VALUE-GUIDED VARIATIONAL LATENT DYNAMICS FOR OFFLINE SKILL STITCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study building agents capable of solving long-horizon tasks in the context of offline reinforcement learning (RL). Existing RL methods effectively learn individual skills. However, seamlessly combining these skills to tackle long-horizon tasks presents a significant challenge, as the termination state of one skill may be unsuitable for initiating the next skill, leading to cumulative distribution shifts. Previous works have studied skill stitching through online RL, while such methods can be time-consuming and may raise safety concerns when learning in the real world. Some other trajectory chaining approaches in previous literature can be utilized as offline skill stitching, but they either rely on time-intensive model-based planning, or neglect the underlying dynamics system implicit in the environment. In this paper, we propose to encode the states separated in time by several steps in a self-supervised manner via a variational auto-encoder. We then align the latent representations with the real actions, modeling the latent dynamics. Given that the aggregated datasets from all skills provide diverse and exploratory data, such latent dynamics can likely be leveraged to generate necessary transitions for stitching skills. During inference, we sample a population of latent actions and make selections according to a score-based state assessment. Extensive experimental results across benchmarks of both navigation and manipulation validate the effectiveness of our approach in comparison to baseline methods under offline settings.

1 INTRODUCTION

In the context of sequential decision making, an essential capability for intelligent agents is to solve long-horizon tasks composed of multiple stages. A composite long-horizon task can be viewed as an ordered combination of several "skills". Consider a scenario where an intelligent robot has independently learned two skills: walking from the initial state to g_0 (skill-1) and walking from g_2 to g_1 (skill-2), as shown in Figure 1. To complete the task of "starting from initial state, go to g_0, then go to g_1", directly executing these skills in sequence could lead to significant failure, as the agent gets stuck in an unseen state for skill-2 after successfully accomplishing skill-1. The termination states of the previous skills may be unsuitable to execute the next skills, especially when there are large discrepancies in the data used to train the different skills. Therefore, it is crucial to introduce a strategy for stitching adjacent skills together to deal with long-horizon tasks.

Reinforcement learning (RL) has emerged as a dominant paradigm for solving sequential decision making tasks, demonstrating remarkable capabilities in learning diverse real-world skills (Margolis & Agrawal, 2023; Yuan et al., 2023; Hafner et al., 2023; Yuan et al., 2024; Huang et al., 2025; Yuan et al., 2025a). While RL excels in simpler tasks with relatively fixed goals, managing long-horizon tasks that require multiple sequential skills remains a challenge that has not been fully addressed. Long-horizon tasks involve more transitions and require higher-level multi-step reasoning, making it difficult for RL to learn efficiently with a single policy. Additionally, if we learn each skill individually using RL, the gap between the transitions observed for different skills can be significantly large, complicating the direct chaining of these skills. To address these challenges, we propose to learn skill stitching to facilitate the integration of RL skills.

Previous studies have explored learning and stitching skills using online RL (Konidaris & Barto, 2009; Bagaria & Konidaris, 2019; Gu et al., 2023), which can be time-consuming and sample-inefficient. Other works (Lee et al., 2019; Kang & Oh, 2022; Chen et al., 2023; Lee et al., 2021) have

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

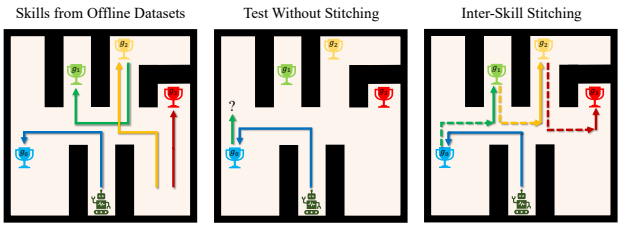


Figure 1: An illustration of skill-level stitching. *(left)* The offline dataset distribution and visualization. Suppose a task requiring executing the skill sequence $g_0 \to g_1 \to g_2 \to g_3$ from an initial position. *(middle)* Without skill stitching, the agent fails early because the first skill's terminal state is unseen during the training of the next skill, whereas *(right)* our stitching approach successfully completes the full trajectory, visualized with solid lines for skills and dashed lines for stitching processes.

managed to pre-train skills from offline datasets but still rely on online RL for stitching. Such online learning stages introduce inefficiencies and safety concerns, especially in real-world settings. The aforementioned challenges and concerns naturally associate with a new question: *Can we learn both skills and stitching policies from offline data?*

In the domain of fully offline learning, there has been previous works (Yuan et al., 2025b; Yang & Wang, 2025) on trajectory stitching, while directly applying such methods to skill-level stitching may raise new issues. Some methods (Li et al., 2024; Lee & Choi, 2025; Yu et al., 2025; Qing et al., 2025) formulate the problem as data augmentation, aiming at generating diverse and exploratory state-action data on the basis of original trajectories. However, such approaches require learning a (diffusion) planner or performing standard offline RL on the augmented composite trajectories to get an ultimate policy. This may encounter problems of high computational complexity, since in the context of skill stitching, there can exist various combinations among skills. Besides, some other previous offline stitching methods (Agia et al., 2023; Mishra et al., 2023) delve into applying an **explicit** chaining phase between two skills. However, such methods either require dense-reward annotated state-action pairs of stitching phase to be given in the datasets, or adopt time-consuming model-based planning, which is highly time-intensive and suffers from the problem of accumulated error in long-horizon step-wise planning.

In this paper, we introduce a fully offline method for skill stitching to tackle long-horizon tasks. We assume access to various skills and their associated datasets, recognizing that high-quality datasets for long-horizon tasks can be challenging to obtain. A proper combination of these skills can effectively accomplish a range of long-horizon tasks. Our primary motivation is that the aggregated offline datasets of all skills can offer rich coverage of environment transitions and dynamics, which can be leveraged to learn skill stitching. Specifically, we first train each individual skill using offline RL algorithms, and then propose to encode the current and future states, which are separated in time by several steps, in a self-supervised manner via a variational auto-encoder. Given that after such a self-supervised learning, the latent representations should imply the transition and dynamics in between, we then align the latent space with the real actions to be executed in the environment, turning it into a latent dynamics modeling. We utilize the properties that the implicit dynamics extracted from the aggregated datasets, capturing environment transitions independent of specific skills, can demonstrate promising generalization for predicting transitions in states not observed by a specific skill. During inference, we sample a population of latent actions and make selections according to the estimated scores of the predicted future states in the stitching phase.

In our experiments, we evaluate our method across benchmarks including both navigation and multi-object manipulation tasks. The results demonstrate that our method effectively manages diverse combinations of skills in each environment, outperforming various baselines in fully offline settings.

To summarize, our main contributions are as follows:

- We introduce the problem of offline skill stitching, tackling the challenges of solving long-horizon tasks with **skill-based** reinforcement learning.
- We propose to model latent transitions between states on aggregated skill datasets in a manner of variational auto-encoding, and align the latent space with multi-step actions to form a latent dynamics system.

- We demonstrate the effectiveness of stitching skills through time-efficient latent sampling under score-based value guidance, which provides an effective framework for fully offline skill stitching and highlights the promising potential of latent dynamics modeling in the realm of long-horizon task execution.

2 Related Work

Most prior works on skill stitching adopt online learning, where a policy is trained through environment interaction (Lee et al., 2019; 2021; Kang & Oh, 2022; Chen et al., 2023). While some literature models trajectory stitching within a single task, few works address fully offline skill-level stitching, in which all skill-like policies and the stitching phase must be derived from limited offline data.

Planning-based offline stitching methods. Some methods (Agia et al., 2023; Mishra et al., 2023; Yu et al., 2025) investigate **skill-level** stitching via planning. GSC (Mishra et al., 2023) generates feasible skill parameter sequences as planning solutions, but requires stitching transitions to appear in the offline dataset and trains skills that can be directly chained. In contrast, our method learns each skill from skill-only datasets that contain no stitching states or actions. STAP (Agia et al., 2023) uses model-based planning with the Cross-Entropy Method (CEM) (Rubinstein, 1999; De Boer et al., 2005) to stitch adjacent skills. Such methods are time-intensive and suffer from accumulated error in long-horizon step-wise planning. By comparison, we model latent dynamics, requiring only one-time sampling and selection without multi-turn optimization, saving considerable time over multi-step genetic planning.

Generative trajectory stitching with data augmentation. Some works use generative models in a data-augmentation regime to expand state coverage or diversity. DiffStitch (Li et al., 2024) generates intermediate states with a diffusion-based trajectory generator given current and goal states, then produces actions via a learned inverse dynamics model (IDM) and performs standard offline RL on the augmented data. SCoTS (Lee & Choi, 2025) and ASTRO (Yu et al., 2025) select temporally close trajectories, generate intermediate transitions with a trained diffusion stitcher, and learn diffusion planners or offline-RL policies on the stitched data. BiTrajDiff (Qing et al., 2025) uses forward and backward trajectory diffusion models to generate bidirectional state trajectories and then performs offline RL after data filtering. However, these methods ultimately learn a policy over the augmented dataset, essentially treating the whole task in a top-down fashion. For skill stitching, they would need to learn many policies for different skill combinations. Instead, we combine individual skills bottom-up to accomplish long-horizon tasks with various combinations, requiring only a single shared stitching phase.

Goal-conditioned approaches and prior-based methods over latent skill representation. Goal-conditioned offline RL algorithms (Park et al., 2023; Zeng et al., 2023) have been widely applied to goal-conditioned decision making, yet few works focus on goal-conditioned trajectory stitching that treats the next trajectory's states as goals. This requires the goal state (e.g., the initial state of the next trajectory) to be known, which is problematic in skill-level stitching where future states are typically unknown at test time. Our method, facing unknown states for the next skill, samples and selects proper (latent) actions guided by a score module that steers toward the next skill. DIDI (Liu et al., 2024) maps task-agnostic skills to a latent space and uses a contextual policy conditioned on the latent variable to encourage diversity, showing that skills switch when the latent variable changes. However, at inference no latent variables are given, and sampling latents can be unstable, making consistent policy behavior difficult. Such methods only train and execute individual skills, while our method inserts an explicit stitching phase between adjacent skills.

3 Preliminaries

Consider n different tasks in a certain environment. Following prior work on offline reinforcement learning (RL), we assume access to datasets for training all the skills $\{\mathcal{D}_i\}_{i=1}^n$, where each $\mathcal{D}_i$ contains the trajectories of accomplishing a task i. The task is represented as a Markov decision process (MDP) $\mathcal{M}$ defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \rho_0 \rangle$, where $\mathcal{S}$, $\mathcal{A}$, γ, and ρ_0 represent the state space, action space, discount factor, and the distribution of initial states, respectively. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the transition function, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. We denote the skill set as $\{\pi_i\}_{i=1}^n$, i.e., the policies to accomplish each single task i. Formally, the objective is to learn the joint

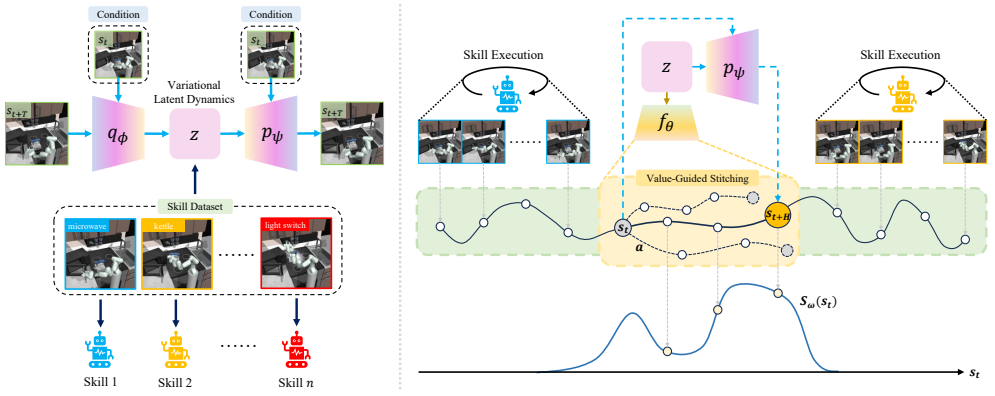


Figure 2: An overview of our method. **Offline training phase (left):** we learn each individual policy (skill) from the offline datasets, which are used to compose long-horizon tasks. Our variational latent dynamics model is trained from the aggregated dataset, which is used to perform value-guided planning during stitching. **Inference phase (right):** we insert a stitching process between the executions of two adjacent skills to enhance long-horizon task execution, leveraging our latent dynamics model and translator (Section 4.1). Our stitching method employs value-guided planning, where the optimization objective is steered by our evaluator (Section 4.2), leading the agent to new states that are suitable for executing the subsequent skill.

policy Π so as to maximize the expected return in long-horizon task combinations:

$$R = \mathbb{E}_{a_t \sim \Pi(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t,a_t)} \left[\sum_t \left(\gamma^t r(s_t, a_t) \right) \right], \tag{1}$$

where the joint policy Π is composed of both the skills $\{\pi_i\}_{i=1}^n$ and the stitching processes between skills.

