Efficient Diffusion Planning with Temporal Diffusion

Jiaming Guo[1], Rui Zhang[1]*, Zerun Li[1,2,3], Yunkai Gao[4], Shaohui Peng[5], Siming Lan[4], Xing Hu[1], Zidong Du[1], Xishan Zhang[1,3], Ling Li[5]

[1]SKL of Processors, Institute of Computing Technology, CAS
[2]University of Chinese Academy of Sciences
[3]Cambricon Technologies
[4]Institute of AI for Industries
[5]Intelligent Software Research Center, Institute of Software, CAS
{guojiaming, zhangrui, lizerun23z}@ict.ac.cn, ykgao@iaii.ac.cn, pengshaohui@iscas.ac.cn, smlan@iaii.ac.cn,
{huxing,duzidong,zhangxishan}@ict.ac.cn, liling@iscas.ac.cn

Abstract

Diffusion planning is a promising method for learning high-performance policies from offline data. To avoid the impact of discrepancies between planning and reality on performance, previous works generate new plans at each time step. However, this incurs significant computational overhead and leads to lower decision frequencies, and frequent plan switching may also affect performance. In contrast, humans might create detailed short-term plans and more general, sometimes vague, long-term plans, and adjust them over time. Inspired by this, we propose the Temporal Diffusion Planner (TDP) which improves decision efficiency by distributing the denoising steps across the time dimension. TDP begins by generating an initial plan that becomes progressively more vague over time. At each subsequent time step, rather than generating an entirely new plan, TDP updates the previous one with a small number of denoising steps. This reduces the average number of denoising steps, improving decision efficiency. Additionally, we introduce an automated replanning mechanism to prevent significant deviations between the plan and reality. Experiments on D4RL show that, compared to previous works that generate new plans every time step, TDP improves the decision-making frequency by 11-24.8 times while achieving higher or comparable performance.

1 Introduction

Learning effective policies from previously collected suboptimal data is appealing for reinforcement learning as it circumvents the need for risky or costly environmental interactions (Kumar et al. 2020; Fujimoto, Meger, and Precup 2019; Kumar et al. 2019; Kostrikov, Nair, and Levine 2022). Recently, diffusion planning has emerged as a promising approach for this offline reinforcement learning problem (Janner et al. 2022; Ajay et al. 2023; Du et al. 2023; Ni et al. 2023), harnessing the powerful generative power of diffusion models (Ho, Jain, and Abbeel 2020) to learn from data and predict future trajectories for effective planning. This method enables decision-making of current action based on long-term plans, thereby mitigating the short-sightedness associated with single-step decision-making.

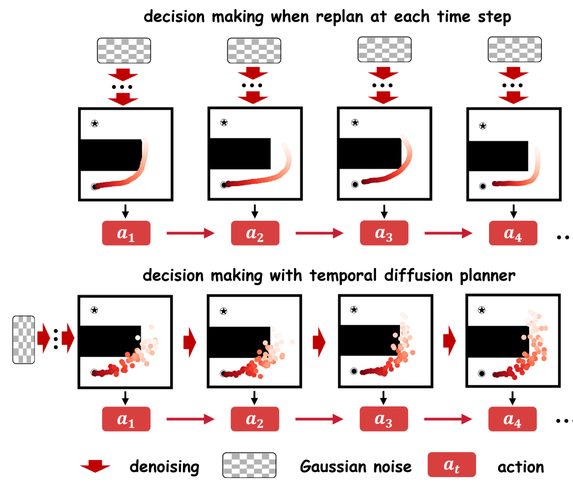

Figure 1: Comparison of temporal diffusion planner and diffusion planning methods which replan at each time step. The visualization of points in the figure illustrates the x-y coordinates of the planned trajectories.

However, there is often a discrepancy between reality and the plan, and this discrepancy tends to accumulate over time. As a result, rigidly following the original plan without adjustments often leads to poor performance. To mitigate this, previous methods (Janner et al. 2022; Ajay et al. 2023) typically execute only the first action of the plan and replan at each time step. However, in diffusion planning, generating a new plan requires multiple denoising steps, involving several forward passes through a large neural network. This leads to expensive computational overhead and low decision frequency. Additionally, frequent plan switching may result in discontinuous actions, which can hinder the agent from achieving a certain goal and degrade the overall performance of the algorithm. Therefore, we explore the possibility of reusing the plan to improve efficiency while reducing the accumulation of planning errors.

How do humans make plans and act according to them? Humans tend to create detailed short-term plans and more general, or even vague, long-term plans. For example, when planning a long journey, a person might detail the first few

*Corresponding Author.

days' route and stops, while leaving the later part more flexible based on time and weather, focusing only on general destinations. Additionally, people continuously adjust their plans based on the current situation. We can draw inspiration from this to improve the diffusion planning process.

In this paper, we propose the Temporal Diffusion Planner (TDP) for efficient diffusion planning. We present a novel temporal diffusion approach that adjusts the number of diffusion steps along the time horizon. When generating a new plan, TDP creates detailed short-term and vague long-term plans, thereby reducing the cost of generating fully detailed long-term plans. With the temporal diffusion, we can refine the time-evolving plan at each time step by applying a small number of denoising steps to the previous plan. To be specific, at each time step, the execution consumes the beginning of the previous plan while TDP extends the plan to a defined horizon and denoises it based on the current state until the first action is fully detailed. This process corrects the plan based on the current state and allows the plan to be reused and updated over time, as illustrated in Figure 1. To further reduce the accumulation of planning errors, we introduce a mechanism to determine whether to replan based on the degree of discrepancy between the plan and reality. TDP maximizes the reuse of previously generated plans to reduce computational load, allowing for efficient diffusion planning, while the plan refinement and automated replanning ensure consistent performance.

In our experiments, we validate the proposed Temporal Diffusion Planner using the D4RL (Fu et al. 2020) benchmark. The results demonstrate that the Temporal Diffusion Planner achieves 11 to 24.8 times greater decision-making efficiency compared to methods that replan at each step while maintaining comparable or improved performance.

