Compositional Training-Free Diffusion Planning for Long-Horizon Multiple Reach-Avoid Tasks

Yu Chen
Department of Automation
Shanghai Jiao Tong University
Yuchen26@sjtu.edu.cn

Ruijia Liu
Department of Automation
Shanghai Jiao Tong University
liuruijia@sjtu.edu.cn

Xiao Yu
Institute of Artificial Intelligence
Xiamen University
xiaoyu@xmu.edu.cn

Xiang Yin*
Department of Automation
Shanghai Jiao Tong University
yinxiang@sjtu.edu.cn

Abstract

Long-horizon robotic planning often requires satisfying temporally ordered safety and visitation constraints rather than a single goal condition. We study multiple reach-avoid (MRA) tasks, in which a robot must sequentially visit target regions while remaining within corresponding safe sets. Existing approaches either rely on explicit system models or use data-driven planners whose test-time conditioning mechanisms do not scale well to long-horizon temporally structured tasks. We propose a training-free compositional diffusion planning framework for MRA tasks that operates directly on a pre-trained task-agnostic diffusion trajectory prior at test time, without requiring a known dynamics model. Our approach uses established factor-graph composition and neighboring-score averaging for overlapping short-horizon priors, while adding MRA-specific stage-wise projection and explicitly shared milestones. The same construction also extends naturally to prefix-suffix tasks through a looped compositional graph. We prove that, whenever the sampler returns a stitched discrete state trajectory, that planned trajectory satisfies the MRA specification. This guarantee does not establish dynamic feasibility, collision-free motion between sampled states, or safe execution. Experiments on long-horizon constrained planning benchmarks report empirical controller-executed success, substantially lower planning time than representative guidance-based baselines, and transfer to richer dynamics.

1 Introduction

Many robotic missions require temporally extended behavior rather than one-shot goal reaching. In inspection, warehouse logistics, autonomous navigation, and long-horizon manipulation, a robot may need to visit multiple regions in a prescribed order while remaining safe throughout execution. Such objectives are naturally expressed using temporal logics such as LTL and STL [1, 2, 3]. Classical solution pipelines therefore rely on abstraction-based synthesis, optimization-based planning, or sampling-based search to construct trajectories or controllers that satisfy the specification [2, 3, 4, 5]. While effective when accurate models are available, these approaches typically assume that the system dynamics, transition structure, and environment are known sufficiently well in advance. This assumption is restrictive in many robotic settings with complex contacts, nonlinear dynamics, or partially known environments.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Another line of work addresses temporally extended objectives through learning-based control synthesis, especially reinforcement learning (RL) [6, 7, 8, 9, 10, 11]. At a high level, these methods encode task semantics directly into training. For LTL tasks, a common approach is to translate the specification into an automaton or progression state and learn on the corresponding augmented MDP or hierarchical decomposition [6, 7, 8]. For STL tasks, a common approach is to optimize rewards derived from logical robustness values or related surrogate objectives [9, 10, 11]. However, these methods are typically task-oriented rather than task-agnostic: moving from classical goal-reaching to richer temporal specifications often requires specification-dependent reward design, product-state construction, hierarchy design, or dedicated retraining on the temporally extended objective.

A complementary data-driven alternative is offered by diffusion models [12, 13], which learn trajectory priors directly from offline data without explicit system identification. In robotics, diffusion-based planners have already shown strong performance across a broad range of planning and control settings [14, 15, 16, 17, 18]. Their main appeal is test-time flexibility: one can first learn a task-agnostic trajectory prior and then impose downstream objectives during sampling via guidance or projection-based conditioning [19, 20, 21, 22, 23]. Yet existing test-time conditioning mechanisms are most effective for relatively simple objectives, such as goal reaching, local safety constraints, or waypoint satisfaction. They do not naturally scale to long-horizon temporally structured tasks, where multi-stage constraints and long-range dependencies must be enforced consistently across an entire trajectory.

Motivated by this gap, we study multiple reach-avoid (MRA) tasks [24], in which a robot must reach a prescribed sequence of target regions while remaining inside corresponding safe sets. MRA tasks arise naturally in long-horizon planning and also serve as a useful building block for handling broader temporal specifications, including LTL and STL [24, 25]. We propose a compositional diffusion planning framework that uses only a pre-trained task-agnostic diffusion prior, without requiring a known dynamics model or any additional training. We adapt established factor-graph composition and neighboring-score averaging to MRA segments, then combine explicitly shared milestone states with stage-wise reach-avoid projection. A looped overlap graph extends this MRA-specific construction to prefix-suffix tasks.

Taken together, our contributions are fourfold. First, we integrate stage-wise hard MRA projection and explicitly shared milestone states with an established compositional diffusion sampler, without task-specific retraining. Second, we extend this MRA-specific construction from finite-horizon tasks to prefix-suffix tasks through a looped overlap graph. Third, we prove MRA satisfaction for a returned stitched discrete plan under the stated overlap assumptions; the theorem makes no claim about dynamic feasibility or safe execution. Fourth, experiments on long-horizon constrained planning benchmarks report empirical success of controller-executed rollouts, substantially lower planning time than guidance-based baselines, and transfer to richer dynamics.

2 Preliminaries

Let $\mathbb{R}$ and $\mathbb{R}^n$ denote the set of real numbers and the n-dimensional real vector space, respectively. Consider a discrete-time robotic system with state space $\mathcal{S} \subseteq \mathbb{R}^{d_s}$. A finite-horizon trajectory of horizon H is denoted by $\boldsymbol{\tau} = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_H)$.

2.1 Diffusion Trajectory Priors

Let $p_{data}(\boldsymbol{\tau}^0)$ denote the distribution of trajectory demonstrations. We adopt the standard denoising diffusion probabilistic model (DDPM) formulation to learn a trajectory prior through a forward noising process and a reverse denoising process over diffusion steps $j \in \{0, 1, \ldots, J\}$ [12].

The forward process gradually perturbs a clean trajectory with Gaussian noise according to

$$q(\boldsymbol{\tau}^j \mid \boldsymbol{\tau}^{j-1}) = \mathcal{N}\Big(\boldsymbol{\tau}^j; \sqrt{\alpha_j}\boldsymbol{\tau}^{j-1}, (1-\alpha_j)\mathbf{I}\Big), \tag{1}$$

where $\alpha_j = 1 - \beta_j$ and $\{\beta_j\}_{j=1}^J$ is a fixed noise schedule. Equivalently, the noisy sample at level j admits the closed form $\boldsymbol{\tau}^j = \sqrt{\bar{\alpha}_j}\boldsymbol{\tau}^0 + \sqrt{1-\bar{\alpha}_j}\boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bar{\alpha}_j = \prod_{\ell=1}^j \alpha_\ell$.

The reverse process starts from Gaussian noise $\boldsymbol{\tau}^J \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively denoises the sample back to the data manifold:

$$p_\theta(\boldsymbol{\tau}^{j-1} \mid \boldsymbol{\tau}^j) = \mathcal{N}\big(\boldsymbol{\tau}^{j-1}; \boldsymbol{\mu}_\theta(\boldsymbol{\tau}^j, j), \boldsymbol{\Sigma}^j\big), \tag{2}$$

where $\boldsymbol{\Sigma}^j$ is fixed in the original DDPM parameterization. The model is typically parameterized by a neural network $\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}^j, j)$ that predicts the injected noise and is trained with

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{j, \boldsymbol{\epsilon}, \boldsymbol{\tau}^0 \sim p_{data}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}^j, j) \right\|_2^2 \right]. \tag{3}$$

Under the DDPM parameterization, the predicted noise determines the reverse mean through

$$\boldsymbol{\mu}_\theta(\boldsymbol{\tau}^j, j) = \frac{1}{\sqrt{\alpha_j}} \left(\boldsymbol{\tau}^j - \frac{1 - \alpha_j}{\sqrt{1 - \bar{\alpha}_j}} \boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}^j, j) \right). \tag{4}$$

A key interface for our method is the score of the forward process marginal q_j, which is related to noise prediction by [13]

$$\nabla_{\boldsymbol{\tau}^j} \log q_j(\boldsymbol{\tau}^j) \approx -\frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}^j, j)}{\sqrt{1 - \bar{\alpha}_j}}. \tag{5}$$

Thus, the learned diffusion model can be viewed as providing the score information that drives reverse denoising. The noise-prediction form above is the standard DDPM presentation; our experimental backbone instead predicts the clean trajectory and converts that prediction to an implied noise estimate using the forward-noising equation. Appendix C.6 gives the experimental training and DDIM settings.

2.2 Multiple Reach-Avoid Tasks

We consider the *multiple reach-avoid (MRA)* task [24], in which the system must visit a sequence of target regions in a prescribed order while satisfying state constraints between consecutive target arrivals. Formally, let

$$\mathbb{T} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N) \quad \text{and} \quad \mathbb{G} = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N) \tag{6}$$

denote sequences of N target regions and N safe regions, respectively, where $\mathcal{T}_i, \mathcal{G}_i \subseteq \mathcal{S}$ for each $i = 1, \ldots, N$. Let $k_s, k_e \in \mathbb{N}_{\geq 0}$ denote the start and end times of the overall task. The MRA task is then defined by the 4-tuple

$$\Phi = (k_s, k_e, \mathbb{T}, \mathbb{G}). \tag{7}$$

For simplicity, we denote an MRA task by $\Phi^{[k_s, k_e]}$ when the target and safe regions are clear from the context.

A trajectory $\boldsymbol{\tau}$ satisfies the MRA task $\Phi^{[k_s, k_e]}$, denoted by $\boldsymbol{\tau} \models \Phi^{[k_s, k_e]}$, if there exists a sequence of milestone times

$$k_s = m_0 \leq m_1 \leq m_2 \leq \cdots \leq m_N = k_e \tag{8}$$

such that for every $i = 1, \ldots, N$,

$$[\tau_{m_i} \in \mathcal{T}_i] \wedge [\forall k \in \{m_{i-1}, m_{i-1} + 1, \ldots, m_i\} : \tau_k \in \mathcal{G}_i]. \tag{9}$$

Equation (9) requires the sampled states to remain inside the safe region $\mathcal{G}_i$ until the target region $\mathcal{T}_i$ is reached at step m_i. A common special case is the shared-safe-set setting $\mathcal{G}_1 = \cdots = \mathcal{G}_N =: \mathcal{G}$, in which the planned state sequence visits multiple target sets in order while its sampled states remain inside the same global safe region.

2.3 Problem Formulation

We assume access to a pre-trained diffusion model $\mathcal{M}_\theta$ learned from offline trajectories, which serves as a data-driven prior for local motion patterns but does not guarantee dynamic feasibility. Our goal is to use this task-agnostic prior to plan for long-horizon tasks that were not explicitly seen during training. In particular, given an initial state $x_0 \in \mathcal{S}$, a trajectory horizon H, and an MRA task $\Phi = (0, H, \mathbb{T}, \mathbb{G})$, we seek a discrete state trajectory that starts from x_0 and satisfies the specification at its sampled states.

Problem 1 *Given an initial state $x_0 \in \mathcal{S}$, a trajectory horizon H, an MRA task $\Phi = (0, H, \mathbb{T}, \mathbb{G})$, and a pre-trained diffusion prior $\mathcal{M}_\theta$, generate a trajectory $\boldsymbol{\tau}^0$ such that $\tau_0^0 = x_0$ and $\boldsymbol{\tau}^0 \models \Phi$.*

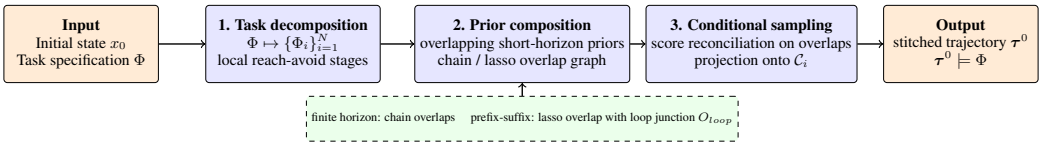


Figure 1: Overview of the proposed compositional diffusion planning framework. Starting from the initial state and a temporally structured task, the method first decomposes the global specification into local reach-avoid sub-tasks, then composes overlapping short-horizon diffusion priors into a long-horizon chain or lasso graph, and finally performs reverse denoising with overlap reconciliation and projection-based constraint enforcement. The prefix-suffix extension reuses the same sampler and changes only the overlap topology.

