Offline Goal-Conditioned Reinforcement Learning with Elastic-Subgoal Diffused Policy Learning

Yaocheng Zhang
Institute of Automation, Chinese Academy of Sciences
Beijing, CHINA
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, CHINA
zhangyaocheng2023@ia.ac.cn

Yuanheng Zhu
Institute of Automation, Chinese Academy of Sciences
Beijing, CHINA
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, CHINA
yuanheng.zhu@ia.ac.cn

Yuqian Fu
Institute of Automation, Chinese Academy of Sciences
Beijing, CHINA
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, CHINA
fuyuqian2022@ia.ac.cn

Songjun Tu
Institute of Automation, Chinese Academy of Sciences
Beijing, CHINA
Pengcheng Laboratory
Shenzhen, CHINA
tusongjun2023@ia.ac.cn

Dongbin Zhao
Institute of Automation, Chinese Academy of Sciences
Beijing, CHINA
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, CHINA
dongbin.zhao@ia.ac.cn

ABSTRACT

Goal-conditioned reinforcement learning (GCRL) aims to learn a policy that generalizes across different goal conditions. Compared to non-hierarchical methods, hierarchical GCRL based on subgoals can alleviate the problem of inaccurately estimating the value function for faraway goals in offline learning scenarios, thereby leading to more effective policy learning. Due to the state complexity of the decision-making process, at different states, we require subgoals from varying future time steps to minimize policy errors caused by noisy value functions, rather than using a fixed future time step for selecting subgoals. Therefore, we propose a hierarchical reinforcement learning algorithm with an elastic subgoal steps, called *ESD* (***E**lastic **S**ubgoal **D**iffused Policy Learning*). Our method defines a novel high-level policy in which all reachable states surrounding the current state are considered as potential subgoals, and the optimal subgoal is selected among them. Moreover, we use diffusion models to represent the hierarchical policies, enhancing their ability to capture the multimodal data distribution introduced by the elastic subgoal steps and offline data. We evaluate the performance of ESD on multiple goal-conditioned benchmarks, and it demonstrates superior performance compared to previous baselines. Our method effectively reduces the impact of inaccurate value function estimates on policy accuracy, especially in complex tasks and high-dimensional image observations. Code is available at https://github.com/zhyaoch/ESD.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

KEYWORDS

Elastic Subgoal, Goal-conditioned Reinforcement Learning, Diffusion Model, Hierarchical Reinforcement Learning

ACM Reference Format:
Yaocheng Zhang, Yuanheng Zhu, Yuqian Fu, Songjun Tu, and Dongbin Zhao. 2025. Offline Goal-Conditioned Reinforcement Learning with Elastic-Subgoal Diffused Policy Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) trains agents to optimize specific rewards of interest. However, designing an appropriate reward function is often challenging for many tasks [22, 28, 31, 35]. Goal-conditioned reinforcement learning (GCRL), as an extension of traditional RL, can leverage large amounts of unlabeled (reward-free) data to learn goal-conditioned policies, avoiding the difficulty of designing reward functions [12, 24]. In real-world scenarios, interacting with the environment often generates large amounts of unlabeled datasets. Offline GCRL, as a new decision-making approach, can utilize these existing static datasets to learn effective policies [7, 16, 18, 29]. However, with these unlabeled datasets, offline GCRL often needs to address the challenges such as sparse rewards and long horizon problems associated with faraway goals. Moreover, when the original goal $\boldsymbol{g}$ is far from the current state, the goal-conditioned value function learned from offline data may have significant noise and fail to provide a clear learning signal [24], leading to a suboptimal policy.

To tackle these challenges, previous work has used hierarchical methods to decompose complex tasks into simpler sub-tasks,

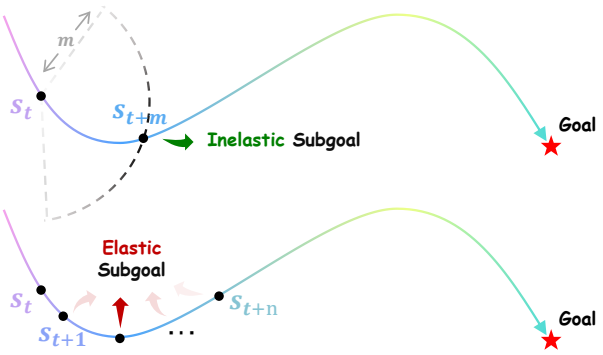

Figure 1: Comparison of inelastic subgoal vs. elastic-subgoal hierarchical policies. The inelastic-subgoal hierarchical policy designates (fixed) m-step away state s_{t+m} as the subgoal for the current state s_t, while the elastic-subgoal hierarchical policy selects the most critical subgoals from a range of time spans $\{1, 2, \ldots, m, \ldots, n\}$.