2 Related Work

Offline Reinforcement Learning. Offline reinforcement learning (RL) focuses on learning policies from static datasets. A series of works have been developed based on off-policy RL methods. These approaches have to mitigate estimation errors for out-of-distribution actions arising from distributional dissimilarities by constraining the learned policy to remain closer to the behavior policy (Kumar et al. 2019; Wu, Tucker, and Nachum 2019; Peng et al. 2019; Wu et al. 2022), penalize learned value functions to produce more conservative estimates (Nachum et al. 2019; Fujimoto, Meger, and Precup 2019; Kumar et al. 2020; Lyu et al. 2022), or performing a single step of evaluation followed by a single policy improvement step rather than iterative policy evaluation (Brandfonbrener et al. 2021; Kostrikov, Nair, and Levine 2022). Some works categorized as model-based offline RL involve a model for data generation (Kidambi et al. 2020; Argenson and Dulac-Arnold 2020; Yu et al. 2020; Yang et al. 2021) or trajectory search (Janner, Li, and Levine 2021). There are also return conditioned approaches that learn policies via return conditioned behavioral cloning from the dataset (Chen et al. 2021; Emmons et al. 2022). In this work, we study using the diffusion model for trajectory generation and planning in decision-making tasks.

Diffusion Models for Decision Making. The denoising diffusion probabilistic model (Diffusion) (Ho, Jain, and Abbeel 2020) frames the generation process as a Markov decision process (MDP) tied to noise, comprising a forward process that adds noise and a reverse process that denoises to recover the original distribution. Leveraging its strong capacity to model arbitrary distributions, Diffusion models have gained increasing traction in decision-making tasks. Several works model policies with diffusion models to capture multimodal action distributions (Wang, Hunt, and Zhou 2022; Kang et al. 2023; Chi et al. 2023). Others utilize diffusion models for reinforcement learning data generation (He et al. 2023; Lu et al. 2023). Recent research also explores diffusion models for trajectory generation and planning, a category our work falls into. Diffuser (Janner et al. 2022) employs a diffusion model with classifier-guided sampling to generate planning trajectories, while DD (Ajay et al. 2023) designs a return-conditioned diffusion model for trajectory generation and uses an inverse dynamic model for action prediction. Building on these pioneering methods, diffusion-based planning has been applied to diverse domains (Ni et al. 2023; Chen et al. 2024a; Zhu et al. 2023; Lee, Kim, and Choi 2023) to achieve higher performance. Several studies (Li et al. 2023; Chen et al. 2024b; Dong et al. 2024) turn to employ multiple diffusion models to form a hierarchical frameworks and make decisions at multiple level, while we focus on single-level planning which is the fundamental component and study improving the denoising manner to achieve efficient diffusion planning. With the same purpose of us, prior methods add noise to previous plans as a warm start for subsequent planning iterations to reduce the denoising steps (Janner et al. 2022; Feng et al. 2024). However, they trade sample quality to achieve acceleration and the resulting speedup remains limited under constraints on performance degradation, In contrast, our approach significantly boosts efficiency while preserving performance.

3 Preliminaries

Problem Setup. We consider the sequential decision-making problem as a discounted Markov Decision Process, defined as the 6-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, r, \gamma \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^m$ and $\mathcal{A} \subseteq \mathbb{R}^n$ are the state and action spaces, $\mathcal{P}$ is the transition distribution, ρ_0 the initial state distribution, r the reward function, and $\gamma \in (0, 1)$ the discount factor. The agent acts with a stochastic policy $\pi : \mathcal{S} \to \mathcal{A}$ and generates a trajectory sequence $\tau = (s_0, a_0, \dots)$ following $s_0 \sim \rho_0, a_t \sim \pi(a_t|s_t), s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. The standard objective in RL is to optimize the expected return $\eta(\pi) = \mathbb{E}_\tau[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. We define the value function as $V(s_t) = \mathbb{E}_{a_t, s_{t+1}, \dots}[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})|s_t]$. We define the expected return achieved under the optimal policy as $Q^*(s, a) = \mathbb{E}_{\tau \sim p_{\pi^*}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a]$.

In offline RL, agents focus on learning from a fixed dataset $D = \{(s, a, r, s')\}$ pre-collected using an unknown behavior policy $\pi_\beta(a|s)$. To solving this problem, diffusion planning methods employ a diffusion model to generate a trajectory $\tau = \{s_t, a_t, s_{t+1}, a_{t+1}, \dots, s_{t+H-1}, a_{t+H-1}\}$ or $\tau = \{s_t, s_{t+1}, \dots, s_{t+H-1}\}$, where H is the planning horizon. The agent will interact with the environment according

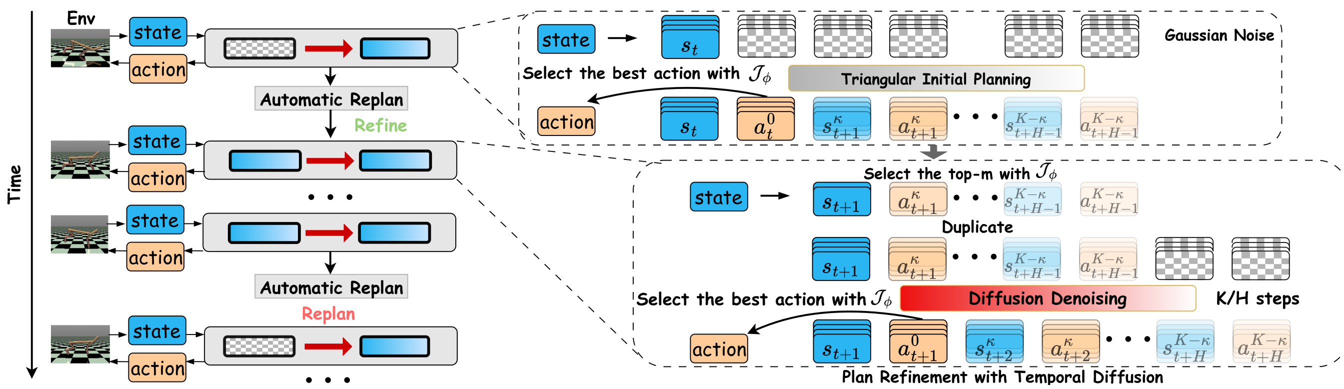

Figure 2: Overview of the Temporal Diffusion Planner. TDP starts with Triangular Initial Planning, creating detailed short-term and vague long-term plans. It then refines previous time-evolving plan at each step via temporal diffusion, applying very small number of denoising steps until automatic-replanning is triggered.

to the trajectory, e.g. directly using the generated first action a_t (Janner et al. 2022) or predicting the action with an inverse dynamic model $p(a_t|s_t, s_{t+1})$ (Ajay et al. 2023). The trajectory should maximize the object value $\mathcal{J}(\tau)$, typically defined as expected return in offline RL.