3 Methodology

3.1 Overview and Notation

Let $\mathcal{M}_\theta$ denote the pre-trained diffusion model for the unconditional trajectory prior $p(\boldsymbol{\tau})$. Given an MRA task Φ, our goal is to generate a denoised trajectory that is likely under this prior and satisfies the specification, namely

$$p(\boldsymbol{\tau}^0 \mid \boldsymbol{\tau}^0 \models \Phi).$$

Directly conditioning a single long trajectory on the full MRA specification is difficult for a task-agnostic, short-horizon diffusion prior. We therefore use a compositional test-time procedure with three stages: decompose Φ into N local reach-avoid sub-tasks, compose the corresponding short-horizon segment priors into a long-horizon prior through overlapping segments, and enforce each local constraint by projection during reverse denoising. This organization leads to a global score consisting of an unconditional compositional term and a conditional task-enforcement term, which are developed below. Figure 1 summarizes this pipeline and highlights how the same sampler extends from finite-horizon MRA tasks to prefix-suffix tasks.

We use three indices throughout the method. The segment index $i \in \{1, \dots, N\}$ identifies the local sub-task and its trajectory segment $\boldsymbol{\tau}_i$. Within segment i, the local time index $k \in \{0, \dots, K_i\}$ denotes the state $\tau_{i,k}$, where K_i is the segment length. The superscript $j \in \{J, J-1, \dots, 0\}$ denotes the diffusion step, so $\tau_{i,k}^j$ is the state at local time k of segment i at denoising step j, and $\boldsymbol{\tau}_i^0$ is the final denoised segment. Consecutive segments overlap at milestone states denoted by O_i. With this notation, we first decompose the MRA likelihood, then compose the long-horizon prior, and finally apply projection-based conditioning. The prefix-suffix case follows from the same compositional construction with a looped overlap structure.

3.2 Decomposition of the MRA Tasks

We write the target conditional distribution using Bayes' theorem as

$$p(\boldsymbol{\tau} \mid \Phi) = \frac{p(\boldsymbol{\tau})p(\Phi \mid \boldsymbol{\tau})}{p(\Phi)} \propto p(\boldsymbol{\tau})p(\Phi \mid \boldsymbol{\tau}), \tag{10}$$

where the proportionality follows because $p(\Phi)$ does not depend on $\boldsymbol{\tau}$, and we first decompose the task-likelihood term $p(\Phi \mid \boldsymbol{\tau})$.

Let $\Phi = (0, H, \mathbb{T}, \mathbb{G})$ be an MRA task with milestone times $0 = m_0 \le m_1 \le \cdots \le m_N = H$, and define the local reach-avoid sub-task $\Phi_i = (m_{i-1}, m_i, \mathcal{T}_i, \mathcal{G}_i)$ for each stage i, where each Φ_i requires the segment to satisfy $\tau_{m_i} \in \mathcal{T}_i$ and $\tau_k \in \mathcal{G}_i$ for all $k \in \{m_{i-1}, \dots, m_i\}$. By the semantics of MRA tasks,

$$\boldsymbol{\tau} \models \Phi \iff \exists 0 = m_0 \le m_1 \le \cdots \le m_N = H, \bigwedge_{i=1}^{N} \boldsymbol{\tau} \models \Phi_i. \tag{11}$$

Let $\boldsymbol{\tau}_i$ denote the trajectory segment from m_{i-1} to m_i. Given the milestone decomposition, the task likelihood factorizes over segments as

$$p(\Phi \mid \boldsymbol{\tau}) = p\left(\bigwedge_{i=1}^{N} \Phi_i \,\middle|\, \boldsymbol{\tau}\right) = \prod_{i=1}^{N} p(\Phi_i \mid \boldsymbol{\tau}_i). \tag{12}$$

This reduces the logical component of the global MRA task to local reach-avoid likelihoods. It remains to represent the long-horizon prior $p(\boldsymbol{\tau})$ using short-horizon segment priors learned by the diffusion model.

3.3 Factor-Graph Composition of the Long-Horizon Trajectory Prior

To compose local segment priors into a long-horizon trajectory prior, we adopt the existing Bethe-style factor-graph construction and neighboring-score averaging used in compositional diffusion [26, 27, 28, 29]. We review these tools before specializing their factors and shared variables to MRA stages and milestones; neither the factorization nor the averaging rule is a new estimator introduced here. Consider a generic trajectory $X = (x_1, x_2, \ldots, x_L)$ and a collection of M local factors y_j, where each y_j corresponds to a contiguous subsequence of X; for example, one may use $y_1 = (x_1, x_2, x_3)$ and $y_2 = (x_3, x_4, x_5)$. This induces a factor graph whose variable nodes are the states x_l and whose factor nodes are the local trajectory segments y_j. Under the Bethe approximation, the resulting global density is $p(X) \approx \frac{\prod_{j=1}^{M} p(y_j)}{\prod_{l=1}^{L} p(x_l)^{d_l - 1}}$, where d_l is the number of local factors that contain x_l [26, 27].

We now specialize this construction to the decomposed MRA segments. Each trajectory segment $\boldsymbol{\tau}_i$ acts as a factor node, and consecutive segments $\boldsymbol{\tau}_i$ and $\boldsymbol{\tau}_{i+1}$ overlap at the milestone state O_i. For an MRA task, the unconditional prior therefore factorizes as

$$p(\boldsymbol{\tau}) \approx \frac{\prod_{i=1}^{N} p(\boldsymbol{\tau}_i)}{\prod_{i=1}^{N-1} p(O_i)}, \tag{13}$$

which simply corrects the over-counting of the shared milestone states. Figure 2 summarizes the resulting chain-structured graph and previews the lasso graph used later for prefix-suffix tasks.

Combining the decomposed likelihood in Eq. (12) with the prior in Eq. (13) and substituting them back into the Bayesian formulation in Eq. (10), we obtain:

$$p(\boldsymbol{\tau} \mid \Phi) \propto \frac{\prod_{i=1}^{N} [p(\boldsymbol{\tau}_i) p(\Phi_i \mid \boldsymbol{\tau}_i)]}{\prod_{i=1}^{N-1} p(O_i)}. \tag{14}$$

Since our generative process relies on the score function, we take the logarithm and the gradient with respect to the trajectory $\boldsymbol{\tau}$ on both sides of Eq. (14) to obtain the generic *Compositional MRA Score Function*:

$$\nabla \log p(\boldsymbol{\tau} \mid \Phi) \approx \sum_{i=1}^{N} \nabla \log p(\boldsymbol{\tau}_i) - \sum_{i=1}^{N-1} \nabla \log p(O_i) + \sum_{i=1}^{N} \nabla \log p(\Phi_i \mid \boldsymbol{\tau}_i). \tag{15}$$

Equation (15) separates into an unconditional compositional part, $\sum_{i=1}^{N} \nabla \log p(\boldsymbol{\tau}_i) - \sum_{i=1}^{N-1} \nabla \log p(O_i)$, and a conditional task-enforcement part, $\sum_{i=1}^{N} \nabla \log p(\Phi_i \mid \boldsymbol{\tau}_i)$. The overlap term is the score of an endpoint marginal, not the score of a one-state trajectory. Writing $\hat{\boldsymbol{\tau}}_i$ for all other states in the short segment, this marginal is $p(O_i) = \int p(O_i, \hat{\boldsymbol{\tau}}_i)\, d\hat{\boldsymbol{\tau}}_i$ (and analogously at each diffusion level). The pre-trained model supplies a score for the full short trajectory; obtaining the exact $\nabla_{O_i} \log p(O_i)$ would require marginalizing over the other states or training a separate marginal model. Evaluating the model on a length-one input, even if supported, instead describes a one-state trajectory distribution and does not give the endpoint marginal of a longer segment.

We therefore apply the existing neighboring-score averaging heuristic at each overlap [28, 29]: $\nabla_{O_i} \log p(\boldsymbol{\tau}) \approx \frac{1}{2}\left[\nabla_{O_i} \log p(\boldsymbol{\tau}_i) + \nabla_{O_i} \log p(\boldsymbol{\tau}_{i+1})\right]$. The end of segment i and the start of segment $i+1$ are the same state variable O_i from initialization onward, rather than two endpoints reconciled only after sampling. We draw each overlap state once at initialization and copy it to both segments. At every reverse step, we read both segment-score predictions at this shared variable, average them, update O_i once, and write the resulting value to both segment boundaries for the next step. The averaging is expected to work best when neighboring priors give compatible overlap scores within the shared target set and the training data cover the needed local transitions. Strong score disagreement, incompatible local modes, or sparse coverage can degrade the generated trajectory; the heuristic has no general error bound or dynamics-feasibility guarantee. The conditional MRA guarantee for a returned discrete plan follows from shared overlaps and final feasible-set projection, not from the accuracy of this score approximation.

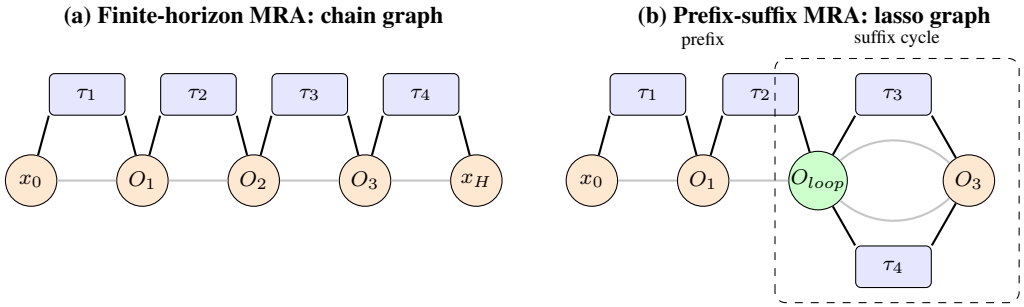


Figure 2: Factor-graph view of compositional trajectory generation. Left: a finite-horizon MRA task induces a chain graph, where circles denote milestone or overlap states and rounded rectangles denote local segment factors. Right: the prefix-suffix factor graph identifies O_p and O_N into the loop junction O_{loop}, producing a finite prefix chain followed by a recurrent suffix cycle (a lasso topology). Three segment-boundary score contributions meet at O_{loop}.

3.4 Projection-Based Conditioning for Hard Constraint Satisfaction

Now we focus on the computation of conditional task-enforcement part in Eq. (15). To handle this term, we use projection-based conditioning during reverse denoising, following recent projection-based approaches for constrained diffusion trajectory generation [22, 23]. Therefore, the task-enforcement part is realized by the correction that projects the current trajectory of each segment onto the nearest trajectory satisfying its corresponding Reach-Avoid specification. Equivalently, it is the projection offset from the unconstrained denoising update to the closest feasible segment.