Our goal is to leverage both the offline datasets $\{\mathcal{D}_i\}_{i=1}^n$ and the skill set $\{\pi_i\}_{i=1}^n$ to tackle any long-horizon task combination that necessitates sequential execution of (a subset of) skills.

Thus, we aim to design an algorithm that can leverage offline datasets for stitching a sequence of skills $(\pi_{i_1}, ..., \pi_{i_m})$ into a joint policy $\Pi = (\pi_{i_1}, \pi^{\text{stitch}}_{i_1,i_2}, \pi_{i_2}, ..., \pi^{\text{stitch}}_{i_{m-1},i_m}, \pi_{i_m})$, so as to tackle the long-horizon task. Here, each π_{i_k} is a skill required to solve the task that is learned from the corresponding skill dataset; each $\pi^{\text{stitch}}_{i_k,i_{k+1}}$ denotes the stitching process to chain the adjacent skills i_k and i_{k+1}; $(i_1, i_2, ..., i_m)$ is chosen from $(1, 2, ..., n)$ $(m \leq n)$, which means we aim at accomplishing skill combinations in arbitrary order given the trained policy for each single skill.

4 METHODS

In this section, we introduce our methods (Figure 2), including the learning of variational latent representations and the alignment with actions to form a latent dynamics (Section 4.1), our evaluator which combines the value function of the skill and the distance to positive and negative states (Section 4.2), as well as how we apply such stitching methods during inference time (Section 4.3).

Next, we base our methods on a set of notations, following our problem formulation in Section 3. We use $\{\mathcal{D}_i\}_{i=1}^n$ to denote the datasets corresponding to n skills, with each $\mathcal{D}_i$ containing a set of trajectories $\{\tau_j\}_{j=1}^{N_i}$. Each trajectory has a shared structure: $(s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T)$.

4.1 LEARNING VARIATIONAL LATENT DYNAMICS

To model the transition from a previous state s_t to a future state s_{t+H}, we learn a representation of such transitions in a fully unsupervised manner. Inspired by LAPA (Ye et al., 2025), in which a VQ-VAE-style (Van Den Oord et al., 2017) architecture is used for latent action pre-training, we adopt a temporal-chunked conditional VAE (CVAE) (Sohn et al., 2015) to fit the transitions between two states that appear successively. During encoding, our temporal-chunked CVAE takes current

state s_t and goal state s_{t+H} as input, and maps the concatenation of the two states to a latent variable z in the latent space:

$$q_\phi(z_t|s_t, s_{t+H}) = \mathcal{N}\left(z_t; \mu_\phi(s_t, s_{t+H}), \sigma^2_\phi(s_t, s_{t+H})\mathbf{I}\right). \tag{2}$$

In terms of decoding phase, the CVAE decoder takes the concatenation of s_t and z, and reconstructs the goal state s_{t+H}:

$$p_\psi(s_{t+H}|s_t, z_t) = \mathcal{N}\left(s_{t+H}; \mu_\psi(s_t, z_t), \sigma^2\mathbf{I}\right). \tag{3}$$

In this process, the goal state s_{t+H} can be viewed as the actual data to be inferred and generated, while the current state s_t can be viewed as the condition. Specifically, we choose to learn the changes that happen within several steps ($H > 1$), based on the general insight that learning action chunks can be beneficial to maintaining the continuity and stability of the behaviors (Zhao et al., 2023), greatly alleviating the accumulated error appeared in step-wise planning. Our temporal-chunked CVAE is optimized using the following ELBO (Kingma & Welling, 2013) objective:

$$\mathcal{L}_{\text{VAE}}(\phi, \psi) = -\mathbb{E}_{z_t \sim q_\phi(z_t|s_t, s_{t+H})}[\log p_\psi(s_{t+H}|s_t, z_t)] + D_{\text{KL}}\left(q_\phi(z_t|s_t, s_{t+H}) \parallel p(z_t)\right), \tag{4}$$

which can be rewritten as

$$\begin{aligned}\mathcal{L}_{\text{VAE}}(\phi, \psi) = \mathbb{E}_{(s_t, s_{t+H})\sim\mathcal{D}}\left[\mathbb{E}_{z_t\sim q_\phi, \hat{s}_{t+H}\sim p_\psi}\left[\|s_{t+H} - \hat{s}_{t+H}\|^2\right]\right] \\ + \lambda_{\text{KL}} D_{\text{KL}}\left(q_\phi(z_t|s_t, s_{t+H}) \parallel \mathcal{N}(0, \mathbf{I})\right).\end{aligned} \tag{5}$$

Such a encoder-decoder architecture provides a latent representation of state-level transitions in terms of temporal changes. Next, we align this latent space with the actual actions using a translator f, parameterized by θ: $\hat{A}_t = f_\theta(s_t, z_t)$, where z_t is encoded from $q_\phi(\cdot|s_t, s_{t+H})$, and $A_t = \mathbf{a}_{t:t+H-1} = (a_t, a_{t+1}, \ldots, a_{t+H-1})$ is an action chunk with size H. Our translator is optimized via the MSE loss:

$$\mathcal{L}_{\text{trans}}(\theta) = \mathbb{E}_{(s_t, A_t, s_{t+H})\sim\mathcal{D}}\left[\|A_t - \hat{A}_t\|^2\right]. \tag{6}$$

Our temporal-chunked CVAE, together with the corresponding translator, forms our variational latent dynamics, since these two modules jointly model the multi-step dynamics system of the environment.

4.2 Score-Based Evaluator

During the stitching phase between two adjacent skills, we adopt model-based planning with our trained variational latent dynamics. To judge whether the next state during planning is suitable enough for next skill execution and when to stop the stitching process, we need a score-based evaluator to take this measurement. Previous work uses the Q-function of the next skill as guidance (Agia et al., 2023), while only taking value functions of the skills into consideration may be inaccurate, causing problems of insufficient over-estimation and raising accuracy concerns.

In our method, we propose to combine the values judged by the skills (policy critics) with state-level distances: we aim at minimizing the distance between the planned new state and those states belonging to the next particular skill in the offline dataset, while maximizing the distance between the new state and states belonging to the other skills in the meanwhile. Since naïve Euclidean L2 distance metric is usually biased in a high-dimensional state space, following some data-augmentation-based methods (Lee & Choi, 2025; Yu et al., 2025), we adopt a temporal-distance-preserving (TDP) distance measurement in latent space by calculating

$$\text{dis}(s_1, s_2) = \|\sigma(s_1) - \sigma(s_2)\|_2, \tag{7}$$

in which $\sigma(\cdot)$ is optimized in a temporal difference objective similar with implicit Q-Learning Kostrikov et al. (2022):

$$\mathcal{L}(\sigma) = \mathbb{E}_{s_t, s_{t+1}, s_{t+H}\sim\mathcal{D}}\left[L^2_\tau\left(-\mathbf{1}(s_t \neq s_{t+H}) - \gamma\|\bar{\sigma}(s_{t+1}) - \bar{\sigma}(s_{t+H})\|_2 + \|\sigma(s_t) - \sigma(s_{t+H})\|_2\right)\right], \tag{8}$$

where $\mathcal{D} = \bigcup_{i=1}^n \mathcal{D}_i$, $\bar{\sigma}(\cdot)$ is a target network, γ is a discount factor, and L^2_τ denotes the τ-expectile.

In practice, we relabel each state in the skill-i dataset with an assigned score as:

$$S^{\text{eval}}(s, i) = V^{\pi_i}(s) - \mathbb{E}_{s'\sim\mathcal{D}_i}\left[\text{dis}(s, s')\right] + \mathbb{E}_{s'\sim\mathcal{D}_j, j\neq i}\left[\text{dis}(s, s')\right], \tag{9}$$

where V^{π_i} denotes the value function learned in the policy of skill-i. Finally, we train a multi-skill evaluator (parameterized by ω) in a supervised manner instead of calculating the actual score at test time according to Equation (9), using the following objective:

$$\mathcal{L}_{\text{eval}}(\omega) = \mathbb{E}_{s,i\sim\mathcal{D}}\left[\|S_\omega(s,i) - S^{\text{eval}}(s,i)\|^2\right], \tag{10}$$

where $\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$.

4.3 Offline Skill Stitching during Inference

During inference, we execute each skill individually with their own policies trained from the skill datasets, and add stitching phases between two adjacent skills. With our temporal-chunked variational latent dynamics as well as the evaluator, we can conduct action planning as the stitching process in between.

The stitching phase starts when the last skill receives a success signal. Within a loop, we sample a population of latent variables z in the latent space of the CVAE module, and decodes z together with current state s_t into predicted future state s_{t+H}. Then we evaluate those future states using our evaluator to select the best latent variable z in the population, and translates $\text{concat}(s_t, z)$ into the actual action chunk A_t to be executed in the environment. After executing each action, we check the score of the actual state (since the actual state after executing actions may not be equal to the predicted future state by CVAE decoder), and such a planning phase loops until the actual score of current step becomes no higher than the last step, serving as the stitching policy $\pi^{\text{stitch}}_{i_k,i_{k+1}}$ in Section 3. See Algorithm 2 in Appendix A for the pseudo code of our inference phase.

5 Experiments

To make a thorough evaluation of our method, we provide experiments on benchmarks with both navigation and manipulation tasks. Comparing with various baselines, the results show that our method is not only effective in improving success rate of skill-level stitching, but also efficient in terms of time cost, which builds a potential for the application in real-world scenarios, e.g., embodied robots.

5.1 Experimental Setup

5.1.1 Environments and Tasks

We evaluate our methods on the PointMaze environment of OGBench (Park et al., 2025), as well as the Kitchen environment in D4RL (Fu et al., 2020).

OGBenchMaze. We slightly modify this environment to fit our skill-level settings. For three PointMaze scenarios with different sizes `medium-stitch-v0`, `large-stitch-v0`, and `giant-stitch-v0`, we fix three goal locations in each scenario, and only keep the trajectories in the original dataset that can reach these three locations, to form our offline dataset. For each size of maze, the three locations (goals) naturally become the three skills, and can be stitched into long-horizon tasks in order. Different from previous researches on offline goal-conditioned RL (Kostrikov et al., 2022; Park et al., 2023; Wang et al., 2023), we formulate the multiple goal locations as a multi-task setting, thereby the goal positions (coordinates) are removed in the observation space, replaced by task indices instead.

Kitchen. A Franka robot arm with gripper works in a kitchen-like environment to accomplish simulated tasks in the kitchen (Fu et al., 2020). This benchmark provides 7 tasks (skills) in total, among which 4 skills appear in the offline dataset, including `bottom burner`, `light switch`, `microwave`, and `kettle`. In this benchmark, the long-horizon tasks are defined as the combinations of multiple skills, e.g. `microwave` $\rightarrow$ `kettle` (open the microwave first and move the kettle afterwards). Since our offline-trained skills (trained with standard implicit Q-learning (IQL) (Kostrikov et al., 2022)) can only accomplish tasks `bottom burner`, `microwave`, and `kettle` with a non-zero success rate, we only evaluate our methods on long-horizon task combinations starting from these three skills.

Table 1: Experimental results on OGBenchMaze environment. For each long-horizon task composed of three skills, the maximum value of AR is 3.00. We test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation of AR, CSR-2, and CSR-3 (denoted as mean±std). The last three rows show the average value of AR, CSR-2, and CSR-3 of each method. The highest values are represented in bold.

