.10 shows that all planners trained in PTE-augmented datasets outperform those trained in DiffStitch-augmented datasets. For further analysis, please see Appendix G.

Adaptive Plan Pondering. Figure 4 (a) illustrates HM-Diffuser's multiscale planning across levels, where $\ell = 1$ denotes the lowest level. By selecting an appropriate level via the depth predictor (orange box), HM-Diffuser maintains effective planning without degradation (see Figure D.10). In contrast, Figure 4 (b) shows that Diffuser and HD generate inefficient trajectories with detours due to fixed planning horizons (see Figure D.11). Unlike other methods, HM-Diffuser performs adaptive planning by selecting the appropriate level per task. Figure D.9 illustrates how hierarchical planning proceeds from the predicted level down to lower levels, enabling efficient and flexible trajectory generation.

HM-Diffuser performs reasonably well on standard benchmarks. Finally, we evaluate the proposed HM-Diffuser on the standard D4RL benchmark. Unlike previous experiments using PTE-processed data, the trajectories here are taken directly from the original D4RL dataset without splitting. Table G.8 demonstrates that HM-Diffuser outperforms all baselines in the long-horizon planning tasks and performs reasonably well in standard control tasks.

Conclusion

In this work, we introduce the Hierarchical Multiscale Diffuser framework for the proposed extendable long-horizon planning problem. Starting from a set of short, suboptimal trajectories, our approach employs Progressive Trajectory Extension to generate longer, more informative sequences. We then train an HM-Diffuser planner on the augmented dataset, leveraging a hierarchical multiscale structure to efficiently model and execute long-horizon plans. Experiments demonstrate that our framework achieves promising results across a diverse set of benchmarks, including the long-horizon Extendable Maze2D, dense-reward Extendable Gym-MuJoCo, and high-dimensional Extendable FrankaKitchen manipulation tasks.

While HM-Diffuser substantially improves long-horizon planning, several limitations remain. First, further improving stitching quality could yield additional performance gains. Second, HM-Diffuser currently operates in state space; extending it to image-based planning is essential for real-world applications. Third, plan pondering is limited to discrete levels, and the lack of temporal abstraction may limit scalability to extremely long horizons. Finally, HM-Diffuser lacks test-time adaptivity; integrating search-based refinement like MCTS could enhance execution flexibility.

Acknowledgments

This research was supported by Brain Pool Plus Program (No. 2021H1D3A2A03103645) and GRDC (Global Research Development Center) Cooperative Hub Program (RS-2024-00436165) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT). This work was also partly supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (25ZR1100) and Samsung Advanced Institute of Technology.

References

Ajay, A.; Du, Y.; Gupta, A.; Tenenbaum, J.; Jaakkola, T.; and Agrawal, P. 2022. Is Conditional Generative Modeling all you need for Decision-Making? *arXiv preprint arXiv:2211.15657*.

Bachmann, G.; and Nagarajan, V. 2024. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*.

Char, I.; Mehta, V.; Villaflor, A.; Dolan, J. M.; and Schneider, J. 2022. BATS: Best Action Trajectory Stitching. *arXiv e-prints*, arXiv–2204.

Chen, C.; Baek, J.; Deng, F.; Kawaguchi, K.; Gulcehre, C.; and Ahn, S. 2024a. PlanDQ: Hierarchical Plan Orchestration via D-Conductor and Q-Performer. In *Forty-first International Conference on Machine Learning*.

Chen, C.; Deng, F.; Kawaguchi, K.; Gulcehre, C.; and Ahn, S. 2024b. Simple hierarchical planning with diffusion. *arXiv preprint arXiv:2401.02644*.

Chen, C.; Deng, F.; Kawaguchi, K.; Gulcehre, C.; and Ahn, S. 2024c. Simple Hierarchical Planning with Diffusion. In *The Twelfth International Conference on Learning Representations*.

Dhariwal, P.; and Nichol, A. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34: 8780–8794.

Dong, Z.; Hao, J.; Yuan, Y.; Ni, F.; Wang, Y.; Li, P.; and Zheng, Y. 2024. DiffuserLite: Towards Real-time Diffusion Planning. *arXiv preprint arXiv:2401.15443*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*.

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Hafner, D.; Lee, K.-H.; Fischer, I.; and Abbeel, P. 2022. Deep hierarchical planning from pixels. *arXiv preprint arXiv:2206.04114*.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, 2555–2565. PMLR.