Diffusion Probabilistic Model. Diffusion probabilistic models (Ho, Jain, and Abbeel 2020) are a class of latent variable generative models that learn the data distribution $q(\boldsymbol{x})$ from a dataset. Diffusion models contain a forward noising process for data generation and a trainable reverse denoising process. In the forward process, Gaussian noise is added to origin data x_0 over K steps, following a variance schedule $\beta_{1:K}, 0 < \beta_1 < \beta_2 < \cdots < \beta_K < 1$, generating a sequence of $\boldsymbol{x}_{1:K}$ until it nearly becomes pure Gaussian noise. Then the relation between $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ is $\boldsymbol{x}_k = \sqrt{1-\beta_k}\boldsymbol{x}_{k-1} + \sqrt{\beta_k} z_k$, where $z_k \sim \mathcal{N}(0, I)$. The relation between $\boldsymbol{x}_0$ and $\boldsymbol{x}_k$ is $\boldsymbol{x}_k = \sqrt{\bar{\alpha}_k}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_k} z_k$, where $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \alpha_1\alpha_2 \ldots \alpha_k$. The reverse denoising process is constructed as $p(\boldsymbol{x}_{0:K}) \sim \mathcal{N}(\boldsymbol{x}_k; 0, I)\prod_{k=1}^{K} p(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k)$. Through the Bayesian equation,

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k) := \mathcal{N}(\boldsymbol{x}_{k-1}|\mu_\theta(\boldsymbol{x}_k, k); \Sigma_k)$$

$$\mu_\theta(\boldsymbol{x}_k, k) = \frac{1}{\sqrt{\alpha_k}}(\boldsymbol{x}_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\epsilon}_\theta), \Sigma_k = \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k.$$

Thus, the reverse denoising process p is parameterized through the noise model ϵ_θ, and the optimization object is to maximize the evidence lower bound, the corresponding loss is $L_\theta = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_k}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, k)]\|^2$.

Conditional Diffusion Sampling. To solve reinforcement learning problems with diffusion planning, the generated trajectory should maximize the object function $\mathcal{J}(\tau)$, which is a conditional diffusion sampling problem. There are two main approaches for conditional diffusion sampling: classifier-guided (Nichol and Dhariwal 2021) and classifier-free (Ho and Salimans 2022). Following Diffuser, we use the classifier-guided method. This method first trains a separate model $\mathcal{J}_\phi$ as the objective function. Then it uses the gradients of $\mathcal{J}_\phi$ to guide the trajectory sampling procedure by modifying the means μ of the reverse process: $p_\theta(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k) := \mathcal{N}(\boldsymbol{x}_{k-1}|\mu + \alpha\Sigma_k\nabla_\mu\mathcal{J}(\mu); \Sigma_k)$, where μ refers to $\mu_\theta(\boldsymbol{x}_k, k)$ and α is the scale.

4 Method

In this section, we first describe the proposed temporal diffusion model, along with its training process and architecture. We then introduce the Temporal Diffusion Planner (TDP), which consists of three main components: 1) the process for generating the initial plan in a triangular manner, 2) the process for refining the plan over time using temporal diffusion, and 3) the mechanism for automatic replanning to prevent performance degradation.

4.1 Temporal Diffusion

The standard diffusion planning paradigm generates a clear plan of horizon H at each decision-making step, or in other words, predicts all timesteps of a plan simultaneously. However, as the poem line goes, "*The best laid plans of mice and men often go awry*", executing a complete plan over time is difficult due to cumulative forecasting errors or unexpected environment dynamics. Thus, the further forward the plan, the less important its details are to the current decision-making. When humans plan, they tend to make detailed plans for the near future and more general, macro plans for the long term. Inspired by this, we adopt a similar strategy when using diffusion to generate plans.

We propose the temporal diffusion which uses different diffusion steps at different time steps of the plan. Specifically, for a trajectory of length H: $\tau = (s_t, a_t, s_{t+1}, a_{t+1}, \ldots, s_{t+H-1}, a_{t+H-1})$, we define the noise array of the trajectory as $\boldsymbol{x}_k(\tau)$, where k is the diffusion step for the start of the noise array. For each step in time, the corresponding diffusion step increases by κ, where $\kappa \in \mathbb{N}_+$ and $\kappa \leq K/H$. In other words, during the denoising process, as time progresses, the number of denoising steps decreases by κ at each step. In this way, the noise array of the plan gets more and more ambiguous over time. We can formulate the noise array $\boldsymbol{x}_k(\tau)$ as: $\boldsymbol{x}_k(\tau) = \{\boldsymbol{x}_{k+i\kappa}(s_{t+i}, a_{t+i})\}_{i=0}^{i=H-1}$, where $\boldsymbol{x}_k(s, a) = (\sqrt{\bar{\alpha}_k}s + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, \sqrt{\bar{\alpha}_k}a + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon})$. For brevity, we use s_t^k and a_t^k to represent $\sqrt{\bar{\alpha}_k}s_t + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}$ and $\sqrt{\bar{\alpha}_k}a_t + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}$ respectively in the following. Note that we cut off the noise array when the diffusion step $k + i\kappa$ is greater than or equal to the maximum number of diffusion

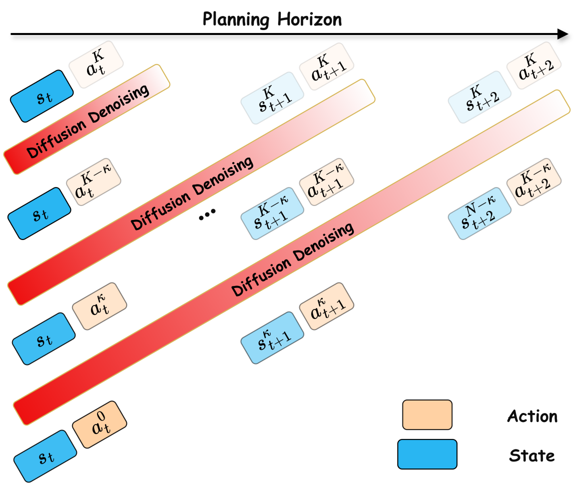

Figure 3: Triangular Initial Planning when the planning horizon is set as 3. s^K_{t+i}, a^K_{t+i} are sampled from Gaussian noise.

steps K. Thus with different diffusion steps k, the length of the noise array $\boldsymbol{x}_k(\tau)$ may be different.