Type	Metric	No-Stitching	Vanilla CEM	STAP	DIDI	DiffStitch	SCoTS	ASTRO	BiTrajDiff	VG-VLD (Ours)
medium	Average Return (AR)	0.93±0.03	1.75±0.05	0.93±0.03	1.80±0.13	0.94±0.03	1.54±0.09	0.93±0.02	0.92±0.02	**2.77±0.08**
	Conditional Success Rate (CSR) - 2	0.000±0.000	0.878±0.026	0.000±0.000	0.752±0.046	0.000±0.000	0.440±0.029	0.000±0.000	0.000±0.000	**0.991±0.001**
	Conditional Success Rate (CSR) - 3	NaN	0.004±0.007	NaN	0.246±0.038	NaN	0.495±0.060	NaN	NaN	**0.953±0.000**
large	Average Return (AR)	1.75±0.11	1.75±0.11	1.75±0.11	2.01±0.12	1.79±0.03	1.90±0.13	1.75±0.09	1.78±0.07	**2.63±0.17**
	Conditional Success Rate (CSR) - 2	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	0.977±0.010	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**
	Conditional Success Rate (CSR) - 3	0.000±0.000	0.000±0.000	0.000±0.000	0.327±0.028	0.000±0.000	0.162±0.037	0.000±0.000	0.000±0.000	**1.000±0.008**
giant	Average Return (AR)	0.87±0.05	2.61±0.14	2.06±0.13	0.88±0.05	0.86±0.02	1.14±0.04	1.00±0.04	0.87±0.04	**2.64±0.15**
	Conditional Success Rate (CSR) - 2	0.000±0.000	0.996±0.006	0.679±0.008	0.004±0.007	0.000±0.000	0.154±0.036	0.073±0.008	0.000±0.000	**1.000±0.000**
	Conditional Success Rate (CSR) - 3	NaN	**1.000±0.000**	**1.000±0.000**	0.333±0.577	NaN	**1.000±0.000**	**1.000±0.000**	NaN	**1.000±0.000**
Average	Average Return (AR)	1.18	2.04	1.58	1.56	1.20	1.53	1.23	1.19	**2.68**
	Conditional Success Rate (CSR) - 2	0.333	0.958	0.560	0.578	0.333	0.531	0.358	0.333	**0.997**
	Conditional Success Rate (CSR) - 3	0.000	0.335	0.500	0.302	0.000	0.552	0.500	0.000	**0.984**

5.1.2 BASELINES

We compare our method, **V**alue-**G**uided **V**ariational **L**atent **D**ynamics (VG-VLD), with the following baselines:

The most naïve baseline is **No-Stitching**, where the next skill is executed directly after the previous one. Planning-based baselines includes **Vanilla CEM** and **STAP** (Agia et al., 2023), both performing model-based planning on the basis of Cross-Entropy Method (CEM) (Rubinstein, 1999; De Boer et al., 2005). **DIDI** (Liu et al., 2024) stands for the prior-based methods, which trains policies conditioned on latent skill abstractions to generate future action chunks, and learns task-agnostic priors to help encourage diversity during trajectory generation. Finally comes the methods based on data augmentation: **DiffStitch** (Li et al., 2024) leverages a diffusion-based generator to inpaint the intermediate states given a current state s_t and a future state s_{t+H}, with an inverse dynamics model (IDM) to predict the actions in between. **SCoTS** (Lee & Choi, 2025) trains a diffusion stitcher with the original offline dataset, and generates augmented data to learn a diffusion planner. Built upon SCoTS, **ASTRO** (Yu et al., 2025) adds a dynamics diffuser and weighted stitch augmentation. **BiTrajDiff** (Qing et al., 2025) learns a bidirectional diffusion to generate the stitching process from both forward and backward views.

5.1.3 METRICS

In this paper, we use the following two evaluation metrics to measure the performances of each method:

- **Average Return (AR).** During test time, a long-horizon task is composed of multiple skills in order. The success of each skill corresponds to a sparse reward of 1 (otherwise 0), so an episode return is equal to the number of skills accomplished in this test episode. We test each task on 3 seeds, each for 100 times, and calculate the mean (AR) and standard deviation.
- **Conditional Success Rate (CSR) - N.** We provide another metric, defined as $\mathrm{CSR}{-}N = \frac{\#\text{episodes whose return} \geq N}{\#\text{episodes whose return} \geq N-1}$. The number of episodes that accomplish the N-th skill, over the number of episodes that accomplish the $(N-1)$-th skill, represents the success rate of the N-th skill conditioned on the fact that the $(N-1)$-th skill is already accomplished. Such an evaluation metric provides a straightforward explanation on how effective a stitching method is, especially for those cases where the success rate of the offline policy of the previous skill is inherently low, resulting in a relatively low AR.

5.2 MAIN RESULTS AND ANALYSES

In this section, we provide the main testing results on OGBenchMaze and Kitchen, comparing our method with baselines. The results on OGBenchMaze are listed in Table 1. We choose three skills for each size of maze environment, which means the maximum value of Average Return (AR) is 3.00. For each size of maze environment, we test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation. For those cases with $\mathrm{CSR}{-}2 = 0$, since the corresponding method never accomplishes the 2-nd skill and is impossible to reach the 3-rd

Table 2: Experimental results on Kitchen environment. For each long-horizon task composed of two skills, the maximum value of AR is 2.00. We test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation of AR and CSR-2 (denoted as mean±std). The last two rows show the average value of AR and CSR-2 of each method. The highest values are represented in bold.

Task	Metric	No-Stitching	Vanilla CEM	STAP	DIDI	DiffStitch	SCoTS	ASTRO	BiTrajDiff	VG-VLD (ours)
bottom burner → light switch	Average Return (AR)	1.16 ± 0.02	1.09 ± 0.03	1.00 ± 0.02	1.09 ± 0.06	1.09 ± 0.07	0.99 ± 0.02	1.09 ± 0.01	1.28 ± 0.02	**1.57 ± 0.02**
	Conditional Success Rate (CSR) - 2	0.234 ± 0.014	0.140 ± 0.035	0.049 ± 0.014	0.167 ± 0.042	0.155 ± 0.038	0.061 ± 0.026	0.160 ± 0.020	0.344 ± 0.026	**0.627 ± 0.037**
bottom burner → microwave	Average Return (AR)	0.93 ± 0.02	1.36 ± 0.03	1.33 ± 0.05	1.66 ± 0.02	1.52 ± 0.07	1.44 ± 0.03	1.57 ± 0.04	1.58 ± 0.03	**1.74 ± 0.03**
	Conditional Success Rate (CSR) - 2	0.007 ± 0.006	0.479 ± 0.038	0.401 ± 0.065	0.766 ± 0.035	0.589 ± 0.06	0.538 ± 0.072	0.658 ± 0.017	0.699 ± 0.046	**0.801 ± 0.014**
bottom burner → kettle	Average Return (AR)	0.92 ± 0.01	1.23 ± 0.05	1.23 ± 0.01	1.22 ± 0.05	1.22 ± 0.01	1.19 ± 0.02	1.19 ± 0.01	1.00 ± 0.01	**1.36 ± 0.03**
	Conditional Success Rate (CSR) - 2	0.000 ± 0.000	0.324 ± 0.073	0.285 ± 0.024	0.276 ± 0.038	0.259 ± 0.022	0.266 ± 0.010	0.263 ± 0.020	0.052 ± 0.001	**0.372 ± 0.046**
microwave → bottom burner	Average Return (AR)	1.47 ± 0.01	1.42 ± 0.03	1.42 ± 0.06	1.33 ± 0.06	1.40 ± 0.13	1.41 ± 0.03	1.54 ± 0.05	1.46 ± 0.07	**1.77 ± 0.02**
	Conditional Success Rate (CSR) - 2	0.747 ± 0.013	0.559 ± 0.050	0.600 ± 0.068	0.485 ± 0.018	0.553 ± 0.045	0.565 ± 0.015	0.693 ± 0.019	0.661 ± 0.018	**0.894 ± 0.022**
microwave → light switch	Average Return (AR)	1.42 ± 0.04	1.06 ± 0.07	1.03 ± 0.01	1.11 ± 0.03	1.09 ± 0.08	1.07 ± 0.05	1.20 ± 0.04	0.99 ± 0.02	**1.61 ± 0.05**
	Conditional Success Rate (CSR) - 2	0.639 ± 0.023	0.127 ± 0.063	0.132 ± 0.012	0.215 ± 0.029	0.193 ± 0.042	0.180 ± 0.023	0.275 ± 0.019	0.072 ± 0.028	**0.790 ± 0.040**
microwave → kettle	Average Return (AR)	1.39 ± 0.04	1.17 ± 0.08	1.25 ± 0.02	1.33 ± 0.04	1.34 ± 0.01	1.32 ± 0.06	1.40 ± 0.05	1.40 ± 0.06	**1.69 ± 0.01**
	Conditional Success Rate (CSR) - 2	0.598 ± 0.016	0.294 ± 0.067	0.422 ± 0.013	0.472 ± 0.006	0.441 ± 0.014	0.445 ± 0.040	0.484 ± 0.045	0.541 ± 0.009	**0.767 ± 0.004**
kettle → bottom burner	Average Return (AR)	0.38 ± 0.02	0.43 ± 0.04	0.43 ± 0.03	0.55 ± 0.04	0.55 ± 0.03	0.48 ± 0.04	0.51 ± 0.06	0.49 ± 0.09	**0.82 ± 0.06**
	Conditional Success Rate (CSR) - 2	0.031 ± 0.009	0.033 ± 0.016	0.074 ± 0.043	0.168 ± 0.05	0.172 ± 0.011	0.140 ± 0.065	0.247 ± 0.087	0.119 ± 0.060	**0.615 ± 0.131**
kettle → light switch	Average Return (AR)	0.35 ± 0.01	0.44 ± 0.02	0.48 ± 0.01	0.49 ± 0.06	0.49 ± 0.05	0.44 ± 0.02	0.48 ± 0.03	0.44 ± 0.05	**0.65 ± 0.04**
	Conditional Success Rate (CSR) - 2	0.094 ± 0.059	0.047 ± 0.022	0.151 ± 0.014	0.120 ± 0.038	0.123 ± 0.014	0.082 ± 0.012	0.108 ± 0.047	0.031 ± 0.034	**0.443 ± 0.063**
kettle → microwave	Average Return (AR)	0.26 ± 0.02	0.45 ± 0.03	0.46 ± 0.03	0.46 ± 0.05	0.47 ± 0.06	0.46 ± 0.05	0.47 ± 0.01	0.47 ± 0.04	**0.59 ± 0.04**
	Conditional Success Rate (CSR) - 2	0.000 ± 0.000	0.109 ± 0.085	0.079 ± 0.019	0.150 ± 0.013	0.115 ± 0.063	0.103 ± 0.035	0.188 ± 0.070	0.061 ± 0.017	**0.320 ± 0.022**
Average	Average Return (AR)	0.92	0.96	0.96	1.03	1.02	0.98	1.05	1.01	**1.31**
	Conditional Success Rate (CSR) - 2	0.261	0.235	0.244	0.313	0.289	0.264	0.342	0.287	**0.625**

skill, we denote their CSR-3 as "NaN". While some prior-based methods (DIDI) can be advantageous in terms of `medium-stitch-v0` and `large-stitch-v0`, other planning-based methods (Vanilla CEM and STAP) are talented in `giant-stitch-v0`. Data-augmentation-based methods (SCoTS, ASTRO, and BiTrajDiff) tend to perform worse than planning-based methods, especially in `medium-stitch-v0` and `giant-stitch-v0`. By contrast, our method significantly outperforms all the baselines, with a nearly 100% rate of successfully stitching.