Specifically, each local specification Φ_i induces the feasible set, with the first segment additionally constrained to start from the given initial state and each later segment constrained to start in the previous target region:

$$\mathcal{C}_i := \begin{cases} \{\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_1 \models \Phi_1,\ \tau_{1,0} = x_0\}, & i = 1, \\ \{\boldsymbol{\tau}_i \mid \boldsymbol{\tau}_i \models \Phi_i,\ \tau_{i,0} \in \mathcal{T}_{i-1}\}, & i = 2, \dots, N. \end{cases} \tag{16}$$

Given the current noisy segment $\boldsymbol{\tau}_i^j$, let $\bar{\boldsymbol{\tau}}_i^{j-1}$ denote the trajectory obtained after one unconditional denoising step. Full reach-avoid projection onto $\mathcal{C}_i$ takes the form

$$\boldsymbol{\tau}_i^{j-1} \in \arg\min_{\tilde{\boldsymbol{\tau}}_i \in \mathcal{C}_i} \left\| \tilde{\boldsymbol{\tau}}_i - \bar{\boldsymbol{\tau}}_i^{j-1} \right\|_2^2. \tag{17}$$

When the projection exists, this update selects a segment satisfying the sampled-state constraints of Φ_i that is closest to the denoised sample in Euclidean distance. For fixed milestone times, the current MRA feasible set is a product of admissible sets over time: each state is constrained by its stage-safe set, with additional initial and target constraints at the segment boundaries. Consequently, the Euclidean projection is *state-wise* (temporally separable). This property follows from the pointwise task constraints, not from the circular geometry of our benchmark regions; other geometries can also be used when their per-state projections are efficient. Constraints on velocity or acceleration computed from successive positions instead couple multiple time steps and require a joint trajectory-level projection. If velocity or acceleration is explicitly represented in each diffusion state and only bounded pointwise, those bounds can remain state-wise.

Algorithm 1 presents the full projection at each reverse step. In practice, it is not necessary to impose the avoid constraint from the beginning: projecting highly noisy early trajectories onto obstacle boundaries can make large, unhelpful corrections. Our implementation projects target endpoints throughout denoising and activates avoid projection only in the second half, after the coarse trajectory has formed. The final reverse step applies the full reach-avoid projection, which is sufficient for the conditional satisfaction guarantee for a returned plan.

The projection-based reverse denoising procedure is summarized in Algorithm 1. Starting from Gaussian noise for all segments, each reverse step combines compositional denoising using the unconditional part of Eq. (15), overlap reconciliation through the shared-state averaging rule, and projection of each segment onto its local feasible set. The resulting denoised segments are stitched

through the shared overlap states to form the full MRA trajectory. Theorem 1 establishes the soundness of Algorithm 1. The proof and a discussion of the assumptions are provided in Appendix D.

Algorithm 1 Projection-Based Conditional Reverse Denoising for MRA

Require: MRA sub-tasks $\{\Phi_i\}_{i=1}^N$, feasible sets $\{\mathcal{C}_i\}_{i=1}^N$, reverse horizon J, pre-trained diffusion model $\mathcal{M}_\theta$
Initialize $\boldsymbol{\tau}_i^J \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for all $i = 1, \ldots, N$
for $j = J, J-1, \ldots, 1$ **do**
  Use $\mathcal{M}_\theta$ together with the unconditional part of Eq. (15) to compute the one-step denoised trajectories $\{\bar{\boldsymbol{\tau}}_i^{j-1}\}_{i=1}^N$
  for $i = 1, 2, \ldots, N$ **do**
    Project $\bar{\boldsymbol{\tau}}_i^{j-1}$ onto $\mathcal{C}_i$ via Eq. (17); if no projection is returned, terminate without a trajectory
    Set the projected result as $\boldsymbol{\tau}_i^{j-1}$
  end for
end for
Stitch $\{\boldsymbol{\tau}_i^0\}_{i=1}^N$ through the shared overlap states to obtain $\boldsymbol{\tau}^0$
Ensure: Stitched discrete MRA trajectory $\boldsymbol{\tau}^0$ if all projections return

Theorem 1 (Soundness of Algorithm 1) *Let $\Phi = (0, H, \mathbb{T}, \mathbb{G})$ be an MRA task and let $x_0 \in \mathcal{S}$ be the initial state. For each $i = 1, \ldots, N-1$, assume that $\mathcal{T}_i \subseteq \mathcal{G}_i \cap \mathcal{G}_{i+1}$ and that the Euclidean projection onto $\mathcal{T}_i$ is single-valued. If Algorithm 1 returns a stitched discrete state trajectory $\boldsymbol{\tau}^0$, then $\tau_0^0 = x_0$ and $\boldsymbol{\tau}^0 \models \Phi$.*

Theorem 1 is a conditional guarantee about the returned plan. It does not assert that a feasible MRA task will be solved, that consecutive planned states admit a dynamically feasible transition, that motion between them is collision-free, or that a controller will track the plan safely. The success rates reported below instead measure empirical MRA satisfaction of controller-executed rollouts.

3.5 Generation of Prefix-Suffix Trajectories

We now extend the finite-horizon MRA setting to a prefix-suffix execution. In such a task, a finite prefix is executed once and a suffix cycle is then repeated indefinitely, as in automata-theoretic temporal-task synthesis [2, 30, 5]. For example, an inspection robot may first reach its assigned area and then repeatedly patrol inspection regions; a warehouse robot may complete an initial pickup and cycle between delivery and charging stations; a manipulator may finish initialization and then repeat an assembly or sorting operation. Let Φ be decomposed into N local Reach-Avoid sub-tasks $\Phi_1, \ldots, \Phi_N$, and fix an index $p \in \{1, \ldots, N-1\}$ that separates the one-time prefix from the periodic suffix. The prefix consists of $\Phi_1, \ldots, \Phi_p$, while the suffix consists of $\Phi_{p+1}, \ldots, \Phi_N$ and is required to repeat indefinitely.

A finite trajectory $\boldsymbol{\tau}$ is said to realize this prefix-suffix task if there exist milestone times

$$0 = m_0 \leq m_1 \leq \cdots \leq m_p \leq \cdots \leq m_N \tag{18}$$

such that $\boldsymbol{\tau}_i \models \Phi_i$ for all $i = 1, \ldots, N$, and the periodic boundary condition

$$\tau_{m_N} = \tau_{m_p} \tag{19}$$

holds. Under Eq. (19), the finite trajectory induces the infinite-horizon execution

$$\boldsymbol{\tau}^\omega = (\tau_0, \ldots, \tau_{m_p})\,(\tau_{m_p+1}, \ldots, \tau_{m_N})^\omega, \tag{20}$$

where $(\tau_{m_p+1}, \ldots, \tau_{m_N})^\omega$ denotes infinite repetition of the suffix cycle.

We call the resulting structure a *prefix-suffix factor graph*: the prefix segments form a finite factor chain, while the suffix segments form a cycle that represents indefinite repetition. Equation (19) identifies the milestone states at m_p and m_N as one loop junction O_{loop}, turning the original chain into a lasso topology, as shown in Fig. 2. The loop junction has three segment-boundary incidences: the end of $\boldsymbol{\tau}_p$, the start of $\boldsymbol{\tau}_{p+1}$, and the end of $\boldsymbol{\tau}_N$ (two incidences may belong to the same segment when the suffix has only one segment); hence $d_{loop} = 3$. Applying the same score-averaging heuristic as in Section 3.3, the compositional score at the loop junction becomes $\nabla_{O_{loop}} \log p(\boldsymbol{\tau}) \approx \frac{1}{3}\Big[\nabla_{\tau_{p,K_p}} \log p(\boldsymbol{\tau}_p) + \nabla_{\tau_{p+1,0}} \log p(\boldsymbol{\tau}_{p+1}) + \nabla_{\tau_{N,K_N}} \log p(\boldsymbol{\tau}_N)\Big]$, with each

gradient evaluated at its corresponding boundary copy of O_{loop}. Compared with the finite-horizon chain, the only structural difference is therefore at the loop junction: all other segments are handled exactly as before, while the shared state O_{loop} now aggregates three neighboring score contributions rather than two. Consequently, the same short-horizon priors, projection operator, and reverse-denoising routine can be reused directly, and the prefix-suffix extension is realized entirely by changing the overlap topology at sampling time rather than by retraining a new model. Therefore, generating a single finite trajectory that satisfies Eq. (19) is sufficient to define an infinite-horizon prefix-suffix execution.

4 Experiments

The main goal of our experiments is to evaluate whether the proposed composition-projection framework can use only short-horizon, task-agnostic diffusion priors to solve long-horizon multiple reach-avoid problems with reliable actual execution. Each environment's diffusion prior is trained once on its benchmark offline data and then frozen for all MRA tasks and comparisons; Appendix C.6 details the backbone and sampling settings. In particular, we aim to examine three questions. First, can the method maintain strong rollout-level task satisfaction and planning efficiency as the complexity of environment geometry and the number of reach-avoid stages increase? Second, does the same compositional viewpoint extend naturally from finite-horizon MRA tasks to prefix-suffix tasks with repeated suffix cycles? Third, can the framework transfer beyond simple planar navigation to domains with substantially richer underlying dynamics, such as locomotion and manipulation? To answer these questions, we conduct three groups of experiments: a comparative benchmark on Maze2D against guidance-based baselines, a prefix-suffix evaluation on Maze2D, and transfer experiments on AntMaze and Cube. All experiments were run on a PC with an Intel i9-10940X CPU and Nvidia 3090 GPU.

4.1 Comparative Results on Maze2D

Experimental settings. We evaluate the main benchmark on the three Maze2D layouts—*U-Maze*, *Medium*, and *Large*—using randomized multiple reach-avoid (MRA) instances constructed from ordered target regions and obstacle regions. All methods use diffusion priors trained only on the task-agnostic D4RL trajectories [31], generate state sequences at planning time, and execute them with the same simple PD controller. The column **Seg.** in Table 1 denotes the number of local reach-avoid stages in the task, so larger values correspond to longer-horizon MRA problems. For each environment and each segment setting, we evaluate 200 randomly generated task instances. We use DoG-SW as the primary guidance-based baseline because, like our method, it decomposes the MRA task into stage-wise reach-avoid problems before planning each segment. We also report DoG-OS, which applies the same LTLDoG guidance mechanism [32] once to a single long trajectory. Comparing DoG-OS with DoG-SW mainly illustrates the effect of task decomposition within guidance-based planning; comparing our method with DoG-SW examines the additional effect of our composition and projection procedure under a stage-wise task formulation. More experimental details are provided in Appendix C.2.

Evaluation metrics. We report two metrics in Table 1. **Success Rate** is the fraction of controller-executed rollouts whose sampled states satisfy the full ordered MRA task; it is an empirical execution metric, distinct from the theorem's conditional claim about returned plans. **Planning Time** is the wall-clock time required to generate a plan with each method. Together, these metrics measure observed task completion under the controller and planning cost.

Results and analysis. We first compare our method with the primary guidance baseline, DoG-SW, since both methods solve the task in a stage-wise manner. In the simplest environment, U-Maze, DoG-SW remains quite competitive: its success rate stays close to ours across all four segment cases. However, as the environment becomes more complex, the difference grows substantially. In Medium, the gap increases from 0.080 in the one-segment case to 0.170 in the four-segment case, and in Large it grows from 0.105 to 0.250. These results suggest that stage-wise guidance is reasonably effective in easy settings, but its performance degrades more noticeably than ours as both the geometry and the horizon become more challenging.

The comparison between the two guidance variants separately shows the role of decomposition. Moving from DoG-OS to DoG-SW introduces decomposition, and this change clearly improves

Table 1: Comparative results on Maze2D. Success rate measures MRA satisfaction of controller-executed rollouts, not the theorem's guarantee for returned discrete plans. Planning time is reported as mean $\pm$ standard deviation across evaluation cases, in seconds. **Seg.** denotes the number of local reach-avoid stages in the MRA task. DoG-OS denotes the one-shot LTLDoG baseline, and DoG-SW denotes the stage-wise LTLDoG baseline.