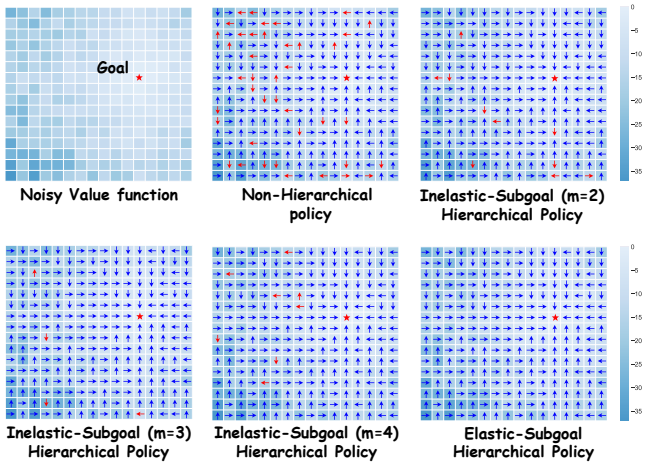

Figure 2: Elastic-subgoal hierarchical policy can effectively address the issue of noisy value estimates compared to inelastic-subgoal hierarchical policy, and make the optimal decision. The red arrows indicate the sub-optimal actions generated by the policy, while the blue arrows represent the optimal actions. In this gridworld environment, the ground-truth value function gives higher values for states that are closer to the goal. However, the estimated value function commonly suffers from noise disturbance and leads to incorrect action selection.

known as hierarchical GCRL [10, 17, 30]. This hierarchical structure effectively addresses the long-horizon and sparse reward problems, achieving good performance in many scenarios. In particular, the subgoal-based hierarchical structure decomposes the decision-making process into subgoals, which to some extent mitigates the impact of noise in the value function on the policy, as seen in methods like HIQL [24]. Nonetheless, the selection of subgoals plays a critical role in the performance of hierarchical policies. Inappropriate subgoals can exacerbate the negative impact of noise in the value function, leading to reduced policy accuracy. For example, it is unreasonable to use a long subgoal step for decision-making when the agent is close to the goal, or a short subgoal step when the agent is far from the goal. This implies that using an inelastic (fixed) span for subgoal selection (Fig. 1) in all decision stages, as is the case in HIQL, is an unreasonable hierarchical approach. Fig. 2 demonstrates an example, showing that if the subgoal steps is fixed to a constant length, the hierarchical policies suffer the failure of finding optimal solutions, just the same as non-hierarchical policy. Therefore, a more natural and appropriate hierarchical approach is to employ subgoal steps of different lengths at different decision stages (i.e., using an elastic subgoal). Fig. 2 confirms this intuitive idea, showing that hierarchical policy based on elastic subgoals is better at resisting noise, and it is also more likely to learn an optimal solution.

In this work, we introduce the concept of the elastic subgoals (Fig. 1) into the hierarchical RL to mitigate the impact of noise in the value function on the policy accuracy. Similar to previous work, we decompose the original complex Markov Decision Process (MDP) into a high-level MDP and a low-level MDP [17, 30]. However, unlike previous approaches, we use the neighborhood of states as the action set in the high-level MDP, enabling the high-level policy to select optimal subgoals with an elastic subgoal step. Experimental results show that using state neighborhoods as the action set for the high-level policy can effectively mitigate the impact of noise in the value function on the algorithm. Moreover, when using state neighborhoods as candidate subgoals, there may be multiple neighboring states on the optimal trajectory that are equivalent subgoals (Appendix **??**). For the policy model, these equivalent subgoals exhibit a typical multimodal distribution. In this case, using a Gaussian model to represent the policy is inadequate, and a generative model capable of effectively representing multimodal distributions is needed. Therefore, we use diffusion models, known for their strong representational capabilities [13, 19, 20, 27], to represent both high-level and low-level policy models. This ensures that the policy model can effectively fit the multimodal distribution resulting from multiple equivalent subgoals and accurately represent actions under different goals. Meanwhile, we employ an advantage-weighted noise objective to train the diffusion model, implicitly guiding the distribution of the policy during training [11, 34]. This approach avoids the Out-of-Distribution (OOD) problems [2, 8, 33] that arise from directly using the value function to guide noisy data during the denoising process, generating the higher-quality policies.