For conditional sampling, we train a separate model $\mathcal{J}_\phi(\boldsymbol{x}_k(\tau))$ to predict the value function $V(\tau)$ conditioned on the original trajectory with the noise array as input. Then the reverse denoising process transitions can be formulated as:

$$p_\theta(\boldsymbol{x}_{k-1}(\tau)|\boldsymbol{x}_k(\tau)) := \mathcal{N}(\boldsymbol{x}_{k-1}(\tau)|\mu + \alpha\Sigma_k \nabla_\mu \mathcal{J}(\mu); \Sigma_k), \tag{1}$$

where μ refers to $\mu_\theta(\boldsymbol{x}_k(\tau), k)$ and α is the scale. When the length of $\boldsymbol{x}_{k-1}(\tau)$ is larger than $\boldsymbol{x}_k(\tau)$, we can pad $\boldsymbol{x}_k(\tau)$ with noise array sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Besides, we always replace the first state of the noise array with the current state s_t during training and inference. In this way, the diffusion model denoises the array conditioned on the current state.

Training Process. For the Temporal Diffusion, we train the reverse denoising process p_θ, parameterized through the noise model ϵ_θ as traditional diffusion models (Ho, Jain, and Abbeel 2020; Janner et al. 2022; Ajay et al. 2023). The key difference lies in the diffusion process, where the diffusion steps differ across time steps. As a result, we adopt a temporal noise addition strategy in the forward process of diffusion. Specifically, for a trajectory $\tau = (s_t, a_t, s_{t+1}, a_{t+1}, \ldots, s_{t+H-1}, a_{t+H-1})$, we sample noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and the diffusion step $k \sim \mathcal{U}\{1, \ldots, K\}$ to obtain a noise array of the trajectory $\boldsymbol{x}_k(\tau)$. Then we train the noise model with the following loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \in D, k}[\|\epsilon - \epsilon_\theta(\boldsymbol{x}_k(\tau), k)\|^2] \tag{2}$$

Architecture. We parameterize ϵ_θ with a temporal U-Net (Janner et al. 2022), comprising repeated convolutional residual blocks. It resembles image-based U-Nets but replaces 2D spatial convolutions with 1D temporal ones. As a fully convolutional model, the prediction horizon depends on input dimension, which aligns with the requirement of temporal diffusion for handling variable data lengths.

4.2 Triangular Initial Planning

We now explain how to generate a plan using the temporal diffusion model. In practice, we set κ as K/H and assume H can divide K exactly. Since the diffusion steps vary across time steps, the process of generating a plan from scratch resembles a triangular shape, with the slope of the hypotenuse denoted as κ. Thus, we refer to this process as Triangular Initial Planning. We illustrate the process when the planning horizon is set as 3 in Figure 3. We start by sampling the first state-action pair $\{s^K_t, a^K_t\}$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and replace s^K_t with the current state s_t. Then we use the diffusion model for denoising κ steps according to Equation 1 and also replace the first state to obtain $(s_t, a^{K-\kappa}_t)$. Next, we sample the second state-action pair $\{s^K_{t+1}, a^K_{t+1}\}$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and append to the previous state-action pair: $\{s_t, a^{K-\kappa}_t, s^K_{t+1}, a^K_{t+1}\}$. As the diffusion model is able to accept variable length inputs, we use it to denoise the new sequence κ steps: $\{s^{K-2\kappa}_t, a^{K-2\kappa}_t, s^{K-\kappa}_{t+1}, a^{K-\kappa}_{t+1}\}$. And so on, we repeat the sampling and denoising process to finally get the initial plan: $\{s_t, a^0_t, s^\kappa_{t+1}, a^\kappa_{t+1}, \cdots, s^{K-\kappa}_{t+H-1}, a^{K-\kappa}_{t+H-1}\}$ which can also be represented as $\boldsymbol{x}_0(\tau)$. In this way, we get plans that blur over time and at the same time reduce the amount of computation by about half. In practice, we use this process simultaneously to obtain M candidate plans $\{\boldsymbol{x}_0(\tau)\}_M$. We predict the cumulative rewards of the candidate plans with $\mathcal{J}_\phi$ and select the best plan to extract the action.

4.3 Plan Refinement with Temporal Diffusion

In the proposed TDP, we do not replan at each step as in the previous works but rather use the temporal diffusion model to refine the initial plan at each time step according to the new state. This will greatly reduce the average diffusion steps and improve efficiency. We illustrate the refinement process in Figure 2 and provide the details below.