		Success Rate ↑			Planning Time ↓		
Dataset	Seg.	Ours	DoG-OS	DoG-SW	Ours	DoG-OS	DoG-SW
U-Maze	1	**0.950**	**0.950**	0.945	$\mathbf{0.534 \pm 0.018}$	7.483 ± 0.162	7.589 ± 0.082
	2	**0.925**	0.630	0.910	$\mathbf{0.598 \pm 0.019}$	8.440 ± 0.076	11.522 ± 2.519
	3	**0.895**	0.275	0.865	$\mathbf{0.637 \pm 0.017}$	9.171 ± 0.069	15.242 ± 3.374
	4	**0.840**	0.055	0.825	$\mathbf{0.703 \pm 0.023}$	10.322 ± 0.065	19.167 ± 4.697
Medium	1	**0.915**	0.845	0.835	$\mathbf{0.970 \pm 0.072}$	12.679 ± 0.099	12.747 ± 0.114
	2	**0.845**	0.195	0.745	$\mathbf{1.350 \pm 0.210}$	14.263 ± 0.089	18.192 ± 3.907
	3	**0.725**	0.035	0.670	$\mathbf{1.550 \pm 0.230}$	15.659 ± 0.122	23.105 ± 5.466
	4	**0.715**	0.000	0.545	$\mathbf{1.940 \pm 0.570}$	17.658 ± 0.159	27.221 ± 7.614
Large	1	**0.825**	0.710	0.720	$\mathbf{1.048 \pm 0.172}$	12.978 ± 0.144	12.696 ± 0.163
	2	**0.710**	0.130	0.515	$\mathbf{1.490 \pm 0.270}$	14.310 ± 0.142	20.499 ± 4.166
	3	**0.700**	0.005	0.405	$\mathbf{2.000 \pm 0.700}$	15.717 ± 0.168	26.386 ± 6.512
	4	**0.645**	0.000	0.395	$\mathbf{2.290 \pm 0.810}$	17.303 ± 0.114	34.979 ± 8.447

success in the more difficult settings: for example, in Large the success rate rises from 0.13 to 0.515 in the two-segment case, from 0.005 to 0.405 in the three-segment case, and from 0 to 0.395 in the four-segment case. Similar improvements also appear in Medium. Thus, the contrast between DoG-OS and DoG-SW provides direct empirical support for the decomposition strategy used in our method, showing that breaking a long MRA task into shorter local subproblems is already a key ingredient for improving task completion in complex long-horizon settings. At the same time, DoG-SW still underperforms our method throughout the table, with the gap becoming substantial on the most difficult settings. This suggests that, beyond task decomposition, our combined overlap composition and projection procedure improves empirical rollout reliability in these settings.

Planning efficiency shows an equally strong advantage for our approach. Across all twelve settings, our planning time ranges from 0.534 to 2.29 seconds, while DoG-OS requires roughly 7.5 to 17.7 seconds and DoG-SW ranges from about 7.6 to 35 seconds. In other words, our method has higher rollout success than DoG-SW in all twelve settings and is more than ten times faster than either guidance baseline on average. Therefore, with the efficiently computable state-wise projectors used in these benchmarks, projection-based enforcement is substantially cheaper than guidance-based sampling while also yielding higher empirical rollout success. The observed speed advantage does not establish the same cost for trajectory-coupled constraints that require joint projection.

Aggregating over all twelve Maze2D settings, our method achieves an average rollout success rate of 80.8%, compared with 69.8% for DoG-SW and 31.9% for DoG-OS. At the same time, the average planning time is 1.26 seconds for our method, versus 13.0 seconds for DoG-OS and 19.1 seconds for DoG-SW, corresponding to $10.3\times$ and $15.2\times$ speedups, respectively. This compact summary reinforces the main empirical message of Table 1: decomposition and projection are associated with higher observed rollout success and substantially lower planning time in these benchmark settings. These empirical rollout results are separate from the guarantee for returned discrete plans. Additional Maze2D experiments examine non-circular target and obstacle geometry (Appendix C.3.1), CDGS overlap inference under otherwise matched settings (Appendix C.3.2), sensitivity to local horizon allocation (Appendix C.3.3), and optional smoothness guidance (Appendix C.3.4).

4.2 Prefix-Suffix Tasks on Maze2D

We further evaluate the prefix-suffix extension of our framework on the three Maze2D layouts, where each task consists of a one-time prefix followed by a recurrent suffix cycle. Here, a trial is counted successful only if the executed rollout completes the one-time prefix and then realizes the suffix loop twice while satisfying the corresponding avoid constraints throughout. We use two loops rather than one because a single traversal can be achieved by following the finite lasso plan once, whereas two

successful traversals provide a stronger check that the learned loop is genuinely repeatable under closed-loop execution. Using the same task-agnostic priors and the same PD execution controller as in the finite-horizon Maze2D benchmark, the method achieves consistently strong **success rates** in U-Maze, ranging from 76.5% to 85.5%, with **planning times** between 0.649 and 0.760 seconds across the tested prefix-suffix settings. Performance degrades in the more complex Medium and Large layouts, but remains robust overall, with success rates between 69.0% and 78.0% in Medium and between 54.5% and 74.0% in Large, while planning times stay within 1.378 to 1.802 seconds and 1.557 to 2.092 seconds, respectively. These results indicate that the proposed composition-projection framework extends naturally from finite-horizon MRA planning to prefix-suffix tasks, while also showing that longer repeated suffix cycles become increasingly demanding in harder environments. Full experimental settings, detailed results, and analyses are provided in Appendix C.4.

4.3 Results in Cube and AntMaze

We further evaluate the generality of our framework on two dynamics-rich OGBench domains [33]: Cube (6-DoF UR5e manipulator) and AntMaze (8-DoF Ant). As in the Maze2D experiments, training uses only task-agnostic offline trajectories, while evaluation replaces the original goal-conditioned objectives with multi-stage MRA tasks. In Cube, we plan and track end-effector trajectories in Cartesian space; in AntMaze, we plan in the two-dimensional workspace and execute with an inverse-dynamics controller. Overall, the framework achieves consistently strong **success rates** in Cube, ranging from 93.8% to 99.2%, with **planning times** between 0.868 and 1.382 seconds across the tested obstacle settings. AntMaze is more challenging, but the method still maintains robust success rates between 74.1% and 95.1%, while planning times remain within 1.018 to 2.443 seconds. The performance gap between the two domains is consistent with how the geometric plans are executed. In Cube, planning is carried out directly in end-effector Cartesian space, so the local reach-avoid constraints align closely with the commanded motion and the controller mainly has to track relatively smooth manipulator trajectories. In AntMaze, the same low-dimensional workspace plan must be realized by contact-rich locomotion with strongly coupled body dynamics, so small rollout errors are harder to correct once the feasible corridor narrows. These results indicate that the proposed composition-projection framework transfers beyond simple navigation to higher-dimensional manipulation and locomotion domains. Full experimental settings, detailed segment-wise results, and analyses are provided in Appendix C.5.

5 Conclusion

We presented a data-driven compositional framework for long-horizon multiple reach-avoid (MRA) planning using a pre-trained short-horizon diffusion model. Its MRA-specific contribution combines stage-wise reach-avoid projection and shared milestones with established factor-graph composition and neighboring-score averaging, and extends this construction to prefix-suffix tasks. The finite-horizon theorem guarantees sampled-state MRA satisfaction only for a returned stitched discrete plan. In simulation, controller-executed rollouts show strong empirical success on Maze2D and its prefix-suffix tasks with substantially lower planning time than guidance-based baselines. Cube and AntMaze likewise test low-dimensional task-space plans—Cartesian end-effector positions and a two-dimensional workspace, respectively—executed by controllers under more complex simulated manipulation and locomotion dynamics.

The method requires structured, task-relevant state constraints with efficient projection operators and an offline prior covering the local motions needed for new tasks; it does not directly plan or project in raw visual observation space. Fixed segment lengths can miss feasible milestone allocations, and neighboring-score averaging is a heuristic without a general error bound. The discrete-plan guarantee does not establish dynamic feasibility, collision-free motion between sampled states, or safe controller execution. Future work should address data coverage, adaptive stage lengths, overlap inference, and dynamics-aware projection and execution.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (62573291, 62533017).

References

[1] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *1st Symposium in Logic in Computer Science (LICS)*. IEEE Computer Society, 1986.

[2] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[3] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 89.

[4] X. Yin, B. Gao, and X. Yu, "Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges," *Annual Reviews in Control*, vol. 57, p. 100940, 2024.

[5] Y. Chen, J. Tumova, A. Ulusoy, and C. Belta, "Temporal logic robot control based on automata learning of environmental dynamics," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 547–565, 2013.

[6] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 2107–2116.

[7] P. Vaezipoor, A. C. Li, R. A. T. Icarte, and S. A. Mcilraith, "Ltl2action: Generalizing ltl instructions for multi-task rl," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 10 497–10 508.

[8] Z. Feng, H. Luan, K. Y. Ma, and H. Soh, "Diffusion meets options: Hierarchical generative skill composition for temporally-extended tasks," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 10 854–10 860.

[9] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6565–6570.

[10] H. Venkataraman, D. Aksaray, and P. Seiler, "Tractable reinforcement learning of signal temporal logic objectives," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 308–317.

[11] S. Wang, S. Li, L. Yin, and X. Yin, "Synthesis of temporally-robust policies for signal temporal logic tasks using reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 503–10 509.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[13] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[14] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9902–9915.

[15] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10–11, pp. 1684–1704, 2025.

[16] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1916–1923.

[17] N. Botteghi, F. Califano, M. Poel, and C. Brune, "Trajectory generation, control, and safety with denoising diffusion probabilistic models," *arXiv preprint arXiv:2306.15512*, 2023.

[18] K. Mizuta and K. Leung, "Cobl-diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 801–13 808.

[19] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8780–8794, 2021.

[20] Z. Zhong, D. Rempe, D. Xu, Y. Chen, S. Veer, T. Che, B. Ray, and M. Pavone, "Guided conditional diffusion for controllable traffic simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3560–3566.

[21] W. Xiao, T.-H. Wang, C. Gan, R. Hasani, M. Lechner, and D. Rus, "Safediffuser: Safe planning with diffusion probabilistic models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[22] J. K. Christopher, S. Baek, and F. Fioretto, "Constrained synthesis with projected diffusion models," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.

[23] J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto, "Simultaneous multi-robot motion planning with projected diffusion models," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267. PMLR, 2025, pp. 37 162–37 180.

[24] Y. Chen, S. Li, and X. Yin, "Control synthesis for multiple reach-avoid tasks via hamilton-jacobi reachability analysis," in *2025 IEEE 64th Conference on Decision and Control (CDC)*, 2025, pp. 5980–5985.

[25] J. Choi, K. A. Pant, Y. Nam, H. Hellmann, K. Nune, and I. Hwang, "Signal temporal logic verification and synthesis using deep reachability analysis and layered control architecture," *arXiv preprint arXiv:2602.23313*, 2026.

[26] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[27] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, 2005.

[28] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, "Compositional visual generation with composable diffusion models," in *European Conference on Computer Vision*, 2022, pp. 423–439.

[29] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *Proceedings of The 7th Conference on Robot Learning*, vol. 229, 2023, pp. 2905–2925.

[30] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[31] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[32] Z. Feng, H. Luan, P. Goyal, and H. Soh, "Ltldog: Satisfying temporally-extended symbolic constraints for safe diffusion-based planning," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8571–8578, 2024.

[33] S. Park, K. Frans, B. Eysenbach, and S. Levine, "OGBench: Benchmarking offline goal-conditioned RL," in *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[34] S. Zhou, Y. Du, S. Zhang, M. Xu, Y. Shen, W. Xiao, D.-Y. Yeung, and C. Gan, "Adaptive online replanning with diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[35] Y. Meng and C. Fan, "Diverse controllable diffusion policy with signal temporal logic," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8354–8361, 2024.

[36] R. Liu, A. Hou, X. Yu, and X. Yin, "Zero-shot trajectory planning for signal temporal logic tasks," in *Advances in Neural Information Processing Systems*, vol. 38, 2025.

[37] C. Chen, F. Deng, K. Kawaguchi, C. Gulcehre, and S. Ahn, "Simple hierarchical planning with diffusion," in *International Conference on Learning Representations*, 2024.

[38] Y. Luo, U. A. Mishra, Y. Du, and D. Xu, "Generative trajectory stitching through diffusion composition," in *Advances in Neural Information Processing Systems*, vol. 38, 2025.