In summary, our contributions include:**(i)** We propose ***Elastic Subgoal Diffused Policy Learning (ESD)*** for goal-conditioned offline RL problems, which introduces state neighborhoods as the action set for high-level policy within a hierarchical structure. **(ii)** To model the multimodal distribution produced under elastic subgoal steps, we introduce diffusion models to represent policies, and employ an advantage-weighted noise objective to train the diffusion model to generate high-quality policy. **(iii)** We evaluate ESD on multiple goal-conditioned benchmarks and achieve competitive performance compared to existing offline GCRL algorithms.

2 PRELIMINARIES

2.1 Goal-Conditioned Offline RL

The offline GCRL problem is formulated in the context of a Markov decision process $\{\mathcal{S}, \mathcal{A}, \rho_0, T, R, \gamma, \mathcal{G}\}$ and a dataset $\mathcal{D}$, where $\rho_0 \in \mathcal{P}(\mathcal{S})$ denotes an initial state distribution, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $T \in \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the environmental dynamics, $\mathcal{G}$ denotes goal space, $R(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{a}_t)$ denotes a goal conditioned reward function, and γ is the discount factor. Following the work [24], we assume that the goal space is the same as the state space (*i.e.*, $\mathcal{G} = \mathcal{S}$). The goal of offline GCRL is to find an optimal policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{g}) : \mathcal{S} \times \mathcal{G} \to \mathcal{P}(\mathcal{A})$ to maximize the expected return:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{a}_t) | \boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t, \boldsymbol{g}), \boldsymbol{s}_{t+1} \sim T(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)\right]. \quad (1)$$

Unlike online GCRL methods, offline GCRL learns from a fixed dataset $\mathcal{D}$. Since this dataset often has sparse coverage and contains suboptimal data, the estimated optimal value function is noisy. Therefore, deriving a more accurate policy π_ϕ from the noisy value function becomes crucial.

2.2 In-sample Learning via Expectile Regression

Offline RL often encounters the challenge of out-of-distribution problems, leading to overestimation of the value function during training. To solve this issue, IQL [16] avoids querying out-of-sample actions by replacing the *max* operator in the Bellman optimal equation with expectile regression function:

$$\min_{V} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)}\left[L_2^\lambda(\overline{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t) - V(\boldsymbol{s}_t))\right], \quad (2)$$

$$\min_{Q} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})}\left[(\boldsymbol{r}_t + \gamma V(\boldsymbol{s}_{t+1}) - Q(\boldsymbol{s}_t, \boldsymbol{a}_t))^2\right], \quad (3)$$

where $\overline{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ denotes the target network [21]. The expectile regression function $L_2^\lambda(\boldsymbol{u})$ is represented as $|\lambda - \mathbb{I}(\boldsymbol{u} < 0)\boldsymbol{u}^2|$, where $\mathbb{I}(\cdot)$ denotes the indicator function. After training the value functions $Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and $V(\boldsymbol{s}_t)$, IQL extracts the policy from behavior policy with advantage weighted regression (AWR)[25]:

$$\max_{\pi} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)}\left[\exp(\beta(\overline{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t) - V(\boldsymbol{s}_t))) \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\right], \quad (4)$$

where $\beta \in \mathbb{R}_0^+$ is the temperature parameter. In addition, there are also action-free variant [9, 32] of IQL to learn an offline goal-conditioned V-function without Q-function:

$$\min_{V} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})}\left[L_2^\lambda(\boldsymbol{r}_t + \gamma \overline{V}(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t))\right]. \quad (5)$$

Similar to IQL, the policy can be extracted using the following variant of AWR:

$$\max_{\pi} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})}\left[\exp(\beta(\boldsymbol{r}_t + \gamma \overline{V}(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t))) \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\right]. \quad (6)$$