For the Kitchen environment, we test the methods on skill combinations starting from `bottom burner`, `microwave`, or `kettle`, since the IQL policies only have non-zero success rate on these three skills (`bottom burner` and `microwave` $\sim$90%; `kettle` $\sim$30%) when they serve as the first skill. Besides, the skill `light switch` can also be accomplished when it serves as the skill afterwards. We list the results in Table 2. For each long-horizon task composed of two skills, we test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation. For some of the skill combinations (e.g., `bottom burner` $\rightarrow$ `light switch` and `microwave` $\rightarrow$ `light switch`), prior-based, planning-based, and augmentation-based baselines all fall short in comparison to No-Stitching, which means adding unsuitable stitching phases may even bring disadvantages and negative effects using these methods. VG-VLD comprehensively outperforms all the baselines in these manipulation tasks which are considered more complicated, demonstrating the effectiveness of our method. We provide additional results of long-horizon tasks composed of three or four skills in Appendix E.

Apart from the high success rate in accomplishing long-horizon tasks composed of skills, our method also demonstrates efficiency in terms of time cost. We calculate the time consumption of each method during a single stitch, and plot the relationship between CSR-2 and time cost in the Kitchen environment, as shown in Figure 3 (all the rollouts conducted on a single RTX 4090). A gray dashed line is drawn to indicate that from the top-left to the bottom-right corner of the figure, the methods gradually drop in performance with an increasing time cost. Planning-based methods (Vanilla CEM, STAP) suffer heavy time cost due to the time-consuming cross-entropy method, while prior-based and generative methods (DIDI, DiffStitch) are time-efficient but relatively weak in stitching performance. Data-augmentation methods (SCoTS, ASTRO, BiTrajDiff) are excluded because they lack an explicit stitching phase. Our VG-VLD, located at the top-left corner, achieves the best overall performance with relatively low time consumption, demonstrating its time efficiency.

5.3 Ablation Study

5.3.1 Ablations on Chunk Size

We aim to find a suitable chunk size H (which is a hyperparameter actually) for our variational latent dynamics, and conduct experiments with a wide range of chunk sizes. We test our methods using different chunk sizes on the Kitchen environment, and plot the relationship between Conditional Success Rate (CSR) - 2 and chunk sizes, as shown in Figure 4. The overall trend is that with chunk

size increases, the CSR-2 metric rises first and then falls, reaching its maximum value at a size of 16 (which is also the one used in our main experiments). An explanation is that when chunk size is relatively small, the latent dynamics model suffers from accumulated error during the stitching phase; when chunk size is relatively large, both our CVAE module and our translator face the curse of dimensionality, and can only be learned to a sub-optimal performance.

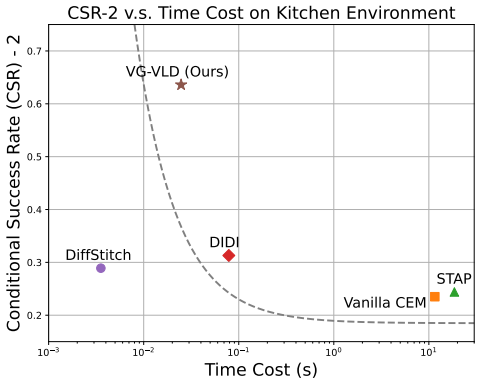


Figure 3: The relationship between CSR-2 and time cost (seconds) in the Kitchen environment. We calculate the time consumption of each method during a single stitch for 10 times and calculate an average to represent the time cost of this particular method.

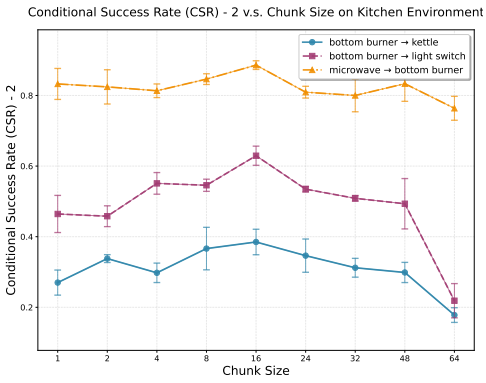


Figure 4: Ablation studies on chunk size. We test our methods using different chunk sizes, and plot the relationship between Conditional Success Rate (CSR) - 2 and chunk size.

5.3.2 Ablations on Score-Based Value Guidance

During inference, we need an evaluator to guide the stitching phase at test time. In our work, we adopt an evaluator that fits a weighted summation of **values** judged by skill critics (which represent value functions) and **distances** from other states coming from same and different skills datasets. In this ablation, we aim to illustrate that both measurements (values and distances) are important in skill stitching. We compare our VG-VLD with the following two variants:

- **Value-Only.** This means the evaluator only fits the values judged by the critics of next skill policy.
- **Distance-Only**. This means the evaluator only fits the two distance terms in Equation (9).

We conduct this experiment on the Kitchen environment, with the other settings remaining same as the main experiment. The results for this ablation are listed in Table 8 in Appendix E. While both **Value-Only** and **Distance-Only** outperforms the baselines in Table 2 overall, VG-VLD achieves a much better performance than these two variants in most of the tasks. These results show that although simply using values or distances can be helpful, combining both can provide a better insight for the process of skill stitching. In essence, the two variants can be complements of each other: **values** measure the progress of accomplishing the task, while they suffer the problem of distribution shift at test time; temporal-preserving **distances** provide a straightforward measurement of two states, yet a numerically smaller distance does not necessarily mean the states are closer due to possible data overlap. Therefore, combining the two can leverage the advantages of both measurements.

6 Conclusion and Limitations

In this paper, we propose VG-VLD, a method with variational latent dynamics to tackle the problems of offline skill stitching. With our variational latent auto-encoder, translator, and evaluator learned from the offline skill-level datasets, our method demonstrates significant advantages, outperforming various baselines including generative and planning-based methods. During inference, our VG-VLD is effective in time by latent sampling and score-based value guidance, with the potential for real-world deployment.

AI USE STATEMENT

In this work, we used generative AI tools for [formulate mathematical claims]. We have not used generative AI tools for [generate synthetic data sets, help develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results], and [provide critical ingredients for proving mathematical claims, assist in the writing of proofs] are not applicable to this work. Additionally, we used generative AI tools for [edit a research paper to improve readability]. We have reviewed all AI-assisted work. We checked LLM-generated mathematical claims and formulas, and LLM-generated text polishment was verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The main text defines the algorithm, model, and evaluation. Appendices provide benchmark details, training settings, and additional results. We will release the code to support reproduction after acceptance.

REFERENCES

Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023.

Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019.

Yuanpei Chen, Chen Wang, Li Fei-Fei, and Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. In *Conference on Robot Learning*, pp. 3809–3829. PMLR, 2023.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67, 2005.

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

Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-skill mobile manipulation for object rearrangement. In *The Eleventh International Conference on Learning Representations*, 2023.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Ziye Huang, Haoqi Yuan, Yuhui Fu, and Zongqing Lu. Efficient residual learning with mixture-of-experts for universal dexterous grasping. In *The Thirteenth International Conference on Learning Representations*, 2025.

Minjae Kang and Songhwai Oh. Deep latent-space sequential skill chaining from incomplete demonstrations. *Intelligent Service Robotics*, 15(2):203–213, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22, 2009.

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

Kyowoon Lee and Jaesik Choi. State-covering trajectory stitching for diffusion planners. *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2019.

Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In *5th Annual Conference on Robot Learning*, 2021.

Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *International Conference on Machine Learning*, pp. 28597–28609. PMLR, 2024.

Jinxin Liu, Xinghong Guo, Zifeng Zhuang, and Donglin Wang. Didi: Diffusion-guided diversity for offline behavioral generation. In *International Conference on Machine Learning*, pp. 31032–31047. PMLR, 2024.

Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, 2023.

Utkarsh A. Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *arXiv preprint arXiv:2401.03360*, 2023.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *The Thirteenth International Conference on Learning Representations*, 2025.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Yunpeng Qing, Shuo Chen, Yixiao Chi, Shunyu Liu, Sixu Lin, Kelu Yao, and Changqing Zou. Bitrajdiff: Bidirectional trajectory generation with diffusion models for offline reinforcement learning. *arXiv preprint arXiv:2506.05762*, 2025.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

Qianlan Yang and Yu-Xiong Wang. Rtdiff: Reverse trajectory synthesis via diffusion for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hang Yu, Di Zhang, Qiwei Du, Yanping Zhao, Hai Zhang, Guang Chen, Eduardo E Veas, and Junqiao Zhao. Astro: Adaptive stitching via dynamics-guided trajectory rollouts. *arXiv preprint arXiv:2511.23442*, 2025.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Skill reinforcement learning and planning for open-world long-horizon tasks. *arXiv preprint arXiv:2303.16563*, 2023.

Haoqi Yuan, Zhancun Mu, Feiyang Xie, and Zongqing Lu. Pre-training goal-based models for sample-efficient reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Haoqi Yuan, Ziye Huang, Ye Wang, Chuan Mao, Chaoyi Xu, and Zongqing Lu. Demograsp: Universal dexterous grasping from a single demonstration. *arXiv preprint arXiv:2509.22149*, 2025a.

Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. Efficient multi-agent offline coordination via diffusion-based trajectory stitching. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Zilai Zeng, Ce Zhang, Shijie Wang, and Chen Sun. Goal-conditioned predictive coding for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:25528–25548, 2023.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Pseudo Code of Our Algorithm

In this section, we provide the pseudo code of our method VG-VLD, including the training phase (Algorithm 1) and inference phase (Algorithm 2).

Algorithm 1 VG-VLD: Training Phase

Input: Temporal-chunked CVAE with encoder q_ϕ and decoder p_ψ, translator f_θ, evaluator S_ω, offline trained skill policies $\{\pi_i\}_{i=1}^n$, offline datasets $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^n$
Initialize temporal-chunked CVAE with encoder q_ϕ and decoder p_ψ, translator f_θ, evaluator S_ω, offline trained skill policies $\{\pi_i\}_{i=1}^n$
Train skill policies $\{\pi_i\}_{i=1}^n$ from offline datasets $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^n$ using implicit Q-learning (IQL) (Kostrikov et al., 2022)
Train temporal-chunked CVAE (q_ϕ and p_ψ) according to Equation (5)
Train translator f_θ according to Equation (6)
Annotate the datasets $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^n$ with assigned scores calculated by Equation (9)
Train evaluator S_ω according to Equation (10)

Algorithm 2 VG-VLD: Inference Phase

Require: Temporal-chunked CVAE with encoder q_ϕ and decoder p_ψ, translator f_θ, evaluator S_ω, population size N, offline trained skill policies $\{\pi_i\}_{i=1}^n$
Given a long-horizon task, denoted as a skill sequence $\pi_1, ..., \pi_n$
Reset the environment and get initial state s_0
Current state $s_t \leftarrow s_0$
for each skill i in the long-horizon task **do**
  The estimated score of current state $v_{\text{new}} \leftarrow S_\omega(s_t, i+1)$
  repeat
    $v_{\text{old}} \leftarrow v_{\text{new}}$ {Stitching Phase: add connections from skill i to skill $i+1$}
    Sample a population of random latent variables $\{z_1, z_2, ..., z_N\} \sim \mathcal{N}(0, \mathbf{I})$
    For each z_j, predict the future state $s_{t+H}^j \sim p_\psi(s_{t+H}|s_t, z_j)$
    Evaluate the scores of those future states $\{S_\omega(s_{t+H}^1, i+1), ..., S_\omega(s_{t+H}^N, i+1)\}$
    Choose the best elite z_t according to estimated scores
    Translate latent variable into action chunks $A_t = f_\theta(s_t, z_t)$
    for each action a_t in action chunk A_t **do**
      Execute a_t in the environment and get a new state s'
      $s_t \leftarrow s'$
      The score of current state $v_{\text{new}} \leftarrow S_\omega(s_t, i+1)$
      if $v_{\text{new}} < v_{\text{old}}$ **then**
        Break (early-stopping the stitching phase)
      end if
    end for
  until $v_{\text{new}} < v_{\text{old}}$ or maximum total steps of testing exceeded
  repeat
    Execute policy π_i in the environment {Execution Phase: accomplish skill i}
  until success or reaching maximum steps of testing
end for

B Benchmark Details

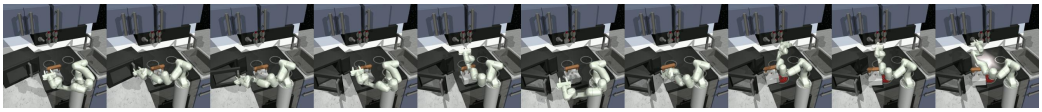
In this section, we provide further introductions of the benchmarks we use, including OGBenchMaze and Kitchen. A visualization of the two environments can be found in Figure 5.