[39] U. A. Mishra, D. He, Y. Chen, and D. Xu, "Compositional diffusion with guided search for long-horizon planning," in *International Conference on Learning Representations*, 2026.

[40] U. A. Mishra, Y. Chen, and D. Xu, "Generative factor chaining: Coordinated manipulation with diffusion-based factor graph," in *Proceedings of The 8th Conference on Robot Learning*, vol. 270, 2025, pp. 1461–1472.

[41] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

A Related Works

A.1 Diffusion Planning and Constrained Generation

Diffusion-based methods have become an important data-driven paradigm for robotics planning and control, with prior works showing that trajectory generation from offline data can support a broad range of downstream planning tasks without explicit system identification [14, 15, 16, 17, 18, 34]. A key attraction of this paradigm is that the reverse denoising process can be conditioned at test time in several ways. One way is to steer reverse denoising with gradients of downstream task objectives so that sampling is biased toward high-value or low-cost trajectories: in [14], planning is formulated as guided sampling, where gradients of cumulative reward bias reverse denoising and inpainting is used to impose start-goal trajectory constraints, while [16] similarly performs classifier-guidance-style sampling using differentiable motion-planning objectives such as trajectory smoothness costs and obstacle-avoidance penalties. Another way is to treat constraints as hard feasibility conditions and explicitly project each denoising iterate back to a feasible set during sampling: [22] introduces projected diffusion for general constrained synthesis, and [23] brings the same idea to multi-robot motion planning under collision-avoidance and kinematic-feasibility constraints. These methods demonstrate the promise of conditioning diffusion planners toward rewards, goals, collision avoidance, and hard safety constraints, but they still typically address a single global objective or a static constraint structure. They do not directly handle our setting, where a long trajectory must satisfy a temporally ordered sequence of reach-avoid constraints and be generated compositionally from overlapping short-horizon priors.

A.2 Diffusion Planning Under Temporally Extended Constraints

Several recent works combine diffusion models with temporally extended symbolic constraints in both LTL and STL settings [32, 8, 20, 35, 36]. For example, [32] uses differentiable regressors as sampling-time guidance to bias denoising toward trajectories satisfying LTL_f instructions. For STL constraints, [20] guides diffusion sampling with gradients of STL robustness values to favor rule-compliant trajectories, while [35] considers STL-conditioned diffusion policy learning with synthetic data augmentation to produce diverse behaviors indexed by STL parameters. [36] addresses a broader class of finite-horizon STL specifications, including explicit temporal intervals. It decomposes the formula into progress and time constraints, searches for timed waypoints, and uses a pre-trained diffusion model to generate and stitch connecting segments. Our method instead specializes to untimed MRA tasks with a prescribed target order: target milestones remain set-valued and are shared across neighboring segments during denoising, while projection enforces the local reach-avoid constraints. This avoids timed-waypoint search and an additional time-allocation model, but it does not support explicit STL timing constraints and currently requires pre-specified local segment lengths. A different length allocation may succeed where the fixed one fails, so the restriction limits completeness rather than the soundness of a returned plan.

A.3 Long-Horizon Diffusion Planning and Trajectory Composition

Another related line of work studies structured ways to extend diffusion planning beyond the horizon seen in training [37, 38, 29, 39, 40]. Among these directions, [37] proposes a hierarchical planner in which a high-level diffuser generates sparse subgoals and a low-level diffuser refines them, thereby improving temporal abstraction on long-horizon tasks. In a different direction, [38] composes overlapping short trajectory chunks to synthesize longer coherent plans, explicitly targeting long-horizon generalization from short-horizon data. Closer to our graphical perspective, [40] formulates coordinated manipulation as a spatial-temporal factor graph with modular diffusion factors. Taken together, these works show that modular structure can substantially extend the effective horizon of diffusion-based generation. However, they are not developed for temporally ordered reach-avoid semantics with explicit stage-wise feasible sets and formal satisfaction guarantees. We use these established composition tools with factors induced by MRA reach-avoid stages and variables identified with shared milestones. The MRA-specific projection, overlap-preserving stitching, prefix-suffix construction, and conditional planned-satisfaction analysis distinguish the resulting framework.

B Limitations

While the proposed framework offers a clean compositional perspective on long-horizon MRA planning, several limitations remain.

Planned-feasibility versus execution-feasibility. Our formal guarantee applies only when the sampler returns a stitched discrete state trajectory: its sampled states satisfy the MRA target and safe-set constraints. It does not require or establish that the diffusion prior captures dynamically feasible transitions. Consecutive planned states may not be connected by admissible controls, and the continuous motion between them may intersect obstacles. Actuation limits, modeling mismatch, and execution noise can further cause the controller-executed rollout to depart from the plan and fail the MRA task. The reported rollout success rates measure this execution outcome empirically, without certifying safe execution.

Timed/untimed gap between MRA semantics and implementation. The semantic definition of MRA is existential in the milestone times: a trajectory satisfies the task as long as there exists some milestone sequence under which each reach-avoid stage is completed. In contrast, our algorithm assigns each local reach-avoid stage a same pre-specified finite segment length, which fixes a timing allocation in advance. Therefore, the current implementation does not realize the full existential-milestone MRA semantics: a task may be feasible for some milestone allocation but remain unrecovered under an unfavorable choice of segment lengths. A stronger algorithm should adapt the length of each stage to the reach-avoid task itself, for example by jointly planning the trajectory and the milestone-time allocation or by dynamically adjusting local horizons during sampling.

Approximate long-horizon score composition. Our compositional prior relies on a Bethe-style factorization of the long trajectory and replaces the unavailable overlap marginal scores with heuristic score averaging. This approximation is practical but not exact and has no general error bound. It can degrade when neighboring segment priors disagree strongly at an overlap, the required local transitions are poorly covered by training data, overlap states fail to summarize inter-segment coupling, or the number of stages grows substantially.

No optimality guarantee for trajectory quality metrics. The default sampler uses the learned prior to favor plausible local motion while projecting sampled states toward the specified reach-avoid constraints; it does not explicitly minimize path length, control effort, energy consumption, or traversal time. Optional differentiable quality guidance can reduce a chosen empirical cost (Appendix C.3.4), but adds computation and does not guarantee a global optimum.

Dependence on local data coverage. The method assumes that short-horizon offline data contain enough local motion diversity to support the transitions required by downstream long-horizon MRA tasks. If the required maneuvers are weakly represented in the dataset, the learned diffusion prior may provide poor support for successful composition.

Computational burden of projection oracles. The measured planning-time advantage depends on constraints that admit efficient state-wise projection. Our benchmark target and obstacle regions have analytic per-state projectors, so the full trajectory projection separates across time steps and is inexpensive. More complex state-set geometry can still be handled when its per-state projector is efficient, but difficult high-dimensional state sets may increase the cost. Velocity, acceleration, or smoothness constraints defined through differences between consecutive positions are temporally coupled and require joint trajectory-level projection; such constraints can substantially reduce the speed advantage. Pointwise bounds on velocity or acceleration can remain state-wise if those quantities are explicit components of each diffusion state.

C Details of Experiments

C.1 Environments

We conduct experiments on benchmark domains covering planar navigation, complex locomotion, and robotic manipulation. This setup tests whether the same composition-projection pipeline remains effective as the problem moves from low-dimensional geometric motion to higher-dimensional control. In all cases, training uses only the task-agnostic offline trajectories released with the original benchmarks. At evaluation time, we replace the original goal-conditioned formula-

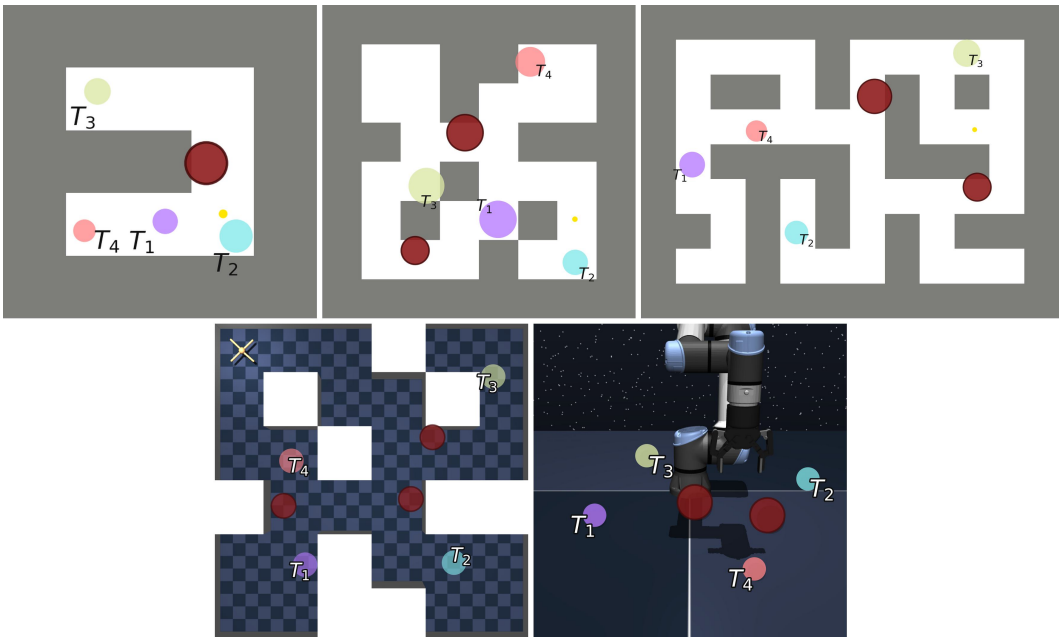


Figure 3: Visualization of the environments used in our experiments. From left to right and top to bottom: the three Maze2D environments (*U-Maze*, *Medium*, *Large*), the AntMaze environment, and the Cube environment.

tions with ordered multiple reach-avoid (MRA) tasks. Visualizations of all environments are shown in Figure 3.

- **D4RL Maze2D [31] (*U-Maze*, *Medium*, *Large*)**. Maze2D is our primary navigation testbed because it isolates the long-horizon planning challenge from complicated robot dynamics. Starting from the original start-goal benchmark, we sample an ordered set of target regions in the maze and introduce a prescribed number of obstacle regions that must be avoided throughout execution. Each instance is then converted into an MRA task in which the point agent reaches the targets in sequence while its evaluated states avoid the obstacle regions. The three layouts provide a controlled increase in geometric complexity, allowing us to examine whether the proposed composition-projection framework can reliably extend short-horizon diffusion priors to longer constrained plans.
- **OGBench AntMaze [33]**. AntMaze applies the same target-and-obstacle construction in a locomotion domain with substantially richer dynamics. We define tasks in the ant's workspace by selecting ordered target regions together with obstacle regions, and require the robot to visit the targets in sequence while avoiding those obstacles. Because feasible motion depends strongly on the coupling between the ant's internal configuration and its global displacement, this environment tests whether our segmented diffusion prior and compositional score coupling remain effective beyond simple kinematic navigation.
- **OGBench Cube [33]**. Cube serves as the manipulation-domain counterpart of the navigation benchmarks. Instead of relying on the original object-level success labels, we formulate planning directly in Cartesian task space through the end-effector trajectory. We then construct long-horizon MRA tasks by specifying ordered target regions together with obstacle regions in that space, so that the robot must complete sequential reaching motions while avoiding forbidden regions. This setting evaluates whether the same framework transfers to manipulation problems with higher-dimensional continuous control.