2.3 Diffusion Model

The diffusion model [13] consists of two processes: the noising (forward) process and the denoising (sampling) process. The noising process is modeled as a Markov chain with length K, denoted as $x^{0:K}$. This process involves continuously injecting noise into the original data $\boldsymbol{x} \sim \mathcal{D}$[1] with conditional probability:

$$q(\boldsymbol{x}^k|\boldsymbol{x}^{k-1}) = \mathcal{N}(\boldsymbol{x}^k|\sqrt{\alpha^k}\boldsymbol{x}^{k-1}, (1-\alpha^k)\boldsymbol{I}), \quad (7)$$

where $k \in [1, K]$, $\alpha^{0:k}$ are hyperparameters controlling the variance schedule. Eventually, the data follows a white noise distribution. The denoising process is parameterized by the Markov chain:

$$p_\theta(\boldsymbol{x}^{k-1}|\boldsymbol{x}^k) = \mathcal{N}(\boldsymbol{x}^{k-1}|\boldsymbol{\mu}_\theta(\boldsymbol{x}^k, k), \mathcal{E}^k), \quad (8)$$

where $\boldsymbol{x}^K \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\mathcal{E}^k$ denotes $(1-\alpha^k)\boldsymbol{I}$. This process iteratively samples from a Gaussian distribution to eliminate noise from the data until the original data is reconstructed. The training is performed by minimizing the variational bound $\mathbb{E}_q\left[-\log \frac{p_\theta(\boldsymbol{x}^{0:K})}{q(\boldsymbol{x}^{1:K}|\boldsymbol{x})}\right]$ of the cross-entropy loss:

$$\mathcal{H}(q(\boldsymbol{x}), p_\theta(\boldsymbol{x})) = \mathbb{E}_q\left[-\log p_\theta(\boldsymbol{x})\right]. \quad (9)$$

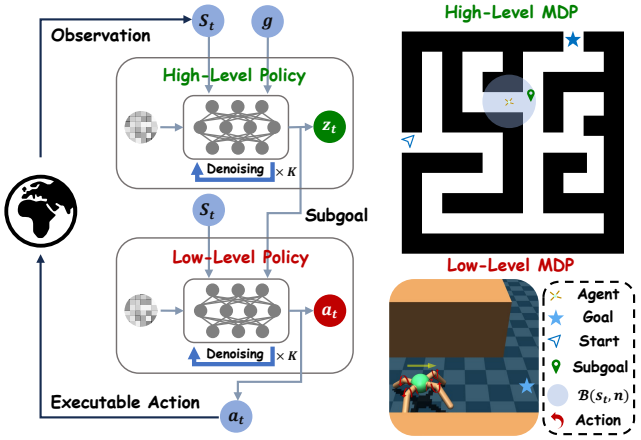

Figure 3: Hierarchical diffusion policy based on elastic subgoals. We decompose the original MDP into a high-level and a low-level MDPs, and construct the high-level policy and low-level policy from them by diffusion models.

3 HIERARCHICAL DECISION STRUCTURE FOR GOAL-CONDITIONED OFFLINE RL

The combination of hierarchical structure with GCRL decomposes difficult tasks arising from distant goals, thereby improving policy performance. However, different hierarchical structures lead to different decision processes and result in different final results. In this section, we first introduce two hierarchical structures with elastic subgoals and inelastic subgoals, respectively, and compare their performance in a grid environment. Second, the theoretical analysis of two different structures on a toy problem shows the advantage of elastic subgoals in mitigating policy errors.

3.1 Hierarchical Policy Structure

Following prior work [17, 30], we extend the standard RL setup to a hierarchical two-layer structure (Fig. 3) with a high-level policy $\pi^h(\boldsymbol{z}_t|\boldsymbol{s}_t, \boldsymbol{g})$ and a low-level policy $\pi^\ell(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{z}_t)$. Here, $\boldsymbol{z}_t$ can be

[1]For convenience, we use $\boldsymbol{x}$ to denote $\boldsymbol{x}^0$.