B.1 OGBenchMaze

OGBenchMaze is the maze environment provided in OGBench (Park et al., 2025), a benchmark and testbed for offline goal-conditioned reinforcement learning. The agents is controlled to navigate in the



(a) Visualization of PointMaze environment in OGBench (the 3 rows above representing medium, large, and giant, respectively). A point agent (yellow) navigates in the maze towards the goals (pink) in order.



(b) Visualization of Kitchen environment. A 9-DoF Franka Panda robot placed is set to perform the manipulation tasks in order. From left to right: microwave → bottom burner → kettle → light switch.

Figure 5: Visualization of our agents performing long-horizon tasks in the two environments used in this paper: (a) OGBenchMaze, and (b) Kitchen. Each line shows a progress of our agent accomplishing the multi-skill long-horizon task, sliced by timesteps. Notably, our method achieves non-zero success rate in stitching **four skills** in Kitchen, which was only achieved by online methods (Pertsch et al., 2021) that require intensive interactions with the environment previously. We further provide demonstration videos in the supplementary materials.

maze to reach several pre-defined goal locations. The observation space includes self proprioception (joints) and the current position (x and y), while the action space controls the angles and velocities of each joint. We utilize the PointMaze environment, which has three variants in size: medium, large, and giant. For each variant, we choose three goal locations to form three skills:

- **Medium:** starting from $(0.0, 12.0)$, three goals: $(4.0, 0.0) \rightarrow (12.0, 16.0) \rightarrow (20.0, 0.0)$
- **Large:** starting from $(4.0, 16.0)$, three goals: $(4.0, 8.0) \rightarrow (8.0, 0.0) \rightarrow (20.0, 12.0)$
- **Giant:** starting from $(20.0, 16.0)$, three goals: $(28.0, 24.0) \rightarrow (44.0, 20.0) \rightarrow (52.0, 12.0)$

To adapt to our setting of skill-level stitching, we extract only the trajectories that terminate at these goals from the original dataset (`medium-stitch-v0`, `large-stitch-v0`, and `giant-stitch-v0`), to form our dataset $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{3}$. We use such datasets to train our skill policy (via IQL), and to train our VG-VLD modules. During test time, we adopt a maximum steps of 4000, and once reaching this maximum steps, the termination condition will be triggered and this episode will be terminated.

B.2 KITCHEN

Kitchen implements a 9-DoF Franka Panda robot placed in a simulated kitchen environment to manipulate several household items (Fu et al., 2020). There are seven skills in this environment originally, each corresponding to interacting with an item to reach a target state. By default, the robot always starts from a fixed posture. The 59-dimensional observation space of Kitchen includes the proprioception of the robot arm and the states of the seven objects, while the 9-dimensional action space is constructed with the control of the velocity of the joints and the gripper fingers. The seven skills (Fu et al., 2020) are defined as:

1. `Bottom burner:` twist the bottom control knob to activate the bottom burner of the stove.
2. `Top burner:` twist the top control knob to activate the top burner of the stove.
3. `Light switch:` twist the switch to turn on the light.
4. `Slide cabinet:` slide the door of the cabinet to the right to open it.

Table 3: The hyperparameters used in the IQL algorithm.

Hyperparameters	Value
learning rate	3×10^{-4}
dropout rate	0.1
learning rate decay	True
γ	0.99
τ	0.005
expectile	0.7
temperature	0.5
batch size	256

5. `Hinge cabinet`: open the door of a hinge cabinet by rotating it.
6. `Microwave`: open the door of the microwave.
7. `Kettle`: move the kettle from the bottom burner onto the top burner.

In this paper, we assume access to datasets of 4 (out of 7) skills (`bottom burner`, `light switch`, `microwave`, and `kettle`), each containing the trajectories of a single skill. In the Kitchen benchmark, the long-horizon tasks are the combinations of several skills, e.g., `bottom burner` $\rightarrow$ `kettle` $\rightarrow$ `microwave` is a long-horizon task with three skills. During test time, we adopt a maximum step for each long-horizon task (1000 steps if the task is composed of 2 skills; 1500 steps for 3 skills; 2000 steps for 4 skills). Once reaching this maximum steps, the termination condition will be triggered and this episode will be terminated.

C Implementation Details

In this section, we provide details of implementing our method, VG-VLD. VG-VLD is composed of three stages: training policies of each skill (Appendix C.1), learning three modules of VG-VLD (Appendix C.2), as well as test-time inference under a score-based guidance (Appendix C.3).

C.1 Training Policies of Each Skill

We use implicit Q-learning (IQL) (Kostrikov et al., 2022) to train the policies of each skill. Specifically, we train a multi-task policy for the OGBenchMaze and Kitchen environment, by concatenating a one-hot vector with the policy observation. For instance, we have three skills in OGBenchMaze. We construct 3-dimensional vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ to represent three skills respectively. We list the related hyperparameters for the IQL algorithm in Table 3.

C.2 Details of VG-VLD Modules

C.2.1 Training Conditional Variational Auto-Encoder

Our CVAE module is trained according to Equation (5). For each state pair (s_t, s_{t+H}), the encoder q_ϕ encodes $\mathrm{concat}([s_t, s_{t+H}])$ into a latent variable z, while the decoder p_ψ decodes $\mathrm{concat}([s_t, z])$ into a reconstructed future state $\hat{s}_{t+H}$. We implement both encoder and decoder with multi-layer perceptrons (MLPs), and list the related hyperparameters for training our CVAE module in Table 4.

C.2.2 Training Translator

Our translator is trained according to Equation (6). For each chunk-level sub-trajectory $(s_t, a_t, s_{t+1}, a_{t+1}, ..., s_{t+H})$, we encode $\mathrm{concat}([s_t, s_{t+H}])$ into a latent variable z, and translates $\mathrm{concat}([s_t, z])$ into an action chunk $A_t = [a_t, a_{t+1}, ..., a_{t+H-1}]$. We implement our translator with an MLP, and train it in a supervised manner, with the related hyperparameters listed in Table 5.

Table 4: The hyperparameters used in training our conditional variational auto-encoder.

Hyperparameters	Value
latent dim	64
input dim of encoder q_ϕ	$2 \times$ observation dim
output dim of encoder q_ϕ	latent dim
input dim of decoder p_ψ	observation dim + latent dim
output dim of decoder p_ψ	observation dim
learning rate	1×10^{-3}
beta (weight for KL divergence)	1.0
batch size	2048
chunk size H	16
epochs	100

Table 5: The hyperparameters used in training our translator.

Hyperparameters	Value
chunk size	16
hidden dim	256
input dim	observation dim + latent dim
output dim	action dim $\times$ chunk size
learning rate	1×10^{-3}
batch size	2048
epochs	100

C.2.3 Training Evaluator

In order to train our evaluator, we need to annotate the offline datasets with state scores according to Equation (9), and the evaluator is learned to fit the annotated score using Equation (10). We list the hyperparameters in this stage in Table 6.

C.3 Score-Based Value-Guided Inference

Having trained all the modules of VG-VLD, we can perform skill stitching at test time. During inference, we follow the process in Algorithm 2. We list the hyperparameters in inference phase in Table 7.

D Details of Baselines

In this section, we provide the details of implementing the baselines used in this paper. Some of the baselines should be properly adjusted to adapt to our experimental setting, which we will introduce in this part.

D.1 No-Stitching

For the most basic baseline, **No-Stitching**, we directly execute the skills one after another without the stitching process. We set a maximum number of steps for each rollout as in Table 7, and once reaching this maximum steps, the termination condition will be triggered and this episode will be terminated. During inference, a skill is continuously executed (select an action and interact with the environment) until success or max-step termination, and if the success happens before max-step termination, the policy is switched to the next skill until all the skills are done to accomplish the whole long-horizon task.

Table 6: The hyperparameters used in training our evaluator.

Hyperparameters	Value
learning rate	3×10^{-4}
batch size	256
epochs	100
γ in Equation (8)	0.99
τ in Equation (8)	0.95
H in Equation (8)	16

Table 7: The hyperparameters during inference phase.

Hyperparameters	Value
population size N to sample latent variables	100,000
action chunk size H	16
max steps before termination	4000 (OGBenchMaze, 3 skills) 1000 (Kitchen, 2 skills) 1500 (Kitchen, 3 skills) 2000 (Kitchen, 4 skills)

D.2 VANILLA CEM

Vanilla CEM is a standard baseline using Cross-Entropy Method (CEM) (Rubinstein, 1999; De Boer et al., 2005) for stitching, which is a kind of planning method. Apart from the IQL policies of each skill, we need to train a dynamics model which takes in s_t and a_t, and predicts the next state s_{t+1}. In the stitching process, we assume the actions follow a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with initialization $\mu = 0, \sigma = \mathbf{I}$, and gradually updates μ and σ during planning. Within a loop, each time we sample a population of actions (the population size is the same as our VG-VLD, as in Table 7), and predict the next states with our learned dynamics model. Next, we select the elite states (top-k) according to their value estimated by the value critic network of the next IQL skill. These elite states correspond to the elite actions, and we calculate the mean and standard deviation of these k actions to update μ and σ. The planning loop continues until the value converges, that is, the newly estimated values are not higher than the values from the last loop, which is consistent with $v_{new} < v_{old}$ in our Algorithm 2. Finally, the converged μ is taken as the real action to be executed in the environment.

D.3 STAP

STAP (Agia et al., 2023) proposes **Policy CEM**, a similar method built base on CEM planning. The main differences between **STAP** and **Vanilla CEM** are:

- The initialization of μ for the Gaussian distribution is defined as $\mu \sim \pi_{i+1}(a_t|s_t)$, where π_{i+1} is the next skill policy. This is different from Vanilla CEM in which μ starts from 0.
- During elite selection, the metric is not the value critic $V^\pi(s')$, but the Q critic $Q^\pi(s_t, a_t)$ instead. This means STAP does not need to train a dynamics model, which avoids the distribution shift and accumulated error.

As planning-based methods, **Vanilla CEM** and **STAP** are time-consuming, since a loop until convergence brings a heavy load of time (one loop before a real action is taken in the environment), and many loops are consumed in the stitching phase.

D.4 DIDI

Different from those planning-based methods, **DIDI** (Liu et al., 2024) advocates a single network for both policy learning and dynamics modeling, avoiding compounding rollout errors over an

additional learned proxy dynamics model. This paper formulates such a network as a sequence model $\pi_\theta(\tau_t|s_t, z)$, where $\tau_t = [s_t, a_t, s_{t+1}, ..., s_T, a_T]$, and z denotes a latent skill representation. However, it only executes $a_t \in \tau_t$ during inference phase and turns to the next inference, which is wasteful in terms of computation and time. In contrast, our method stitches adjacent policies by translating a latent action sampled from the latent space into an action chunk, and execute all the actions in the chunk (or move towards the next inference if all the actions in the current chunk are executed) until termination, which is more efficient.

For this baseline, we train π_θ as it is in the DIDI paper, and the adjacent skills are still IQL policies to adapt to our setting. During stitching phase, at each time step t, we sample $z \sim \mathcal{N}(0, \mathbf{I})$ and generate $\tau_t \sim \pi_\theta(\tau_t|s_t, z)$, then execute $a_t \in \tau_t$, and move to the next time step $t + 1$. This is repeated until termination (which is guided by the value critic of the next IQL skill).

D.5 DIFFSTITCH

DiffStitch (Li et al., 2024) proposes a diffusion-based trajectory stitching method to stitch sub-trajectories sampled from different behavior policies. However, it requires training a single diffusion model after proper data augmentations in terms of state coverage, which is different from our approach of training multiple IQL policies and stitching them together. To adapt to our setting, we still use the IQL policies trained on different sub-datasets as in our method, and train an inverse dynamics model (IDM) f_ψ as well as a diffusion-based generative model $g_\theta(\tau|[s_T, \text{MASK}, \text{MASK}, ...])$ as in DiffStitch paper.