In the initial plan, only the action of the first step is totally *clear* as it is obtained from the full K denoising steps. After we execute the first action a^0_t and remove s_t, a^0_t, the remaining candidate plans are $\{s^\kappa_{t+1}, a^\kappa_{t+1}, \ldots, s^{K-\kappa}_{t+H-1}, a^{K-\kappa}_{t+H-1}\}_M$, which is exactly $\{\boldsymbol{x}_\kappa(\tau_{t+1})\}_M$, where τ_{t+1} means the trajectory starts from s_{t+1}. Thus, we can naturally use the temporal diffusion model to refine the plan by denoise the remaining candidate plans κ steps. In practice, we do this with a candidate selection process. We select the top-m plans from these candidate plans based on the objective function $\mathcal{J}_\phi$ to eliminate inferior plans while maintaining diversity. We replace s^κ_{t+1} with the current state s_{t+1} and duplicate the plans to obtain M candidate plans. Then we sample M state-action pair $\{s^K_{t+H}, a^K_{t+H}\}$ from Gaussian Noise $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and append to the candidate plans. Finally, we diffuse these candidate plans κ steps and obtain the refined plans: $\{\boldsymbol{x}_0(\tau_{t+1})\}_M = \{s_{t+1}, a^0_{t+1}, s^\kappa_{t+2}, a^\kappa_{t+2}, \cdots, s^{K-\kappa}_{t+H}, a^{K-\kappa}_{t+H}\}_M$. As κ is a small number, e.g. usually set as 1 in our experiments, the process is very efficient. Same as the triangular initial planning process, we also execute the first action of the plan with the highest objective function $\mathcal{J}_\phi$. For each new time step, we can repeat this process to refine the plan from the previous step to obtain the new plan and the current action. Note that the horizon of the initial plan does not limit the amount of refinement, because each time step we append a new state-action pair at the end of the remaining plans thus obtaining a full plan of length H.

Dataset	Environment	BC	CQL	IQL	DT	TT	MOReL	Diffuser	DD	TDP	TDP_{Inv}
Medium-Expert	HalfCheetah	55.2	91.6	86.7	86.8	**95.0**	53.3	88.9	**90.6**	**91.6** ± 0.26	**93.9** ± 0.2
Medium-Expert	Hopper	52.5	105.4	91.5	**107.6**	**110.0**	**108.7**	103.3	**111.8**	**112.4** ± 0.17	**110.1** ± 0.92
Medium-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	95.6	**106.9**	**108.8**	**108.2** ± 0.19	**108.4** ± 0.07
Medium	HalfCheetah	42.6	44.0	**47.4**	42.6	**46.9**	42.1	42.8	**49.1**	43.9 ± 0.24	43.6 ± 0.19
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	**95.4**	74.3	79.3	74.0 ± 3.41	**96.3** ± 1.17
Medium	Walker2d	75.3	72.5	78.3	74.0	**79.0**	77.8	**79.6**	**82.5**	**78.8** ± 1.12	**80.1** ± 1.22
Medium-Replay	HalfCheetah	36.6	**45.5**	**44.2**	36.6	41.9	40.2	37.7	39.3	39.2 ± 0.32	39.7 ± 0.28
Medium-Replay	Hopper	18.1	**95.0**	94.7	82.7	91.5	93.6	93.6	**100**	94.9 ± 0.64	**96.1** ± 1.50
Medium-Replay	Walker2d	26.0	77.2	73.9	66.6	82.6	49.8	70.6	75	71.2 ± 2.83	70.5 ± 6.15
Average		51.9	77.6	77.0	74.7	78.9	72.9	77.5	**81.8**	**79.4**	**82.1**
Mixed	Kitchen	51.5	52.4	51	-	-	-	50.0	**65**	57.5	57.5
Partial	Kitchen	38	50.1	46.3	-	-	-	56.2	57	**65.0**	60.0
Average		44.8	51.2	48.7	-	-	-	53.1	**61.0**	**61.3**	**58.8**

Table 1: **Offline Reinforcement Learning Performance.** The performance of TDP, TDP_{Inv} and a variety of prior algorithms on the D4RL tasks (Fu et al. 2020). Results for TDP and TDP_{Inv} correspond to the mean and standard error over 5 random seeds. Following Diffuser (Janner et al. 2022), we emphasize in bold scores within 5 percent of the maximum per task ($\geq 0.95 \cdot max$)

Environment	Metric	Diffuser	DD	TDP	TDP_{Inv}
Mujoco	Runtime (s)	0.734	1.65	**0.086**	0.125
	Frequency (Hz)	1.4	0.6	**11.6**	8.0
Kitchen	Runtime (s)	0.731	1.57	**0.052**	0.063
	Frequency (Hz)	1.4	0.64	**19.2**	15.9
Average		0.733	1.61	**0.069**	0.094
		1.4	0.62	**15.4**	12.0

Table 2: **Efficiency comparison of diffusion planning methods.** The runtime cost per step of decision-making and the decision frequency.

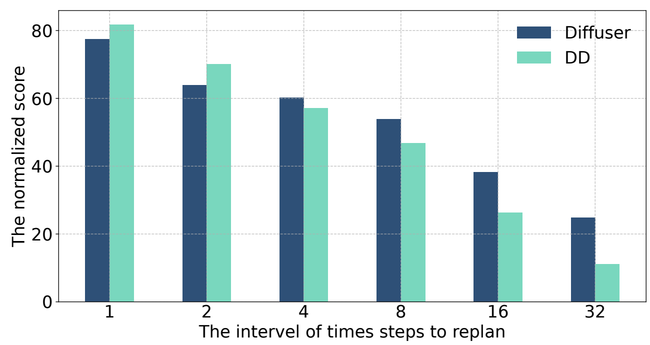

Figure 4: The average scores of previous diffusion planning methods on Mujoco when replan at different intervals.

4.4 Automatic Replanning

Although plan refinement can adjust the plan based on the new state, the plan may still deviate more and more over time, leading to the refinement process failing to guarantee a good plan. To address this issue, we propose to replan automatically based on the plan errors and design two criteria:

State-based criterion. We can decide whether to replan based on how far the current state obtained from the environment differs from the current state in the previous plan. In the plan obtained in the previous time step, the current state s_{t+1}^{κ} has not yet gone through the complete diffusion process. According to the diffusion forward noising process, we can calculate the probability distribution of $s_{t+1}^{*\kappa}$ after adding noise over κ steps to the ground truth s_{t+1}^{*}: $p(s_{t+1}^{*\kappa}|s_{t+1}^{*})$. In this way, we can use the probability of s_{t+1}^{κ} in this distribution as the criterion of replanning:

$$\mathcal{C}_{state} = -logp(s_{t+1}^{\kappa}|s_{t+1}^{*}).$$

Value-based criterion. When we generate the plan or refine the plan, we estimate the objective function $\mathcal{J}_\phi$ for the plan at the same time. The objective function $\mathcal{J}_\phi(\boldsymbol{x}_0(\tau))$ estimates the value function conditioned on the plan: $V(\tau)$. We use the TD-loss for values for new plan τ_{t+1} and the previous plan τ as the criterion of replanning:

$$\mathcal{C}_{value} = \mathcal{J}_\phi(\boldsymbol{x}_0(\tau)) - r - \gamma \mathcal{J}_\phi(\boldsymbol{x}_0(\tau_{t+1})).$$

At every time step, we compute the criterion and if the criterion is greater than a threshold, we use the Triangular Initial Planning process to get the plan. In practice, we use the value-based criterion and we show how the design choice of the criterion affects the performance in Ablation Study. We provide the algorithm of the TDP in the Appendix.