C.2 Details of Experiment in Maze2d Environment

Baseline Algorithm. As stated in Section 4.1, our primary guidance baseline is DoG-SW, which applies the LTLDoG guidance mechanism [32] after decomposing the task into stage-wise subproblems. We also report DoG-OS, which guides a single long trajectory for the whole MRA task, to

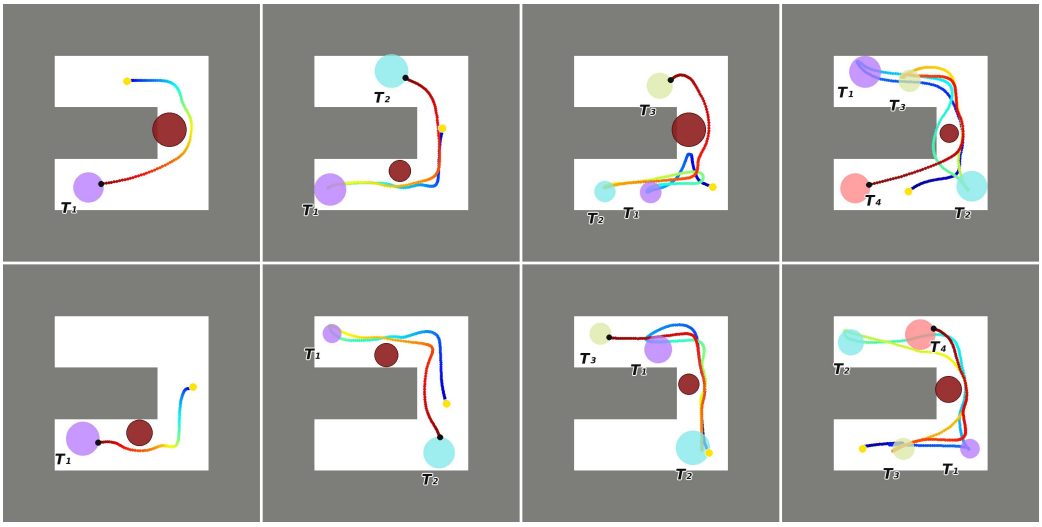


Figure 4: Representative rollout results of our method on U-Maze under selected 1-, 2-, 3-, and 4-segment settings.

show the effect of decomposition within the same guidance family. The key difference is how the MRA objective is encoded as an LTL_f instruction. For DoG-SW, each stage-i reach-avoid subtask is written as a local formula of the form $\varphi_i^{\mathrm{SW}} = p_{\mathcal{G}_i} \mathrm{U} p_{\mathcal{T}_i}$, meaning that the trajectory should remain in the stage-safe region until it reaches the current target region; LTLDoG guidance is then applied separately to each segment. For DoG-OS, the whole MRA task is instead written as one nested LTL_f formula, e.g., $\varphi^{\mathrm{OS}} = p_{\mathcal{G}_1} \mathrm{U}(p_{\mathcal{T}_1} \wedge (p_{\mathcal{G}_2} \mathrm{U}(p_{\mathcal{T}_2} \wedge \cdots (p_{\mathcal{G}_N} \mathrm{U} p_{\mathcal{T}_N}) \cdots)))$, and the same guidance is applied once over the full horizon. Following LTLDoG, these LTL_f instructions are enforced during diffusion by a differentiable satisfaction score: atomic propositions are first assigned differentiable real-valued labels from the predicted clean trajectory, and the soft formula evaluator in LTLDoG then provides the guidance term $\nabla_\tau f(\varphi, L(\hat{\tau}_0))$ used at each reverse denoising step.

Experimental Settings. As stated in Section 4.1, we evaluate randomized MRA instances on the three Maze2D layouts—*U-Maze*, *Medium*, and *Large*—using diffusion priors trained on the task-agnostic D4RL trajectories [31], and all methods generate state sequences that are executed with the same simple PD controller. Here we specify the instance construction in more detail. For each environment and each segment setting, we evaluate 200 randomly generated test cases. In these cases, each U-Maze instance contains one obstacle region, while each Medium and Large instance contains two obstacle regions. In all these Maze2D settings, each target region and each obstacle region is represented as a circular disk with fixed radius. Consequently, whenever a target or obstacle constraint is violated during projection-based conditioning, the corresponding Euclidean projection is computed analytically as the nearest point on the relevant circular boundary.

More Cases. To complement the quantitative comparison in Table 1, we provide additional representative rollout visualizations for our method in Figures 4, 5, and 6. Each figure corresponds to one Maze2D layout and contains eight rollout examples: two executed trajectories for each of the 1-, 2-, 3-, and 4-segment settings. These visualizations provide a qualitative supplement to the success-rate and planning-time statistics in the main benchmark, illustrating how the composed plans are executed under progressively longer MRA horizons in the three layouts. For some failed cases, the trajectories generated by our algorithm are shown in Figure 7.

Failure-case Analysis. By inspecting the failed cases, we find that the dominant issue is not simply a large mismatch between the executed rollout and the diffusion-generated plan; rather, the diffusion-generated trajectory itself can already be inconsistent with the actual system dynamics. Figure 7 highlights two recurring forms of dynamic infeasibility: penetration into obstacle regions and unrealistically large gaps between consecutive states. We attribute this dynamic infeasibility to two factors. First, the diffusion prior itself may occasionally generate trajectories that do not faithfully follow the underlying dynamics. Second, when newly generated obstacle regions must be avoided, the denoising samples are forcibly projected toward the boundary of the safe region, and

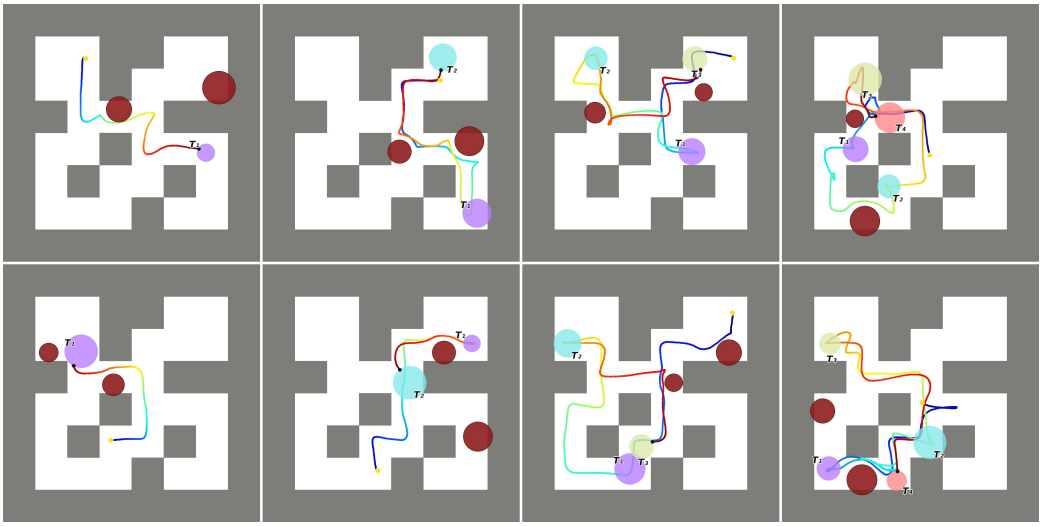


Figure 5: Representative rollout results of our method on Medium under selected 1-, 2-, 3-, and 4-segment settings.

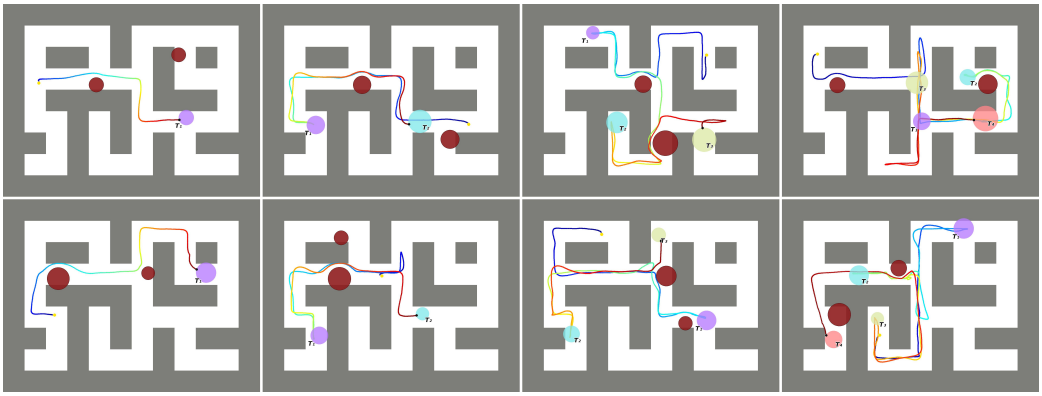


Figure 6: Representative rollout results of our method on Large under selected 1-, 2-, 3-, and 4-segment settings.

this projection shifts the trajectory away from the original data distribution, increasing the chance of producing dynamically implausible motions. Since our formulation assumes that the diffusion prior is given in advance, the second factor is the one that should be addressed more directly at the algorithmic level. A natural direction for future work is therefore to design denoising and projection procedures that preserve the original data distribution as much as possible while explicitly accounting for dynamic feasibility during projection.

C.3 Additional Maze2D Experiments

C.3.1 Complex constraint geometries and state-wise projection

The circular regions in the main benchmark admit radial projections. To test the dependence on this simple geometry, we additionally consider rectangular, convex polygonal (polyhedral in state space), capsule-shaped, and non-convex L-, U-, and T-shaped target and obstacle regions on Maze2D Medium and Large, with $N = 2, 3, 4$ segments. These remain pointwise state constraints, so projection across a trajectory separates over time; the cost of each state-wise projection nevertheless depends on the shape.

For an axis-aligned rectangle, the nearest point in the region is obtained by clipping each spatial coordinate to the corresponding interval; a state inside a forbidden rectangle is moved to its nearest face. For a convex polygon, we project onto each edge segment and select the nearest admissible

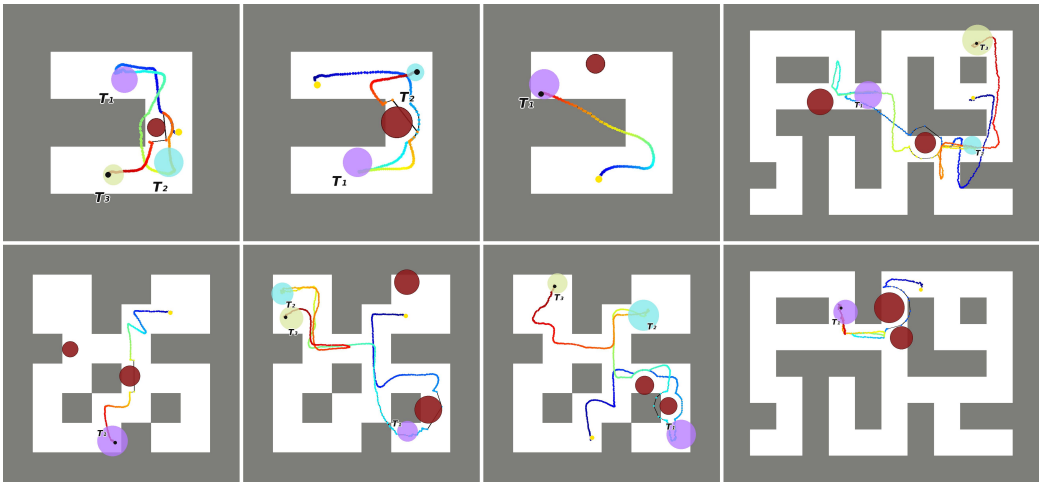


Figure 7: Representative failed cases in Maze2D. The figure contains eight examples from U-Maze, Medium, and Large, showing diffusion-generated trajectories whose corresponding rollouts fail to satisfy the intended MRA task under execution.

Table 2: Maze2D results with complex constraint geometries. Feasibility is measured on controller-executed rollouts. Planning time is mean $\pm$ standard deviation across evaluation cases, in seconds.