Hamrick, J. B.; Friesen, A. L.; Behbahani, F.; Guez, A.; Viola, F.; Witherspoon, S.; Anthony, T.; Buesing, L.; Veličković, P.; and Weber, T. 2020. On the role of planning in model-based deep reinforcement learning. *arXiv preprint arXiv:2011.04021*.

Hansen, N.; Wang, X.; and Su, H. 2022. Temporal Difference Learning for Model Predictive Control. *arXiv preprint arXiv:2203.04955*.

He, H.; Bai, C.; Xu, K.; Yang, Z.; Zhang, W.; Wang, D.; Zhao, B.; and Li, X. 2024. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36.

Hepburn, C. A.; and Montana, G. 2022. Model-based Trajectory Stitching for Improved Offline Reinforcement Learning. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33: 6840–6851.

Hu, E. S.; Chang, R.; Rybkin, O.; and Jayaraman, D. 2023. Planning Goals for Exploration. In *The Eleventh International Conference on Learning Representations*.

Janner, M.; Du, Y.; Tenenbaum, J.; and Levine, S. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning*.

Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*.

Kang, B.; Ma, X.; Du, C.; Pang, T.; and Yan, S. 2024. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Lambert, N.; Pister, K.; and Calandra, R. 2022. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*.

Li, G.; Shan, Y.; Zhu, Z.; Long, T.; and Zhang, W. 2024. DiffStitch: Boosting Offline Reinforcement Learning with Diffusion-based Trajectory Stitching. *arXiv preprint arXiv:2402.02439*.

Li, J.; Tang, C.; Tomizuka, M.; and Zhan, W. 2022. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4): 10216–10223.

Li, W.; Wang, X.; Jin, B.; and Zha, H. 2023. Hierarchical Diffusion for Offline Decision Making. In *International Conference on Machine Learning*.

Liang, Z.; Mu, Y.; Ding, M.; Ni, F.; Tomizuka, M.; and Luo, P. 2023. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*.

Lu, C.; Ball, P. J.; Teh, Y. W.; and Parker-Holder, J. 2023. Synthetic Experience Replay. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lyu, J.; Li, X.; and Lu, Z. 2022. Double Check Your State Before Trusting It: Confidence-Aware Bidirectional Offline Model-Based Imagination. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Mattar, M. G.; and Lengyel, M. 2022. Planning in the brain. *Neuron*, 110(6): 914–934.

Park, S.; Frans, K.; Eysenbach, B.; and Levine, S. 2024. OGBench: Benchmarking Offline Goal-Conditioned RL. *arXiv preprint arXiv:2410.20092*.

Rigter, M.; Yamada, J.; and Posner, I. 2023. World models via policy-guided trajectory diffusion. *arXiv preprint arXiv:2312.08533*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2256–2265. PMLR.

Song, J.; Meng, C.; and Ermon, S. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Wang, J.; Li, W.; Jiang, H.; Zhu, G.; Li, S.; and Zhang, C. 2021. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34: 29420–29432.

Wang, Z.; Hunt, J. J.; and Zhou, M. 2023. Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

Wenhao Li, B. J., Xiangfeng Wang; and Zha, H. 2023. Hierarchical Diffusion for Offline Decision Making. *Proceedings of the 40th International Conference on machine learning*.

Zhang, J.; Lyu, J.; Ma, X.; Yan, J.; Yang, J.; Wan, L.; and Li, X. 2023. Uncertainty-driven trajectory truncation for data augmentation in offline reinforcement learning. In *ECAI 2023*, 3018–3025. IOS Press.

Zhou, Z.; Zhu, C.; Zhou, R.; Cui, Q.; Gupta, A.; and Du, S. S. 2023. Free from Bellman Completeness: Trajectory Stitching via Model-based Return-conditioned Supervised Learning. *arXiv preprint arXiv:2310.19308*.

Zhu, J.; Wang, Y.; Wu, L.; Qin, T.; Zhou, W.; Liu, T.-Y.; and Li, H. 2023a. Making Better Decision by Directly Planning in Continuous Control. In *The Eleventh International Conference on Learning Representations*.

Zhu, Z.; Zhao, H.; He, H.; Zhong, Y.; Zhang, S.; Yu, Y.; and Zhang, W. 2023b. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*.