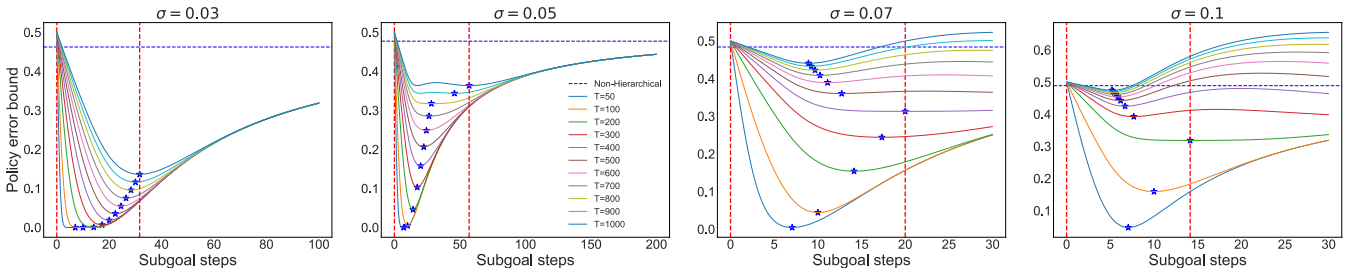

Figure 4: Comparison of the subgoal steps that minimizes policy errors across different states and noise scenarios. The optimal subgoal steps (the horizontal coordinates of blue stars) vary across different agent states, making it unreasonable to use fixed steps and inelastic subgoals.

viewed as a subgoal, or a waypoint of state trajectory. The high-level policy is responsible for decomposing the task and identifying subgoals z_t that effectively guide the agent towards the goal, while the low-level policy is responsible for outputting the optimal action to reach the subgoal z_t. This subgoal-based hierarchical structure provides more accurate learning signals for both the high-level and low-level policies, thereby mitigating the impact of noise in the value function on policy accuracy [24]. For the high-level policy, the value differences between subgoals are more significant compared to the original action space, offering clearer learning signals. For the low-level policy, using nearby states as the goal allows for more accurate value estimation of the action compared to distant goals.

For the high-level policy, one common approach [24] to select the subgoal z_t is to use intermediate states s_{t+m} that are m steps away from the current state s_t, where m is a constant value across all decision stages. We refer to this approach as *inelastic-subgoal hierarchical policy* in this paper (Fig. 1). However, due to the complexity of decision-making process, at different states, we require subgoals from varying time steps to better decompose the original goal (see Section 3.2). Therefore, we propose another approach where the intermediate states s_{t+m} can correspond to different time steps m at various decision stages. We refer to this approach as *elastic-subgoal hierarchical policy* in this paper (Fig. 1). As shown in Fig. 2, the hierarchical structure of the elastic subgoal is more robust against the noise in the value function compared to the inelastic subgoal. This phenomenon is intuitive, as the elastic subgoal naturally simulates the human planning process, where the time span between subgoals is adjusted based on current circumstances, thereby enhancing the robustness and effectiveness of the policy.

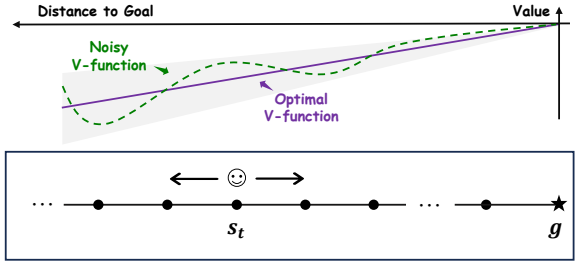

Figure 5: 1-D toy environment.

3.2 Theoretical Analysis: Inelastic vs. Elastic Subgoal

To further illustrate the advantage of the elastic subgoal, we study a toy example (see Fig. 5) with one-dimensional state space proposed by Park et al. [24]. In this environment, the reward signal is a sparse goal-conditioned reward function, $r(s_t, g) = 0$ (if $s_t = g$) or -1 (otherwise). Therefore, the optimal value function can be written as $V^*(s_t, g) = -|s_t - g|$ (assuming $\gamma = 1$). Prior work [24] further assumes that there exists noise in the learned value function and the magnitude of noise is proportional to the magnitude of optimal value, i.e. $\hat{V}(s_t, g) = V^*(s_t, g) + \delta|V^*(s_t, g)|$, where δ is sampled from a Gaussian distribution with standard deviation σ, denoted as $\mathcal{N}(0, \sigma^2)$.