4.5 Acting with Inverse Dynamic Model

In the previous sections, TDP leveraged the diffusion model to generate complete trajectories that included both states and actions for planning. Alternatively, the diffusion model can be used to generate trajectories consisting solely of states. Actions can then be inferred from consecutive states using an inverse dynamics model (Pathak et al. 2018; Ajay et al. 2023): $a_t = \mathcal{I}_\psi(s_t, s_{t+1})$. We refer to this variant of TDP as TDP_{Inv}. To obtain actions through the inverse dynamics model, the next state s_{t+1} in the generated plan needs to undergo a complete denoising process. We adjust the Triangular Initial Planning and Plan Refinement processes such that the number of denoising steps is reduced by κ every two timesteps. Thus, the candidate plans generated at each timestep can be represented

Dataset	TDP_{All}	TDP_{Best}	TDP
Hopper-Medium-Expert	39.6	**111.5**	**112.4**
Hopper-Medium	56.4	**73.3**	**74.0**
Hopper-Medium-Replay	86.2	**94.9**	**94.9**
Average	60.73	**93.2**	**93.7**

Table 3: Ablation results for candidate selection. TDP_{All} and TDP_{Best} means using all candidate plans or only using the best plan for planning refinement respectively.

as $\{s_t, s_{t+1}, s^{\kappa}_{t+2}, s^{\kappa}_{t+3} \cdots, s^{K-\kappa}_{t+H-2}, a^{K-\kappa}_{t+H-1}\}_M$. During training, TDP_{Inv} introduces an additional loss function to train the inverse dynamics model: $\mathcal{L}(\psi) = \mathbb{E}_{(s_t,a,s_{t+1})\in D}[\|a_{t+1} - \mathcal{I}_\psi(s_t, s_{t+1})\|^2]$.

5 Experiment

In this section, we explore the efficacy of the proposed TDP on various decision-making tasks. In particular, we aim to answer the following research questions: **(1)** Compared with diffusion planning methods which generate new plans at each step, to what extent can TDP improve decision-making efficiency? **(2)** How is the performance of TDP on offline reinforcement learning tasks? **(3)** What impact does the design choice of TDP have on performance?

5.1 Experimental Setup

Benchmarks. We evaluate the capacity of our method to recover an effective policy from previously collected data on the D4RL offline RL benchmark suite (Fu et al. 2020) which contains tasks of various domains. We use locomotion tasks of Gym-MuJoCo (Brockman et al. 2016) which contains heterogeneous data of varying quality. We also use FrankaKitchen (Gupta et al. 2019) tasks which require long-term credit assignment ability.

Baselines. We compare our method with the basic imitation learning algorithm BC and existing offline RL methods, including model-free approaches like CQL (Kumar et al. 2020) and IQL (Kostrikov, Nair, and Levine 2022), return conditioning approaches like Decision Transformer (DT) (Chen et al. 2021) and model-based approaches such as trajectory transformer (TT) (Janner, Li, and Levine 2021) and MoReL (Kidambi et al. 2020). For diffusion planning methods, we compare with Diffuser (Janner et al. 2022) and Decision Diffuser (DD) (Ajay et al. 2023). We also compare TDP with methods that enhance the efficiency of the Diffuser by reducing the number of diffusion steps, including Warm-Start (WS) (Janner et al. 2022), Tree-Warm-Start (TWS) (Feng et al. 2024) and DDIM (Song, Meng, and Ermon 2021). We provide more details in the Appendix.

5.2 Results

Efficiency. To evaluate the efficiency of the proposed Temporal Diffusion Planner (TDP), we measured the wall-clock runtime for a single decision-making step (one action inference) and calculated the corresponding decision frequency. The results are summarized in Table 2. For Gym-Mujoco tasks, TDP and TDP_{Inv} reduced runtime costs to 11.7% and 17% of Diffuser, and to 5.2% and 7.6% of Decision Diffuser (DD), respectively. Similarly, for FrankaKitchen tasks, TDP and TDP_{Inv} reduced runtime costs to 7.1% and 8.6% of Diffuser, and to 3.3% and 4.0% of DD, respectively. On average, TDP increased decision frequency by 11 times and 24.8 times compared to Diffuser and DD, respectively, while TDP_{Inv} achieved improvements of 8.6 times and 19.4 times. These results highlight that TDP significantly enhances decision-making efficiency. This improvement is substantially due to the reduction in the average number of denoising steps required per decision-making step. Through temporal diffusion, we can mostly use plan refinement to make decisions instead of generating plans from scratch, reducing the diffusion steps, and thereby improve the efficiency. We record the average replanning interval of TDP and TDP_{Inv}, which are 37.3 and 31.5 for Mujoco, 62.3 and 302.3 for Kitchen. Besides, as Figure 4 shows, Diffuser and DD experience significant performance degradation without replanning at each step, so they cannot improve decision efficiency by reusing plans. For methods which enhance the efficiency of the Diffuser by reducing the number of diffusion steps, we evaluate them across varying denoising steps in the hopper environment. Table 4 reports the minimal runtime of each method under the constraint that performance remains at or above 95% of the Diffuser, denoting with a dash when performance preservation is unattainable with any tested denoising steps. The results demonstrate TDP's significant efficiency advantages over Warm-Start (WS), Tree-Warm-Start (TWS) and DDIM.

Dataset	WS	TWS	DDIM	TDP
Medium-Expert	0.156	0.214	0.379	**0.049**
Medium	0.267	0.309	-	**0.11**
Medium-Replay	0.28	0.298	0.185	**0.067**

Table 4: Comparison in terms of runtime cost (s) with methods that enhance the efficiency of Diffuser by reducing the number of diffusion steps.