Assume the current skill i is just succeeded, and the next skill is $i + 1$. During stitching time, we search among all the trajectories τ^{i+1} from the sub-dataset corresponding to skill $i + 1$, where $\tau^{i+1} = [s_0^{i+1}, a_0^{i+1}, r_0^{i+1}, ..., s_T^{i+1}]$, and sample an end state s_T^{i+1} from these demonstrative trajectories. Standing at the current state s_t, we construct a masked trajectory $\tau_{s,m} = [s_t, \text{MASK}, \text{MASK}, ..., \text{MASK}, s_T^{i+1}]$, where each MASK is a sampled Gaussian noise, and the number of MASK is chunk size minus 1 ($H - 1$), with the hyperparameter "chunk size" same as in Table 7. Then, leveraging the diffusion-based generative model, we denoise the masked trajectory into $\tau_s^s = g_\theta(\tau_{s,m}) = [s_t, s_{t+1}, ..., s_{t+H-1}, s_T^{i+1}]$. Finally, we extract the action sequence $A_t = [a_t, a_{t+1}, ..., a_{t+H-1}]$ from τ_s^s using the trained IDM f_ψ, and execute these actions in the environment.

D.6 SCOTS

SCoTS (Lee & Choi, 2025) tackles the stitching problem by data augmentation. A diffusion stitcher is trained on the original datasets, and is used to generate new sub-trajectories as the stitching phases, aggregated in the augmented dataset. Afterwards, a diffusion planner is trained on the augmented dataset, and executed for the whole long-horizon task.

For this baseline, we train the diffusion stitcher as the SCoTS paper (Lee & Choi, 2025), and gather all the generated trajectories in a buffer. During the test phase, we run each IQL skill policies as other baselines, and sample those trajectories from the buffer to serve as the stitching process.

D.7 ASTRO

Based on SCoTS, ASTRO (Yu et al., 2025) adds a dynamics diffuser and a corrector, which are used to select and refine the generated trajectories before they are sent to serve as the stitching process.

Similarly, we also collect the augmented trajectories in a buffer, and sample one for the stitching phase at each test time.

D.8 BITRAJDIFF

BiTrajDiff (Qing et al., 2025) also formulates the stitching problem as data augmentation, but uses a bidirectional generation scheme. Specifically, a forward and a backward diffusion generator work together to generate both forward and backward state trajectories for a given state. Finally, an inverse dynamics model (IDM) trained with the original datasets is used to annotate actions for these state trajectories.

Table 8: Ablation results of our evaluator on Kitchen environment. We compare our VG-VLD with two other methods, value-only and distance-only. For each long-horizon task composed of two skills, the maximum value of AR is 2.00. We test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation of AR and CSR-2 (denoted as mean±std). The last two rows show the average value of AR and CSR-2 of each method. The highest values are represented in bold.

Task	Metric	Value-Only	Distance-Only	VG-VLD (Ours)
bottom burner → light switch	Average Return (AR)	1.37 ± 0.08	1.47 ± 0.03	**1.57 ± 0.02**
	Conditional Success Rate (CSR) - 2	0.447 ± 0.033	0.523 ± 0.046	**0.627 ± 0.037**
bottom burner → microwave	Average Return (AR)	1.72 ± 0.04	1.71 ± 0.05	**1.74 ± 0.03**
	Conditional Success Rate (CSR) - 2	0.826 ± 0.040	0.786 ± 0.020	**0.801 ± 0.014**
bottom burner → kettle	Average Return (AR)	1.27 ± 0.07	1.19 ± 0.03	**1.36 ± 0.03**
	Conditional Success Rate (CSR) - 2	0.333 ± 0.052	0.221 ± 0.025	**0.372 ± 0.046**
microwave → bottom burner	Average Return (AR)	1.48 ± 0.04	1.69 ± 0.03	**1.77 ± 0.02**
	Conditional Success Rate (CSR) - 2	0.564 ± 0.025	0.801 ± 0.015	**0.894 ± 0.022**
microwave → light switch	Average Return (AR)	1.14 ± 0.10	**1.63 ± 0.04**	1.61 ± 0.05
	Conditional Success Rate (CSR) - 2	0.246 ± 0.058	**0.802 ± 0.014**	0.790 ± 0.040
microwave → kettle	Average Return (AR)	1.38 ± 0.09	1.57 ± 0.03	**1.69 ± 0.01**
	Conditional Success Rate (CSR) - 2	0.514 ± 0.062	0.721 ± 0.049	**0.767 ± 0.004**
kettle → bottom burner	Average Return (AR)	0.56 ± 0.03	0.74 ± 0.04	**0.82 ± 0.06**
	Conditional Success Rate (CSR) - 2	0.376 ± 0.075	0.609 ± 0.095	**0.615 ± 0.131**
kettle → light switch	Average Return (AR)	0.51 ± 0.03	0.54 ± 0.16	**0.65 ± 0.04**
	Conditional Success Rate (CSR) - 2	0.360 ± 0.117	0.368 ± 0.024	**0.443 ± 0.063**
kettle → microwave	Average Return (AR)	0.48 ± 0.04	0.53 ± 0.02	**0.59 ± 0.04**
	Conditional Success Rate (CSR) - 2	0.092 ± 0.061	0.288 ± 0.036	**0.320 ± 0.022**
Average	Average Return (AR)	1.10	1.23	**1.31**
	Conditional Success Rate (CSR) - 2	0.418	0.569	**0.625**

For this baseline, we generate augmented trajectories for each sampled state for each skill, and execute a sampled trajectory from the corresponding skill during the test phase.

E Additional Results

In this section, we provide the detailed results corresponding to the ablation studies on our evaluator, as listed in Table 8. we also provide additional results on long-horizon tasks composed of 3 or 4 skills in the Kitchen environment, as a supplement to the main results in Table 2. The corresponding results are listed in Table 9 and Table 10.

Table 9: Additional experimental results on Kitchen environment. For each long-horizon task composed of three skills, the maximum value of AR is 3.00. We test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation of AR, CSR-2, and CSR-3 (denoted as mean±std). The last three rows show the average value of AR and CSRs of each method.