Map	N	Ours feasible	Ours time	DoG-SW feasible	DoG-SW time
Medium	2	82.0%	5.01 ± 3.05	70.5%	20.40 ± 4.59
Medium	3	72.5%	7.35 ± 4.49	64.5%	26.40 ± 6.53
Medium	4	66.0%	9.14 ± 6.71	56.5%	32.18 ± 8.66
Large	2	75.0%	7.06 ± 4.38	54.0%	21.63 ± 4.19
Large	3	70.5%	9.79 ± 5.95	49.5%	29.92 ± 7.14
Large	4	56.5%	12.59 ± 10.53	37.5%	38.13 ± 8.67

boundary point. For a capsule, we first find the nearest point on its centerline segment and then move radially to the capsule boundary if the state violates the target or avoid condition. The L-, U-, and T-shaped regions are represented as unions of simple rectangular components: we enumerate their exposed boundary segments, project onto these segments, and select the closest valid point. A fixed tie rule selects the same boundary point for a shared milestone whenever nearest points are not unique. Target projection leaves states already in the target set unchanged, while avoid projection leaves states outside the forbidden set unchanged. These operations require finitely many elementary geometric checks per state, rather than a joint trajectory-level optimization.

Table 2 shows an 8–21 percentage-point rollout-feasibility advantage over DoG-SW, with approximately 3.0–4.1 times lower mean planning time. The mean planning times are nevertheless higher than those of our circular-region benchmark in Table 1: more geometric components and candidate boundary checks make the per-state projector more expensive. Because the constraint geometry also changes the task, this comparison does not isolate projector runtime from every other source of timing variation. The speed advantage thus depends on having an efficient state-wise projector; constraints coupling consecutive states would require joint projection.

C.3.2 Controlled comparison with CDGS

Neighboring-score averaging approximates the unavailable overlap-marginal score without additional training or inference-time search. CompDiffuser [38] learns conditional relationships between overlapping segments and therefore requires a separately trained component. For a comparison under a shared frozen prior, we instead replace only our overlap-inference mechanism with CDGS [39], which uses population-based inference-time search and resampling to reject incompatible overlap

Table 3: Overlap-inference comparison on Maze2D. Feasibility is measured on executed rollouts; planning time is mean $\pm$ standard deviation over cases with a recorded planning time, in seconds.

Map	N	Averaging feasible	Averaging time	CDGS feasible	CDGS time
Medium	2	84.5%	1.35 ± 0.21	82.0%	7.95 ± 3.47
Medium	3	72.5%	1.55 ± 0.23	76.5%	11.23 ± 5.77
Medium	4	71.5%	1.94 ± 0.57	70.5%	13.94 ± 5.56
Large	2	71.0%	1.49 ± 0.27	78.0%	11.02 ± 4.00
Large	3	70.0%	2.00 ± 0.70	62.5%	15.93 ± 6.96
Large	4	64.5%	2.29 ± 0.81	56.0%	20.82 ± 10.83
Mean	–	72.3%	1.77	70.9%	13.48

Table 4: Maze2D horizon-allocation ablation: rollout feasibility (%) / mean planning time (s). H is the training-window horizon; the distance-proportional allocation keeps the total horizon NH.

Map	N	$0.6H$	$0.8H$	$1.0H$	$1.2H$	$1.4H$	Distance-proportional
Medium	2	72.5 / 1.13	82.0 / 1.24	84.5 / 1.35	85.5 / 1.41	83.5 / 1.44	87.5 / 2.02
Medium	3	60.5 / 1.32	74.5 / 1.43	72.5 / 1.55	77.5 / 1.69	73.0 / 1.79	77.5 / 3.00
Medium	4	50.0 / 1.53	64.0 / 1.72	71.5 / 1.94	71.0 / 2.12	67.5 / 2.26	72.0 / 4.16
Large	2	62.5 / 1.29	77.5 / 1.44	71.0 / 1.49	61.5 / 1.63	65.5 / 1.79	70.5 / 2.22
Large	3	55.5 / 1.60	69.5 / 1.84	70.0 / 2.00	63.0 / 2.18	60.0 / 2.36	67.0 / 3.46
Large	4	48.5 / 1.83	62.5 / 2.11	64.5 / 2.29	59.0 / 2.61	47.0 / 2.95	53.5 / 4.70
Mean feasibility (%)	–	58.3	71.7	72.3	69.6	66.1	71.3

choices. The two variants use the same MRA decomposition, local horizons, target/avoid projection, rollout controller, and 200 test cases per setting.

As Table 3 shows, CDGS improves rollout feasibility in two of six settings, but averaging has a slightly higher mean feasibility (72.3% versus 70.9%) and a much lower mean planning time (1.77 versus 13.48 seconds). CDGS is about 5.9–9.1 times slower across these settings. Thus, neighboring-score averaging is an empirical choice favoring inference efficiency with competitive rollout feasibility here, not a generally exact or superior overlap estimator; it has no general approximation-error bound.

C.3.3 Sensitivity to local horizon allocation

Let H be the fixed trajectory-window horizon used to train the frozen diffusion backbone. At inference time, we test five uniform local horizons, $0.6H$, $0.8H$, $1.0H$, $1.2H$, and $1.4H$, plus a distance-proportional allocation that distributes the total budget NH among segments according to the Euclidean distances between consecutive targets. The Temporal U-Net accepts these different inference lengths without retraining. On Medium and Large with $N = 2, 3, 4$, each allocation uses the same 200 tasks within a setting, for $2 \times 3 \times 6 \times 200 = 7{,}200$ planning and rollout evaluations. Each entry in Table 4 gives rollout feasibility and mean planning time in seconds.

The best uniform choice in each setting falls between $0.8H$ and $1.2H$; $1.0H$ has the highest mean feasibility across the six settings (72.3%). Short allocations can leave too little time to connect milestones, whereas long allocations move farther from the training-window length and may accumulate more execution error. Distance-proportional allocation helps on Medium but is less reliable on Large, where straight-line distance misses obstacle-induced detours; it is also slower in this implementation because unequal segment lengths require separate model forwards rather than one batched forward per DDIM step. Fixed local lengths restrict which feasible milestone-time allocations the sampler can recover, limiting completeness. They do not weaken soundness: any returned discrete trajectory that satisfies every allocated local reach-avoid constraint still satisfies the original untimed MRA task. Adaptive allocation based on obstacles, reachability, or learned transition times is a useful extension, with additional inference or training cost.

Table 5: Result of Prefix-Suffix Tasks on Maze2D.

Dataset	Prefix Segment	Suffix Segment	Success Rate	Planning Time (s)
umaze	1	2	0.855	0.649 ± 0.020
	1	3	0.845	0.697 ± 0.020
	2	2	0.825	0.709 ± 0.023
	2	3	0.765	0.760 ± 0.023
medium	1	2	0.780	1.378 ± 0.237
	1	3	0.740	1.550 ± 0.270
	2	2	0.755	1.619 ± 0.289
	2	3	0.690	1.802 ± 0.479
large	1	2	0.740	1.557 ± 0.356
	1	3	0.565	1.822 ± 0.479
	2	2	0.585	1.833 ± 0.486
	2	3	0.545	2.092 ± 0.464

C.3.4 Optional smoothness guidance

The default sampler prioritizes MRA satisfaction and does not claim to minimize a trajectory-quality objective. As an optional extension, at each denoising step we estimate a clean trajectory, evaluate a differentiable smoothness cost on successive states, and take gradient steps toward lower cost before applying the MRA projection. The hard projection continues to enforce the local target and avoid constraints on a returned discrete plan, while the guidance only biases which feasible plan is sampled. On Maze2D Medium and Large with $N = 2, 3, 4$, this guidance reduces the average measured smoothness cost by 2.68% relative to a matched no-guidance run; average planning time rises from 2.1 to 2.3 seconds because of the extra gradient computation. These are empirical quality and runtime measurements, not a guarantee of a globally optimal or dynamically feasible trajectory.

C.4 Details of Prefix-Suffix Tasks on Maze2D

Experimental Settings. We further evaluate the prefix-suffix extension on the three Maze2D layouts—*U-Maze*, *Medium*, and *Large*—using the same task-agnostic D4RL trajectory priors and the same PD execution controller as in the finite-horizon MRA benchmark. Each planning instance consists of a one-time prefix followed by one suffix cycle, encoded as a finite lasso-shaped trajectory with the loop junction imposed by the periodic boundary condition in Eq. (19). The columns **Prefix Segment** and **Suffix Segment** in Table 5 denote the numbers of local reach-avoid stages assigned to the prefix and suffix parts, respectively. For each dataset and each prefix-suffix setting, we evaluate 200 randomly generated test cases. As in the Maze2D benchmark above, the reported settings use one obstacle in U-Maze and two obstacles in Medium and Large. The target and obstacle regions are again modeled as fixed-radius circular disks, so any target or obstacle violation during projection is corrected analytically by projecting to the nearest point on the corresponding circular boundary.

Evaluation Metrics. We report two quantitative metrics. **Success Rate** denotes the fraction of roll-outs in which the executed trajectory successfully completes the prefix and satisfies the suffix-cycle requirement while respecting the corresponding avoid constraints. In our evaluation, the suffix-cycle requirement is declared satisfied if the rollout can complete two suffix cycles. Here, completing one suffix cycle means that starting from the initial target set of the suffix cycle, the robot reaches the subsequent suffix targets and finally returns to that same initial target set, without colliding with obstacle regions at any time during the cycle. **Planning Time** denotes the mean wall-clock time required to generate the finite prefix-suffix plan by the diffusion model.

Full Results. The complete results under the selected obstacle settings are presented in Table 5.

Analysis. The prefix-suffix results show a consistent trend across all three layouts: success rate decreases and planning time increases as either the prefix length or the suffix length grows. This is expected because larger prefix-suffix decompositions require executing more stage-wise reach-avoid segments and maintaining a feasible loop-consistent rollout over a longer horizon.

Table 6: Detailed results in Cube under one-obstacle and two-obstacle settings.

Segment	1 obstacle		2 obstacles	
	Success Rate	Planning Time	Success Rate	Planning Time
1	0.992	0.868 ± 0.014	0.992	0.915 ± 0.032
2	0.987	0.967 ± 0.032	0.982	1.026 ± 0.074
3	0.976	1.105 ± 0.057	0.950	1.138 ± 0.077
4	0.972	1.373 ± 0.105	0.938	1.382 ± 0.111

U-Maze remains the easiest setting. Across the four tested prefix-suffix configurations, the success rate stays between 76.5% and 85.5%, while diffusion planning time remains below 0.8 seconds. Medium becomes noticeably harder, with success rates between 69.0% and 78.0% and planning times between 1.378 and 1.802 seconds. Large is the most challenging layout, where the success rate ranges from 54.5% to 74.0% and planning time rises to as much as 2.092 seconds.

A longer suffix is particularly costly, especially in the harder layouts. For example, in Large with one prefix segment, increasing the suffix from 2 to 3 segments lowers the success rate from 74.0% to 56.5% while increasing planning time from 1.557 to 1.822 seconds. This suggests that, beyond the difficulty of reaching the loop once, reliably generating and executing a repeatable suffix cycle becomes the main challenge as the environment and horizon both become more demanding.

C.5 Details of Experiments under More Complex Dynamics

To further evaluate the generality and robustness of our framework, we conduct experiments on two dynamics-rich OGBench domains [33]: *cube-single-play* ("Cube") and *antmaze-medium-navigate* ("AntMaze").

Experimental Settings. In both environments, training relies only on the task-agnostic offline trajectories released with the benchmark, while evaluation replaces the original goal-conditioned objectives with ordered multiple reach-avoid tasks. Each evaluation instance contains ordered target sets together with avoid regions, so success requires composing the locally feasible reach-avoid segments into a single executable trajectory.

For Cube, we formulate planning in the end-effector Cartesian space and track the resulting x-y-z trajectory with a PD controller. We evaluate two obstacle settings in Cube: cases with one obstacle and cases with two obstacles, and for each setting we test 500 randomly generated cases. For AntMaze, planning is performed in the two-dimensional x-y workspace, while execution uses an inverse-dynamics controller that maps the current observation and the next x-y target to a low-level action. We evaluate two obstacle settings in AntMaze: cases with two obstacles and cases with three obstacles, and for each setting we test 1000 randomly generated cases. In AntMaze, each target region and each obstacle region is represented as a fixed-radius circle in the x-y workspace; in Cube, the same fixed-radius construction is used in the Cartesian planning space. Thus, whenever a target or obstacle constraint is violated during projection-based conditioning, the corresponding Euclidean projection is available in closed form as the nearest point on the relevant boundary.