Performance. We evaluated TDP across different offline reinforcement learning tasks from D4RL benchmark to verify whether it can maintain performance when improving efficiency. We report the normalized scores of different algorithms in Table 1. We find that the proposed TDP achieves higher or comparable performance compared to baselines. For diffusion planning methods, Diffuser is the most important baseline as TDP uses the same diffusion model architecture and the same approach for conditional sampling. The superior performance of TDP compared to Diffuser demonstrates that the proposed temporal diffusion approach not only improves decision efficiency but also enhances performance. Compared to Decision Diffuser, which employs a different conditional sampling method and requires longer decision times, TDP achieves comparable performance. Moreover, our variant TDP_{Inv} , which predicts actions using an inverse dynamics model, slightly outperforms Decision Diffuser in average scores on D4RL locomotion tasks. On the D4RL Kitchen tasks, which require long-term credit assignment, TDP significantly outperforms Diffuser.

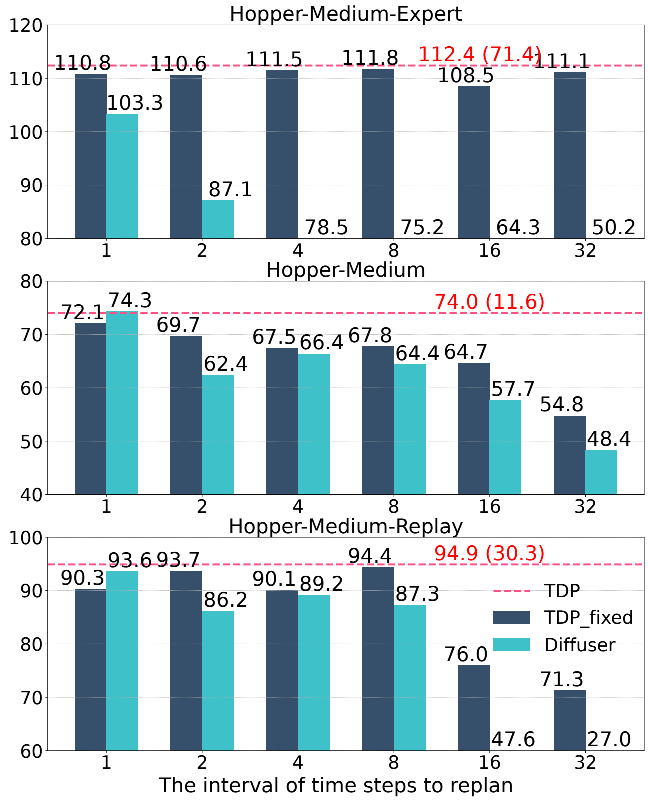

Figure 5: The normalized scores of TDP for replanning at different fixed intervals. We present the normalized scores when automatic replan with the dashed lines.

This may indicate that refining and reusing previous plans to ensure action continuity is beneficial for long-horizon tasks.

5.3 Ablation Study

Candidate Selection during Refinement. In TDP, plan refinement selects the top m m plans from the candidate plans generated in the previous step based on the objective function $\mathcal{J}_\phi(\tau)$ for further refinement. Across all environments, we set the total number of candidate plans as 64 and m as 8. To analyze the impact of candidate selection on performance, we conduct experiments in the Hopper environment and report the results in Table 3. We find that retaining all candidate plans led to a significant performance drop while keeping only the best plan resulted in only a slight performance decrease. This indicates that selecting only the better plans from the candidates is essential, but retaining multiple plans to ensure diversity is not critical.

Replan at Fixed intervals. We verify the effect of the proposed automatic replanning mechanism by ablating it and generating plans from scratch at a fixed interval: $[1, 2, 4, 8, 16, 32]$. We report the normalized scores of Hopper environment in Figure 5. For all data quality, TDP with automated replanning mechanism received higher normalized scores, indicating that our automated scheduling mechanism can effectively determine when it is better to replan. Besides, for Medium-Expert data, replanning at different intervals has little effect on performance, while when using Medium and Medium-replay data, performance decreases significantly when the interval is too long. This shows that plan refinement cannot completely avoid failure when data quality is poor. We also observed that replanning each step did not always result in a better score, and when using Medium-replay data, fewer scheduled intervals resulted in a drop in performance, suggesting that the discontinuity of action caused by frequent plan switching harms performance. We also compare the performance of the fixed-interval TDP with that of Diffuser. The main difference between them is whether using temporal diffusion. As the replanning interval increases, the performance of the Diffuser degrades much faster than TDP. This indicates that using temporal diffusion is critical for reusing the plan.

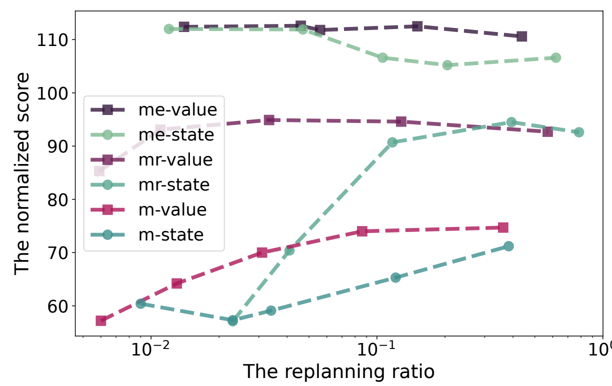

Figure 6: Ablation results for criteria of automatic replan.

Replan with State-based criterion. We compare the effect of automatic replanning using state-based or value-based criterion through experiments in the Hopper environment. We use a series of thresholds for both criteria and record the corresponding performance and the ratio of replanning in all time steps. As Figure 6 shows, The performance of the value-based criterion is above the state-based criterion for all three quality of data. In addition, when using the value-based criterion, the performance gap corresponding to different replanning ratios is smaller. Therefore, we choose the value-based criterion for automatic replanning.