Task	Metric	No-Stitching	Vanilla CEM	STAP	DIDI	DiffStitch	SCoTS	ASTRO	BiTrajDiff	VG-VLD (ours)
bottom burner → light switch → microwave	Average Return (AR)	1.44±0.07	1.03±0.04	1.00±0.03	1.12±0.08	1.12±0.07	1.03 ± 0.02	1.09±0.03	1.30±0.06	1.55±0.02
	Conditional Success Rate (CSR) - 2	0.498±0.049	0.088±0.014	0.046±0.025	0.182±0.049	0.166±0.045	0.074 ± 0.015	0.139±0.012	0.371±0.043	0.638±0.031
	Conditional Success Rate (CSR) - 3	0.000±0.000	0.000±0.000	0.048±0.083	0.000±0.000	0.000±0.000	0.093 ± 0.069	0.000±0.000	0.009±0.016	0.006±0.013
bottom burner → light switch → kettle	Average Return (AR)	1.39±0.06	1.04±0.04	1.00±0.03	1.12±0.01	1.12±0.05	1.01 ± 0.02	1.11±0.02	1.33±0.03	1.60±0.02
	Conditional Success Rate (CSR) - 2	0.526±0.065	0.094±0.019	0.065±0.038	0.182±0.013	0.160±0.030	0.078 ± 0.013	0.145±0.023	0.400±0.054	0.611±0.011
	Conditional Success Rate (CSR) - 3	0.021±0.019	0.000±0.000	0.033±0.058	0.000±0.000	0.021±0.036	0.000 ± 0.000	0.000±0.000	0.000±0.000	0.000±0.021
bottom burner → microwave → light switch	Average Return (AR)	1.01±0.07	1.46±0.09	1.41±0.05	1.82±0.06	1.63±0.08	1.62 ± 0.05	1.67±0.04	1.76±0.02	1.85±0.03
	Conditional Success Rate (CSR) - 2	0.101±0.023	0.493±0.037	0.496±0.077	0.875±0.028	0.685±0.089	0.661 ± 0.030	0.753±0.011	0.807±0.020	0.852±0.052
	Conditional Success Rate (CSR) - 3	0.072±0.027	0.070±0.041	0.029±0.012	0.090±0.028	0.041±0.037	0.052 ± 0.053	0.024±0.014	0.021±0.014	0.118±0.015
bottom burner → microwave → kettle	Average Return (AR)	1.12±0.09	1.59±0.02	1.54±0.15	2.19±0.08	1.81±0.01	1.71 ± 0.09	1.85±0.14	2.00±0.04	2.11±0.04
	Conditional Success Rate (CSR) - 2	0.079±0.051	0.518±0.030	0.425±0.093	0.835±0.045	0.662±0.042	0.578 ± 0.032	0.640±0.065	0.777±0.034	0.858±0.016
	Conditional Success Rate (CSR) - 3	0.415±0.042	0.322±0.060	0.429±0.052	0.453±0.059	0.418±0.116	0.436 ± 0.073	0.455±0.059	0.403±0.030	0.496±0.002
bottom burner → kettle → light switch	Average Return (AR)	1.11±0.09	1.27±0.01	1.22±0.08	1.23±0.02	1.20±0.03	1.25 ± 0.05	1.26±0.01	1.01±0.02	1.31±0.03
	Conditional Success Rate (CSR) - 2	0.180±0.035	0.366±0.015	0.305±0.013	0.290±0.037	0.261±0.021	0.279 ± 0.043	0.310±0.023	0.064±0.012	0.377±0.052
	Conditional Success Rate (CSR) - 3	0.022±0.039	0.010±0.017	0.034±0.036	0.060±0.016	0.085±0.047	0.038 ± 0.006	0.067±0.024	0.122±0.107	0.083±0.033
bottom burner → kettle → microwave	Average Return (AR)	1.12±0.07	1.38±0.07	1.22±0.03	1.28±0.08	1.29±0.08	1.22 ± 0.02	1.20±0.06	1.04±0.01	1.34±0.02
	Conditional Success Rate (CSR) - 2	0.196±0.035	0.418±0.066	0.254±0.014	0.303±0.048	0.305±0.053	0.258 ± 0.010	0.238±0.047	0.073±0.029	0.409±0.014
	Conditional Success Rate (CSR) - 3	0.021±0.036	0.048±0.021	0.095±0.017	0.229±0.177	0.271±0.034	0.198 ± 0.079	0.194±0.035	0.204±0.080	0.192±0.044
microwave → bottom burner → light switch	Average Return (AR)	1.83±0.09	1.49±0.07	1.57±0.05	1.41±0.06	1.46±0.06	1.47 ± 0.03	1.56±0.01	1.65±0.02	2.02±0.10
	Conditional Success Rate (CSR) - 2	0.779±0.039	0.626±0.061	0.652±0.004	0.470±0.031	0.527±0.029	0.647 ± 0.029	0.633±0.029	0.651±0.049	0.854±0.039
	Conditional Success Rate (CSR) - 3	0.316±0.068	0.046±0.026	0.107±0.018	0.113±0.054	0.154±0.030	0.096 ± 0.020	0.136±0.051	0.205±0.011	0.378±0.001
microwave → bottom burner → kettle	Average Return (AR)	1.66±0.10	1.61±0.10	1.61±0.07	1.49±0.04	1.49±0.03	1.56 ± 0.03	1.68±0.08	1.59±0.07	1.86±0.04
	Conditional Success Rate (CSR) - 2	0.770±0.058	0.628±0.036	0.667±0.100	0.537±0.055	0.509±0.008	0.575 ± 0.053	0.691±0.032	0.669±0.015	0.819±0.021
	Conditional Success Rate (CSR) - 3	0.156±0.056	0.189±0.068	0.170±0.032	0.214±0.009	0.200±0.090	0.257 ± 0.031	0.208±0.077	0.150±0.027	0.229±0.040
microwave → light switch → bottom burner	Average Return (AR)	1.97±0.05	1.05±0.02	1.05±0.03	1.13±0.02	1.14±0.00	1.15 ± 0.02	1.15±0.07	1.01±0.03	1.82±0.08
	Conditional Success Rate (CSR) - 2	0.901±0.029	0.096±0.019	0.146±0.004	0.222±0.034	0.213±0.042	0.251 ± 0.013	0.277±0.073	0.086±0.003	0.828±0.034
	Conditional Success Rate (CSR) - 3	0.297±0.030	0.214±0.079	0.024±0.041	0.035±0.031	0.195±0.076	0.074 ± 0.022	0.084±0.037	0.042±0.072	0.321±0.012
microwave → light switch → kettle	Average Return (AR)	1.73±0.02	1.01±0.04	1.03±0.03	1.19±0.09	1.19±0.09	1.10 ± 0.00	1.18±0.06	0.99±0.02	1.63±0.03
	Conditional Success Rate (CSR) - 2	0.898±0.024	0.139±0.040	0.146±0.041	0.337±0.059	0.280±0.069	0.197 ± 0.037	0.302±0.042	0.072±0.017	0.792±0.066
	Conditional Success Rate (CSR) - 3	0.000±0.000	0.000±0.000	0.028±0.048	0.000±0.000	0.014±0.025	0.000 ± 0.000	0.000±0.000	0.000±0.000	0.046±0.026
microwave → kettle → bottom burner	Average Return (AR)	2.12±0.12	1.18±0.05	1.24±0.07	1.51±0.11	1.35±0.12	1.33 ± 0.03	1.43±0.11	1.50±0.06	2.22±0.02
	Conditional Success Rate (CSR) - 2	0.728±0.041	0.322±0.056	0.328±0.065	0.571±0.060	0.438±0.045	0.418 ± 0.009	0.463±0.023	0.598±0.038	0.804±0.038
	Conditional Success Rate (CSR) - 3	0.752±0.038	0.057±0.035	0.176±0.059	0.177±0.050	0.138±0.074	0.161 ± 0.025	0.206±0.094	0.111±0.017	0.829±0.053
microwave → kettle → light switch	Average Return (AR)	1.91±0.08	1.20±0.09	1.24±0.08	1.47±0.07	1.40±0.08	1.30 ± 0.01	1.43±0.04	1.46±0.08	2.02±0.07
	Conditional Success Rate (CSR) - 2	0.742±0.037	0.279±0.055	0.338±0.036	0.531±0.038	0.429±0.077	0.446 ± 0.018	0.471±0.032	0.590±0.037	0.779±0.062
	Conditional Success Rate (CSR) - 3	0.549±0.052	0.087±0.084	0.161±0.099	0.174±0.004	0.176±0.006	0.140 ± 0.036	0.187±0.044	0.095±0.030	0.694±0.008
kettle → bottom burner → light switch	Average Return (AR)	0.64±0.17	0.39±0.03	0.55±0.09	0.54±0.09	0.52±0.08	0.52 ± 0.07	0.55±0.05	0.48±0.05	0.85±0.08
	Conditional Success Rate (CSR) - 2	0.460±0.072	0.036±0.040	0.108±0.033	0.146±0.091	0.189±0.026	0.143 ± 0.018	0.164±0.065	0.101±0.030	0.539±0.057
	Conditional Success Rate (CSR) - 3	0.375±0.050	0.000±0.000	0.208±0.260	0.148±0.170	0.239±0.067	0.048 ± 0.067	0.150±0.049	0.250±0.083	0.328±0.011
kettle → bottom burner → microwave	Average Return (AR)	0.64±0.04	0.46±0.07	0.50±0.05	0.56±0.06	0.49±0.08	0.49 ± 0.07	0.53±0.04	0.49±0.03	0.80±0.05
	Conditional Success Rate (CSR) - 2	0.590±0.046	0.037±0.010	0.071±0.031	0.137±0.030	0.161±0.025	0.140 ± 0.015	0.212±0.028	0.153±0.054	0.572±0.038
	Conditional Success Rate (CSR) - 3	0.059±0.070	0.000±0.000	0.167±0.289	0.519±0.162	0.204±0.081	0.131 ± 0.102	0.133±0.112	0.125±0.125	0.074±0.022
kettle → light switch → bottom burner	Average Return (AR)	0.59±0.11	0.49±0.03	0.49±0.07	0.46±0.08	0.51±0.08	0.46 ± 0.02	0.54±0.06	0.47±0.03	0.76±0.03
	Conditional Success Rate (CSR) - 2	0.284±0.012	0.112±0.099	0.170±0.027	0.082±0.061	0.123±0.058	0.107 ± 0.051	0.116±0.033	0.032±0.037	0.486±0.015
	Conditional Success Rate (CSR) - 3	0.107±0.044	0.000±0.000	0.111±0.096	0.000±0.000	0.056±0.096	0.095 ± 0.135	0.095±0.135	0.111±0.192	0.175±0.046
kettle → light switch → microwave	Average Return (AR)	0.60±0.07	0.43±0.07	0.51±0.02	0.45±0.03	0.50±0.02	0.56 ± 0.04	0.51±0.04	0.48±0.08	0.68±0.08
	Conditional Success Rate (CSR) - 2	0.326±0.041	0.082±0.026	0.171±0.108	0.097±0.064	0.089±0.048	0.142 ± 0.053	0.144±0.029	0.034±0.039	0.497±0.039
	Conditional Success Rate (CSR) - 3	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.167±0.289	0.037 ± 0.052	0.000±0.000	0.000±0.000	0.000±0.000
kettle → microwave → bottom burner	Average Return (AR)	0.63±0.13	0.52±0.02	0.51±0.07	0.65±0.13	0.52±0.09	0.47 ± 0.05	0.50±0.04	0.47±0.11	0.73±0.03
	Conditional Success Rate (CSR) - 2	0.359±0.048	0.071±0.012	0.103±0.030	0.288±0.062	0.168±0.075	0.154 ± 0.019	0.158±0.033	0.069±0.008	0.368±0.077
	Conditional Success Rate (CSR) - 3	0.552±0.119	0.500±0.500	0.389±0.347	0.304±0.244	0.350±0.087	0.222 ± 0.078	0.439±0.172	0.500±0.500	0.549±0.055
kettle → microwave → light switch	Average Return (AR)	0.72±0.11	0.52±0.07	0.53±0.07	0.66±0.13	0.57±0.04	0.52 ± 0.03	0.52±0.08	0.45±0.05	0.65±0.07
	Conditional Success Rate (CSR) - 2	0.399±0.125	0.149±0.059	0.129±0.027	0.241±0.077	0.201±0.009	0.162 ± 0.075	0.158±0.036	0.081±0.032	0.323±0.106
	Conditional Success Rate (CSR) - 3	0.400±0.132	0.122±0.107	0.083±0.144	0.296±0.011	0.174±0.156	0.380 ± 0.203	0.236±0.098	0.111±0.192	0.511±0.117
Average	Average Return (AR)	1.23	1.01	1.01	1.13	1.07	1.04	1.10	1.08	**1.43**
	Conditional Success Rate (CSR) - 2	0.490	0.253	0.257	0.351	0.309	0.295	0.334	0.313	**0.634**
	Conditional Success Rate (CSR) - 3	0.229	0.098	0.127	0.156	0.161	0.137	0.145	0.137	**0.279**

Table 10: Additional experimental results on Kitchen environment. For each long-horizon task composed of four skills, the maximum value of AR is 4.00. We test each method on 3 seeds, each for 100 times, and calculate the mean and standard deviation of AR, CSR-2, CSR-3, and CSR-4 (denoted as mean±std). The last four rows show the average value of AR and CSRs of each method. For some cases where CSR-3= 0, which means the 3-rd skill is never accomplished, we use "NaN" for the CSR-4 since it is impossible to accomplish the 4-th skill.