Evaluation Metrics. We report two quantitative metrics. **Success Rate** denotes the fraction of trials in which the executed segment reaches its target while satisfying the corresponding avoid constraint. **Planning Time** denotes the wall-clock time required to generate the plan by the diffusion.

Full Results. The complete segment-wise results are presented in Tables 6 and 7.

Analysis. The detailed results reveal a similar aggregate trend in both domains: the reported success rate tends to decrease and planning time tends to increase from Segment 1 to Segment 4. This is mainly because, as the number of segments increases, rollout deviation accumulates, making later segments harder to execute reliably and therefore lowering the success rate.

Cube remains the easier of the two environments. Under the one-obstacle setting, the success rate stays between 97.2% and 99.2% across the four reported segment groups, and under the two-obstacle setting it remains between 93.8% and 99.2%. In both cases, planning time stays below 1.4 seconds. Moving from one obstacle to two obstacles causes only a mild degradation in the aggregated

Table 7: Detailed results in AntMaze under two-obstacle and three-obstacle settings.

Segment	2 obstacles		3 obstacles	
	Success Rate	Planning Time	Success Rate	Planning Time
1	0.951	1.018 ± 0.082	0.929	1.136 ± 0.197
2	0.881	1.366 ± 0.117	0.852	1.621 ± 0.336
3	0.821	2.006 ± 0.111	0.794	2.002 ± 0.503
4	0.741	2.401 ± 0.428	0.741	2.443 ± 0.621

segment-wise results. When planning is expressed in end-effector Cartesian space, the composition-projection mechanism maintains reliable local feasibility with modest computational cost.

AntMaze is substantially more challenging. Under both the two-obstacle and three-obstacle settings, the aggregated success rate decreases with the segment index, and planning time rises from about one second for Segment 1 to around 2.4 seconds for Segment 4. The three-obstacle setting is consistently no better than the two-obstacle setting on the first three reported segment groups, which is expected because denser avoid constraints leave less room for correction under the ant's strongly coupled locomotion dynamics. Notably, both settings reach the same success rate of 74.1% for Segment 4, indicating that in the hardest aggregated regime the difficulty is dominated not only by obstacle count, but also by the general challenge of execution under complex locomotion dynamics.

Overall, these results show that the proposed framework transfers beyond simple planar navigation and remains effective in both manipulation-like and locomotion settings. At the same time, they indicate that the main challenge in these richer domains comes from the combination of two factors: maintaining local reach-avoid satisfaction across multiple stages and faithfully executing low-dimensional plans in systems with more complex underlying dynamics.

C.6 Additional Implementation Details

Diffusion backbone and training. We train one task-agnostic Temporal U-Net diffusion prior per environment using fixed-length state-action windows from the benchmark's offline trajectories: D4RL for the three Maze2D layouts [31] and OGBench for AntMaze and Cube [33]. The training-window horizon for each environment is H in Table 8; at test time, each local MRA segment uses this horizon by default. Training applies Gaussian forward noising with a cosine noise schedule. The network predicts the clean window $\hat{\boldsymbol{\tau}}^0_\theta(\boldsymbol{\tau}^j, j)$ and minimizes the L_2 clean-trajectory loss $\mathbb{E}_{\boldsymbol{\tau}^0,j,\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{\tau}}^0_\theta(\boldsymbol{\tau}^j, j) - \boldsymbol{\tau}^0\|_2^2\big]$. For score evaluation, the clean prediction implies $\hat{\boldsymbol{\epsilon}}_\theta = (\boldsymbol{\tau}^j - \sqrt{\bar{\alpha}_j}\hat{\boldsymbol{\tau}}^0_\theta)/\sqrt{1-\bar{\alpha}_j}$, which can be used in Eq. (5). After this once-per-environment training, the backbone is frozen for every MRA experiment: no MRA specification, target or obstacle region, or test instance is used to train or fine-tune it. Task constraints are imposed only by test-time composition and projection.

DDIM inference. Although Section 2.1 presents the standard DDPM training formulation, all experiments in this paper use DDIM sampling at test time [41]. Compared with ancestral DDPM sampling, DDIM replaces the stochastic Markovian reverse chain with a non-Markovian reverse process that preserves the same denoising training objective. As a result, the same pre-trained diffusion prior can be used directly with DDIM during generation, without any additional training or task-specific fine-tuning. We adopt DDIM because long-horizon MRA planning repeatedly invokes the sampler inside the composition-projection loop, so reducing the number of reverse steps materially improves planning efficiency while retaining good trajectory quality. DDIM also exposes a noise-strength parameter η_{DDIM}: setting $\eta_{\mathrm{DDIM}} = 0$ yields deterministic sampling, while larger values inject additional stochastic noise during the reverse process.

Table 8 summarizes the environment-specific diffusion planning horizon together with the number of DDIM reverse steps and the DDIM noise-strength parameter used in our experiments. Within each environment family, the same sampler settings are shared by the corresponding finite-horizon and prefix-suffix evaluations. For fairness, the guidance-based baselines in the Maze2D benchmark use the same DDIM sampler parameters as our method in the corresponding environment, so the

Table 8: Environment-specific diffusion settings. H is the fixed training-window horizon and the default local planning horizon; J is the number of DDIM reverse steps, and η_{DDIM} is the DDIM noise-strength parameter used at test time.

Environment	H	J	η_{DDIM}
Maze2D-U-Maze	128	60	0.02
Maze2D-Medium	256	100	0.05
Maze2D-Large	384	100	0.05
Cube	128	100	0.05
AntMaze	384	100	0.05

comparison differs in the conditioning mechanism rather than in the underlying sampler configuration.

D Proofs

D.1 Proof of Theorem 1

Proof. Suppose Algorithm 1 returns a stitched trajectory. Its full projection at the final step $j = 1$ succeeded for every segment, giving $\boldsymbol{\tau}_i^0 \in \mathcal{C}_i$ for all i. The same conclusion holds under the implementation schedule described above, because it also applies the full projection at $j = 1$. By the definition of these sets, $\boldsymbol{\tau}_i^0 \models \Phi_i$ at its sampled states for all $i = 1, \ldots, N$, the first segment starts at $\tau_{1,0}^0 = x_0$, and each later segment starts in $\mathcal{T}_{i-1}$.

By the factor-graph overlap update rule described after Eq. (15), the unconditional step updates each overlap state O_i once from the averaged score of the two neighboring segments and writes the result to both adjacent segment boundaries. Therefore, for each $i \in \{1, \ldots, N-1\}$ and reverse step j, there is a single nominal overlap state z_i^{j-1} shared by segments i and $i+1$ at O_i. Once the milestone times are fixed, Eq. (9) implies that each local task $\Phi_i = (m_{i-1}, m_i, \mathcal{T}_i, \mathcal{G}_i)$ requires $\tau_{m_i} \in \mathcal{T}_i$ and $\tau_k \in \mathcal{G}_i$ for all $k \in \{m_{i-1}, m_{i-1}+1, \ldots, m_i\}$. Hence each feasible set $\mathcal{C}_i$ imposes only pointwise set-membership constraints on the states of segment i. Therefore the Euclidean projection in Eq. (17) is temporally separable: it projects each state independently onto its admissible set, without requiring separability among the components of a state. For segment i, the overlap state is projected onto $\mathcal{T}_i \cap \mathcal{G}_i$, while for segment $i+1$ it is projected onto $\mathcal{T}_i \cap \mathcal{G}_{i+1}$. Since $\mathcal{T}_i \subseteq \mathcal{G}_i \cap \mathcal{G}_{i+1}$, both sets reduce to $\mathcal{T}_i$. Because the Euclidean projection onto $\mathcal{T}_i$ is single-valued, both neighboring segments assign the same projected overlap value $P_{\mathcal{T}_i}(z_i^{j-1})$. Under the displayed algorithm, full projection preserves equality of the neighboring overlap states at every step by the argument above. Under the delayed avoid-projection schedule, shared target endpoints are projected consistently during early steps, and full projection preserves their equality from the second half onward. Thus after the final reverse step every pair of adjacent segments has matching overlaps.

The final stitching step in Algorithm 1 therefore only concatenates already-matching neighboring segments and does not modify any state. Consequently, the stitched trajectory inherits the initial condition $\tau_0^0 = x_0$, and each of its local segments is exactly $\boldsymbol{\tau}_i^0$, which already satisfies Φ_i.

Finally, the semantic decomposition of the MRA task in Eq. (11) shows that a global trajectory satisfies Φ if and only if there exists an ordered milestone sequence such that every local segment satisfies its corresponding sub-task Φ_i. The stitched trajectory $\boldsymbol{\tau}^0$ satisfies this condition because every one of its local segments satisfies the corresponding Φ_i. Therefore $\boldsymbol{\tau}^0 \models \Phi$.

Thus, whenever Algorithm 1 returns a trajectory, that discrete trajectory satisfies both $\tau_0^0 = x_0$ and $\boldsymbol{\tau}^0 \models \Phi$. The argument establishes sampled-state set membership and overlap consistency; it does not establish dynamically admissible transitions, collision-free interpolation, or safe tracking. □

Discussion of the assumptions. The assumptions on $\mathcal{T}_i$ are used only to guarantee that, after each independent projection step, the two neighboring segments still project their shared overlap state to the same point. In particular, the condition $\mathcal{T}_i \subseteq \mathcal{G}_i \cap \mathcal{G}_{i+1}$ ensures that both sides project onto the same target set, while the single-valued projection assumption rules out ambiguity in the selected overlap point. These assumptions are therefore technical conditions for the proof strategy based on

exact overlap preservation under separate projections, rather than intrinsic requirements of the compositional diffusion framework itself. In practice, they could be replaced by an alternative stitching or boundary-reconciliation procedure that explicitly matches adjacent segment endpoints after projection, for example by a dedicated stitching step, a joint projection of neighboring boundaries, or another consistent overlap-selection rule.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe the paper's main contributions: compositional planning for long-horizon MRA tasks, the prefix-suffix extension, the soundness guarantee, and the empirical comparisons. See the Abstract, Section 1, and the summary of contributions in the Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated "Limitations" section that discusses the approximation in long-horizon score composition, dependence on local data coverage, and the cost of projection oracles. See the "Limitations" section in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper states the assumption needed for the main theorem and provides a complete proof in the appendix. See Theorem 1 and the "Proofs" section.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The submission provides the task construction, evaluation protocol, and planning settings in the paper, and we will include anonymized code to reproduce the main experimental results. Together, these materials provide a concrete path for replication of the reported results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide an anonymized code release in the supplemental material for reproducing the main experiments. The experiments also use public benchmark assets such as D4RL.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the benchmark environments, task construction, evaluation metrics, and DDIM sampling parameters, and the anonymized code release will provide the remaining implementation and configuration details needed to understand the training and test settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The paper reports mean planning times with "±" values in the result tables and reports success rates over at least 200 randomized test cases for each setting. These statistics provide quantitative information about variability and reliability for the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the compute platform used for the experiments, namely an Intel i9-10940X CPU and an Nvidia 3090 GPU, and it reports wall-clock planning times throughout the main result tables. Together with the implementation details provided in the paper and the anonymized code release, this gives a practical description of the compute resources used in our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses standard offline benchmark datasets and simulation environments for robotic planning, does not involve human subjects, and we are not aware of any aspect of the research that conflicts with the NeurIPS Code of Ethics.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: In our opinion the paper does not have direct social impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk foundation model, an Internet-scraped dataset, or another asset of the type targeted by this question. The experiments are conducted on standard robotic benchmark datasets and simulated environments.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have included all information for the assets we used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The submission will include an anonymized code release for the proposed method.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so IRB approval or an equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method in this paper does not use LLMs as an important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.