6 Conclusion

In this paper, we introduce the Temporal Diffusion Planner which improves the efficiency of diffusion planning. TDP uses a novel temporal diffusion approach to generate and improve plans over time and prevents performance degradation through automatic replanning. TDP achieves higher decision efficiency by reducing the average number of denoising steps required per step. Experiments on D4RL show that, compared to previous works which generate new plans every time step, TDP achieves 11-24.8 times faster than previous mainstream methods and has higher or comparable performance. We focus on enhancing the basic diffusion planning while we will extent TDP to complex framework like multi-task or hierarchical framework in the future work and hope TDP can inspire other works of diffusion planning.

Acknowledgements

This work is partially supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grants No.XDB0660200, XDB0660201, XDB0660202), the NSF of China (Grants No.62341411, 62222214), CAS Project for Young Scientists in Basic Research (YSBR-029) and Youth Innovation Promotion Association CAS.

References

Ajay, A.; Du, Y.; Gupta, A.; Tenenbaum, J. B.; Jaakkola, T. S.; and Agrawal, P. 2023. Is Conditional Generative Modeling all you need for Decision Making? In *ICLR*. OpenReview.net.

Argenson, A.; and Dulac-Arnold, G. 2020. Model-based offline planning. *arXiv preprint arXiv:2008.05556*.

Brandfonbrener, D.; Whitney, W.; Ranganath, R.; and Bruna, J. 2021. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34: 4933–4946.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *CoRR*, abs/1606.01540.

Chen, B.; Monso, D. M.; Du, Y.; Simchowitz, M.; Tedrake, R.; and Sitzmann, V. 2024a. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Chen, C.; Deng, F.; Kawaguchi, K.; Gulcehre, C.; and Ahn, S. 2024b. Simple Hierarchical Planning with Diffusion. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *NeurIPS*, 15084–15097.

Chi, C.; Feng, S.; Du, Y.; Xu, Z.; Cousineau, E.; Burchfiel, B.; and Song, S. 2023. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In Bekris, K. E.; Hauser, K.; Herbert, S. L.; and Yu, J., eds., *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*.

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

Du, Y.; Yang, S.; Dai, B.; Dai, H.; Nachum, O.; Tenenbaum, J.; Schuurmans, D.; and Abbeel, P. 2023. Learning Universal Policies via Text-Guided Video Generation. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Emmons, S.; Eysenbach, B.; Kostrikov, I.; and Levine, S. 2022. RvS: What is Essential for Offline RL via Supervised Learning? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Feng, L.; Gu, P.; An, B.; and Pan, G. 2024. Resisting Stochastic Risks in Diffusion Planners with the Trajectory Aggregation Tree. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, 2052–2062. PMLR.

Gupta, A.; Kumar, V.; Lynch, C.; Levine, S.; and Hausman, K. 2019. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In Kaelbling, L. P.; Kragic, D.; and Sugiura, K., eds., *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, 1025–1037. PMLR.

He, H.; Bai, C.; Xu, K.; Yang, Z.; Zhang, W.; Wang, D.; Zhao, B.; and Li, X. 2023. Diffusion Model is an Effective Planner and Data Synthesizer for Multi-Task Reinforcement Learning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Ho, J.; and Salimans, T. 2022. Classifier-Free Diffusion Guidance. *CoRR*, abs/2207.12598.

Janner, M.; Du, Y.; Tenenbaum, J. B.; and Levine, S. 2022. Planning with Diffusion for Flexible Behavior Synthesis. *CoRR*, abs/2205.09991.

Janner, M.; Li, Q.; and Levine, S. 2021. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *NeurIPS*, 1273–1286.

Kang, B.; Ma, X.; Du, C.; Pang, T.; and Yan, S. 2023. Efficient Diffusion Policies For Offline Reinforcement Learning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. MOReL: Model-Based Offline Reinforcement Learning. In *NeurIPS*.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*. OpenReview.net.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *NeurIPS*, 11761–11771.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *NeurIPS*.

Lee, K.; Kim, S.; and Choi, J. 2023. Refining Diffusion Planner for Reliable Behavior Synthesis by Automatic Detection of Infeasible Plans. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Li, W.; Wang, X.; Jin, B.; and Zha, H. 2023. Hierarchical Diffusion for Offline Decision Making. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 20035–20064. PMLR.

Lu, C.; Ball, P. J.; Teh, Y. W.; and Parker-Holder, J. 2023. Synthetic Experience Replay. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Lyu, J.; Ma, X.; Li, X.; and Lu, Z. 2022. Mildly Conservative Q-Learning for Offline Reinforcement Learning. *ArXiv*, abs/2206.04745.

Nachum, O.; Dai, B.; Kostrikov, I.; Chow, Y.; Li, L.; and Schuurmans, D. 2019. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*.

Ni, F.; Hao, J.; Mu, Y.; Yuan, Y.; Zheng, Y.; Wang, B.; and Liang, Z. 2023. MetaDiffuser: Diffusion Model as Conditional Planner for Offline Meta-RL. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 26087–26105. PMLR.

Nichol, A. Q.; and Dhariwal, P. 2021. Improved Denoising Diffusion Probabilistic Models. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 8162–8171. PMLR.

Pathak, D.; Mahmoudieh, P.; Luo, G.; Agrawal, P.; Chen, D.; Shentu, Y.; Shelhamer, E.; Malik, J.; Efros, A. A.; and Darrell, T. 2018. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2050–2053.

Peng, X. B.; Kumar, A.; Zhang, G.; and Levine, S. 2019. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *ArXiv*, abs/1910.00177.

Song, J.; Meng, C.; and Ermon, S. 2021. Denoising Diffusion Implicit Models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Wang, Z.; Hunt, J. J.; and Zhou, M. 2022. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*.

Wu, J.; Wu, H.; Qiu, Z.; Wang, J.; and Long, M. 2022. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31278–31291.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

Yang, Y.; Jiang, J.; Zhou, T.; Ma, J.; and Shi, Y. 2021. Pareto policy pool for model-based offline reinforcement learning. In *International Conference on Learning Representations*.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J. Y.; Levine, S.; Finn, C.; and Ma, T. 2020. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142.

Zhu, Z.; Liu, M.; Mao, L.; Kang, B.; Xu, M.; Yu, Y.; Ermon, S.; and Zhang, W. 2023. MADiff: Offline Multi-agent Learning with Diffusion Models. *CoRR*, abs/2305.17330.