Task	Metric	No-Stitching	Vanilla CEM	STAP	DIDI	DiffStitch	SCoTS	ASTRO	BiTrajDiff	VG-VLD (ours)
bottom burner → light switch → microwave → kettle	Average Return (AR)	1.44±0.02	1.03±0.02	1.00±0.06	1.11±0.05	1.09±0.05	1.02 ± 0.02	1.04±0.03	1.32±0.04	1.62±0.02
	Conditional Success Rate (CSR) - 2	0.551±0.045	0.104±0.028	0.045±0.031	0.159±0.012	0.123±0.036	0.102 ± 0.016	0.106±0.022	0.384±0.059	0.600±0.031
	Conditional Success Rate (CSR) - 3	0.006±0.011	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.042 ± 0.059	0.000±0.000	0.000±0.000	0.018±0.017
	Conditional Success Rate (CSR) - 4	0.000±0.000	NaN	NaN	NaN	NaN	0.000 ± 0.000	NaN	NaN	0.007±0.002
bottom burner → light switch → kettle → microwave	Average Return (AR)	1.46±0.04	1.09±0.05	0.98±0.02	1.09±0.03	1.04±0.07	1.01 ± 0.02	1.06±0.02	1.34±0.05	1.58±0.01
	Conditional Success Rate (CSR) - 2	0.548±0.029	0.151±0.039	0.036±0.016	0.160±0.044	0.111±0.034	0.075 ± 0.031	0.108±0.013	0.411±0.037	0.660±0.034
	Conditional Success Rate (CSR) - 3	0.007±0.012	0.021±0.036	0.000±0.000	0.000±0.000	0.024±0.041	0.000 ± 0.000	0.033±0.047	0.000±0.000	0.032±0.026
	Conditional Success Rate (CSR) - 4	0.000±0.000	0.000±0.000	NaN	NaN	0.000±0.000	NaN	0.000±0.000	NaN	0.000±0.002
bottom burner → microwave → light switch → kettle	Average Return (AR)	1.09±0.08	1.47±0.02	1.46±0.08	1.84±0.04	1.63±0.11	1.63 ± 0.01	1.62±0.05	1.69±0.05	1.81±0.04
	Conditional Success Rate (CSR) - 2	0.126±0.032	0.534±0.034	0.530±0.019	0.875±0.014	0.686±0.054	0.684 ± 0.032	0.728±0.021	0.787±0.017	0.910±0.006
	Conditional Success Rate (CSR) - 3	0.087±0.039	0.020±0.001	0.040±0.003	0.106±0.021	0.037±0.020	0.051 ± 0.029	0.050±0.030	0.060±0.044	0.064±0.036
	Conditional Success Rate (CSR) - 4	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.111±0.192	0.167 ± 0.236	0.000±0.000	0.000±0.000	0.000±0.000
bottom burner → microwave → kettle → light switch	Average Return (AR)	1.18±0.11	1.69±0.03	1.59±0.07	2.19±0.05	1.99±0.17	1.83 ± 0.04	2.07±0.05	2.00±0.07	2.23±0.02
	Conditional Success Rate (CSR) - 2	0.101±0.020	0.563±0.064	0.491±0.049	0.868±0.018	0.728±0.082	0.618 ± 0.026	0.794±0.014	0.756±0.031	0.922±0.007
	Conditional Success Rate (CSR) - 3	0.427±0.063	0.395±0.083	0.435±0.116	0.511±0.042	0.480±0.048	0.492 ± 0.020	0.453±0.034	0.462±0.023	0.430±0.058
	Conditional Success Rate (CSR) - 4	0.058±0.035	0.064±0.077	0.029±0.050	0.088±0.023	0.070±0.012	0.048 ± 0.034	0.063±0.050	0.051±0.063	0.106±0.044
bottom burner → kettle → light switch → microwave	Average Return (AR)	1.09±0.05	1.31±0.07	1.24±0.04	1.23±0.07	1.20±0.11	1.29 ± 0.04	1.19±0.05	0.99±0.05	1.33±0.04
	Conditional Success Rate (CSR) - 2	0.164±0.021	0.388±0.057	0.301±0.054	0.283±0.048	0.253±0.110	0.347 ± 0.029	0.243±0.020	0.072±0.034	0.320±0.053
	Conditional Success Rate (CSR) - 3	0.042±0.037	0.017±0.015	0.039±0.040	0.094±0.057	0.022±0.039	0.069 ± 0.033	0.059±0.043	0.111±0.192	0.137±0.057
	Conditional Success Rate (CSR) - 4	0.000±0.000	0.000±0.000	0.167±0.289	0.000±0.000	0.000±0.000	0.000 ± 0.000	0.000±0.000	0.000±0.000	0.002±0.000
bottom burner → kettle → microwave → light switch	Average Return (AR)	1.09±0.04	1.43±0.06	1.21±0.11	1.27±0.05	1.25±0.08	1.28 ± 0.03	1.25±0.07	1.07±0.03	1.41±0.09
	Conditional Success Rate (CSR) - 2	0.166±0.060	0.418±0.076	0.242±0.072	0.259±0.029	0.229±0.026	0.295 ± 0.058	0.241±0.051	0.108±0.012	0.394±0.030
	Conditional Success Rate (CSR) - 3	0.067±0.115	0.207±0.051	0.195±0.061	0.291±0.097	0.231±0.107	0.224 ± 0.111	0.257±0.056	0.098±0.092	0.299±0.057
	Conditional Success Rate (CSR) - 4	0.167±0.289	0.074±0.128	0.067±0.115	0.000±0.000	0.276±0.240	0.125 ± 0.102	0.167±0.236	0.000±0.000	0.034±0.000
microwave → bottom burner → light switch → kettle	Average Return (AR)	1.84±0.02	1.50±0.06	1.52±0.08	1.41±0.04	1.46±0.11	1.53 ± 0.03	1.68±0.07	1.66±0.06	2.09±0.05
	Conditional Success Rate (CSR) - 2	0.779±0.071	0.610±0.013	0.608±0.063	0.491±0.023	0.517±0.055	0.609 ± 0.020	0.710±0.017	0.663±0.018	0.831±0.034
	Conditional Success Rate (CSR) - 3	0.346±0.062	0.078±0.011	0.117±0.031	0.143±0.054	0.159±0.088	0.114 ± 0.029	0.120±0.029	0.242±0.073	0.467±0.016
	Conditional Success Rate (CSR) - 4	0.013±0.022	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000 ± 0.000	0.056±0.079	0.000±0.000	0.069±0.034
microwave → bottom burner → kettle → light switch	Average Return (AR)	1.83±0.04	1.49±0.10	1.57±0.02	1.45±0.08	1.45±0.11	1.56 ± 0.11	1.57±0.10	1.66±0.12	1.97±0.04
	Conditional Success Rate (CSR) - 2	0.806±0.019	0.540±0.063	0.593±0.039	0.455±0.045	0.553±0.051	0.595 ± 0.007	0.636±0.036	0.666±0.065	0.881±0.012
	Conditional Success Rate (CSR) - 3	0.205±0.025	0.239±0.057	0.203±0.038	0.204±0.047	0.192±0.033	0.190 ± 0.068	0.232±0.026	0.225±0.065	0.264±0.023
	Conditional Success Rate (CSR) - 4	0.024±0.041	0.000±0.000	0.000±0.000	0.042±0.072	0.028±0.048	0.077 ± 0.109	0.042±0.059	0.063±0.057	0.093±0.031
microwave → light switch → bottom burner → kettle	Average Return (AR)	1.90±0.11	1.07±0.10	0.98±0.03	1.18±0.06	1.18±0.06	1.15 ± 0.03	1.19±0.06	0.98±0.04	1.83±0.00
	Conditional Success Rate (CSR) - 2	0.897±0.045	0.144±0.094	0.119±0.026	0.247±0.045	0.270±0.038	0.265 ± 0.008	0.285±0.041	0.081±0.022	0.739±0.009
	Conditional Success Rate (CSR) - 3	0.339±0.064	0.320±0.087	0.072±0.065	0.092±0.051	0.150±0.006	0.156 ± 0.037	0.192±0.037	0.042±0.072	0.314±0.011
	Conditional Success Rate (CSR) - 4	0.000±0.000	0.083±0.144	0.000±0.000	0.000±0.000	0.000±0.000	0.000 ± 0.000	0.000±0.000	0.000±0.000	0.001±0.000
microwave → light switch → kettle → bottom burner	Average Return (AR)	1.71±0.03	1.02±0.03	1.00±0.09	1.13±0.00	1.11±0.06	1.13 ± 0.08	1.19±0.05	1.03±0.02	1.73±0.04
	Conditional Success Rate (CSR) - 2	0.896±0.027	0.113±0.010	0.110±0.074	0.242±0.037	0.215±0.040	0.273 ± 0.033	0.302±0.039	0.117±0.046	0.746±0.054
	Conditional Success Rate (CSR) - 3	0.017±0.019	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.015 ± 0.021	0.000±0.000	0.000±0.000	0.040±0.014
	Conditional Success Rate (CSR) - 4	0.333±0.577	NaN	NaN	NaN	NaN	0.000 ± 0.000	NaN	NaN	0.144±0.284
microwave → kettle → bottom burner → light switch	Average Return (AR)	2.00±0.10	1.21±0.08	1.37±0.04	1.49±0.14	1.37±0.06	1.35 ± 0.09	1.45±0.11	1.52±0.07	2.17±0.00
	Conditional Success Rate (CSR) - 2	0.681±0.002	0.291±0.015	0.399±0.009	0.524±0.063	0.467±0.062	0.407 ± 0.073	0.498±0.043	0.586±0.038	0.693±0.000
	Conditional Success Rate (CSR) - 3	0.723±0.009	0.061±0.040	0.192±0.051	0.195±0.018	0.127±0.014	0.197 ± 0.022	0.208±0.030	0.105±0.033	0.711±0.011
	Conditional Success Rate (CSR) - 4	0.163±0.050	0.000±0.000	0.222±0.192	0.079±0.068	0.111±0.192	0.181 ± 0.027	0.236±0.139	0.342±0.080	0.127±0.016
microwave → kettle → light switch → bottom burner	Average Return (AR)	1.96±0.13	1.20±0.03	1.39±0.03	1.48±0.13	1.42±0.06	1.34 ± 0.01	1.43±0.10	1.50±0.05	2.07±0.13
	Conditional Success Rate (CSR) - 2	0.704±0.042	0.285±0.031	0.416±0.015	0.533±0.083	0.480±0.023	0.410 ± 0.029	0.470±0.070	0.555±0.068	0.746±0.004
	Conditional Success Rate (CSR) - 3	0.578±0.053	0.119±0.080	0.220±0.045	0.168±0.047	0.184±0.014	0.161 ± 0.036	0.194±0.045	0.064±0.007	0.658±0.009
	Conditional Success Rate (CSR) - 4	0.070±0.047	0.000±0.000	0.170±0.101	0.083±0.144	0.042±0.072	0.048 ± 0.067	0.133±0.094	0.000±0.000	0.118±0.051
kettle → bottom burner → light switch → microwave	Average Return (AR)	0.79±0.17	0.46±0.04	0.53±0.05	0.51±0.11	0.51±0.04	0.48 ± 0.06	0.49±0.05	0.50±0.08	0.82±0.04
	Conditional Success Rate (CSR) - 2	0.514±0.005	0.044±0.045	0.089±0.050	0.162±0.071	0.133±0.058	0.108 ± 0.098	0.124±0.028	0.074±0.012	0.630±0.039
	Conditional Success Rate (CSR) - 3	0.338±0.062	0.000±0.000	0.167±0.289	0.042±0.072	0.042±0.072	0.033 ± 0.047	0.000±0.000	0.222±0.192	0.318±0.108
	Conditional Success Rate (CSR) - 4	0.000±0.000	NaN	0.000±0.000	0.000±0.000	0.000±0.000	0.000 ± 0.000	NaN	0.000±0.000	0.098±0.096
kettle → bottom burner → microwave → light switch	Average Return (AR)	0.72±0.06	0.43±0.06	0.46±0.06	0.57±0.14	0.47±0.10	0.51 ± 0.04	0.54±0.05	0.56±0.11	0.82±0.08
	Conditional Success Rate (CSR) - 2	0.602±0.047	0.049±0.027	0.069±0.016	0.187±0.087	0.157±0.025	0.118 ± 0.047	0.284±0.081	0.111±0.037	0.611±0.003
	Conditional Success Rate (CSR) - 3	0.050±0.015	0.000±0.000	0.000±0.000	0.351±0.092	0.095±0.165	0.108 ± 0.082	0.358±0.092	0.458±0.072	0.133±0.055
	Conditional Success Rate (CSR) - 4	0.333±0.577	NaN	NaN	0.056±0.096	0.167±0.289	0.000 ± 0.000	0.167±0.236	0.167±0.289	0.024±0.000
kettle → light switch → bottom burner → microwave	Average Return (AR)	0.68±0.11	0.44±0.10	0.52±0.04	0.47±0.02	0.45±0.04	0.55 ± 0.07	0.56±0.06	0.49±0.04	0.74±0.03
	Conditional Success Rate (CSR) - 2	0.295±0.035	0.076±0.047	0.123±0.030	0.092±0.023	0.140±0.052	0.146 ± 0.002	0.162±0.027	0.068±0.045	0.403±0.082
	Conditional Success Rate (CSR) - 3	0.130±0.119	0.000±0.000	0.048±0.083	0.000±0.000	0.083±0.144	0.042 ± 0.059	0.000±0.000	0.111±0.192	0.166±0.057
	Conditional Success Rate (CSR) - 4	0.000±0.000	NaN	0.000±0.000	NaN	0.000±0.000	0.000 ± 0.000	NaN	0.000±0.000	0.034±0.000
kettle → light switch → microwave → bottom burner	Average Return (AR)	0.54±0.03	0.49±0.04	0.46±0.07	0.44±0.02	0.49±0.04	0.52 ± 0.02	0.51±0.02	0.47±0.08	0.73±0.05
	Conditional Success Rate (CSR) - 2	0.317±0.080	0.121±0.028	0.085±0.033	0.039±0.048	0.117±0.056	0.116 ± 0.039	0.127±0.042	0.022±0.004	0.358±0.021
	Conditional Success Rate (CSR) - 3	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000 ± 0.000	0.000±0.000	0.000±0.000	0.012±0.001
	Conditional Success Rate (CSR) - 4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
kettle → microwave → bottom burner → light switch	Average Return (AR)	0.88±0.09	0.49±0.05	0.50±0.01	0.56±0.08	0.59±0.16	0.56 ± 0.11	0.54±0.04	0.53±0.05	0.90±0.08
	Conditional Success Rate (CSR) - 2	0.545±0.127	0.117±0.062	0.137±0.038	0.268±0.087	0.237±0.084	0.201 ± 0.055	0.166±0.042	0.085±0.047	0.480±0.005
	Conditional Success Rate (CSR) - 3	0.588±0.103	0.532±0.122	0.512±0.232	0.267±0.182	0.282±0.317	0.218 ± 0.050	0.544±0.261	0.222±0.385	0.572±0.184
	Conditional Success Rate (CSR) - 4	0.279±0.029	0.333±0.289	0.111±0.192	0.214±0.258	0.150±0.132	0.000 ± 0.000	0.622±0.328	0.000±0.000	0.358±0.100
kettle → microwave → light switch → bottom burner	Average Return (AR)	0.72±0.10	0.55±0.07	0.53±0.13	0.59±0.08	0.49±0.09	0.59 ± 0.11	0.60±0.10	0.50±0.03	0.85±0.07
	Conditional Success Rate (CSR) - 2	0.514±0.015	0.197±0.127	0.122±0.026	0.255±0.080	0.228±0.110	0.235 ± 0.044	0.258±0.025	0.161±0.084	0.512±0.013
	Conditional Success Rate (CSR) - 3	0.459±0.017	0.145±0.069	0.378±0.039	0.399±0.252	0.257±0.050	0.141 ± 0.033	0.233±0.047	0.116±0.111	0.445±0.000
	Conditional Success Rate (CSR) - 4	0.072±0.065	0.333±0.577	0.000±0.000	0.222±0.255	0.250±0.250	0.111 ± 0.157	0.000±0.000	0.000±0.000	0.057±0.090
Average	Average Return (AR)	1.27	1.02	1.02	1.11	1.07	1.07	1.11	1.10	**1.48**
	Conditional Success Rate (CSR) - 2	0.511	0.264	0.251	0.339	0.314	0.311	0.347	0.317	**0.635**
	Conditional Success Rate (CSR) - 3	0.245	0.120	0.145	0.159	0.131	0.125	0.163	0.141	**0.282**
	Conditional Success Rate (CSR) - 4	0.089	0.074	0.059	0.060	0.080	0.042	**0.114**	0.044	0.071