Following Park et al. [24], the probability of the non-hierarchical policy π selecting an incorrect action is given as $\varepsilon(\pi) = \Phi(-\frac{\sqrt{2}}{\sigma\sqrt{T^2+1}})$, where Φ denotes the cumulative distribution function of the standard normal distribution ($\Phi(x) = P[\delta \le x] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$), T denotes the distance between the agent and the goal. For the inelastic-subgoal hierarchical policy $\pi^h \circ \pi^\ell$, the probability of selecting an incorrect action is tightly bounded as $\varepsilon(\pi^h \circ \pi^\ell) \le \varepsilon(\pi^h) + \varepsilon(\pi^\ell) = \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{(T/m)^2+1}}\right) + \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{m^2+1}}\right)$, where $\circ$ represents function composition.

From the above proposition, we can observe that the optimal subgoal steps m^* (that minimize the $\varepsilon(\pi^\ell \circ \pi^h)$) vary with the distance T between the agent and the goal. As shown in Fig. 4, the curves illustrate how the $\varepsilon(\pi^\ell \circ \pi^h)$ changes with different subgoal steps m for various state distances T. The horizontal coordinates of the blue stars indicate the optimal subgoal steps m^* for each T. As T changes, the optimal m^* varies and does not follow a simple mapping with T. This implies that using fixed horizon steps for subgoal selection across all decision stages is non-optimal. Therefore, selecting elastic subgoals within a sufficient large range of time steps has more potential to find the optimal solutions.

4 ELASTIC SUBGOAL DIFFUSED POLICY LEARNING

Based on the analysis in the previous section, using an elastic hierarchical policy to suppress the impact of noise in the value function is promising. We propose a new algorithm, Elastic Subgoal

Diffused Policy Learning (ESD) to solve the offline GCRL problem. ESD consists of a high-level and a low-level policy, both based on a diffusion model. The high-level policy is responsible for setting subgoals using elastic subgoal steps, while the low-level policy plans the concrete actions to reach the subgoals. In this section, we first describe the definitions of the high-level and low-level MDPs in ESD. Second, we describe how to learn the value function simultaneously in both the high-level MDP and low-level MDP. Finally, we propose a new advantage-weighted noise objective to assist the diffusion model in extracting the policy from the value function.

4.1 High-level and Low-level MDPs

The effectiveness of hierarchical RL largely depends on the selection of candidate subgoals. As motivated in Section 3.2, using fixed-length subgoal steps for selecting subgoals across all decision stages is unreasonable. Adopting different subgoal steps at various decision stages allows for better suppression of noise. Therefore, our method, ESD, employs an approach based on elastic subgoal steps to improve subgoal selection. To achieve this, we first introduce the n-neighborhood $\mathcal{B}(\boldsymbol{s}_t, n)$ of state $\boldsymbol{s}_t$:

$$\mathcal{B}(\boldsymbol{s}_t, n) := \{z_t \mid \forall \boldsymbol{\tau}(\boldsymbol{s}_t, \boldsymbol{z}_t), 1 < d_{\boldsymbol{\tau}} \leq n\},$$

where n represents the maximum subgoal steps of the agent, $d_{\boldsymbol{\tau}}$ refers to the length of the state-only trajectory segment $\boldsymbol{\tau}(\boldsymbol{s}_t, \boldsymbol{z}_t) = (\boldsymbol{s}_t, \boldsymbol{s}_{t+1}, \ldots, \boldsymbol{s}_{t+d_{\boldsymbol{\tau}}(\boldsymbol{s}_t, \boldsymbol{z}_t)-1}, \boldsymbol{z}_t)$. Thus, $\boldsymbol{z}_t \in \mathcal{B}(\boldsymbol{s}_t, n)$ indicates that the subgoal $\boldsymbol{z}_t$ can be reached from $\boldsymbol{s}_t$ within n steps.

Based on $\mathcal{B}(\boldsymbol{s}_t, n)$, we define the high-level MDP MDP^h, where its state space $\mathcal{S}^h$ remains the same as the original MDP. For each state $\boldsymbol{s}_t \in \mathcal{S}^h$, $\mathcal{A}^h(\boldsymbol{s}_t)$ denotes the set of subgoal states within the neighborhood $\mathcal{B}(\boldsymbol{s}_t, n)$. Each subgoal $\boldsymbol{z}_t \in \mathcal{A}^h(\boldsymbol{s}_t)$ represents a deterministic transition from state $\boldsymbol{s}_t$ to the subgoal $\boldsymbol{z}_t$. The reward for transitioning from state $\boldsymbol{s}_t$ to subgoal $\boldsymbol{z}_t$ is given by:

$$R^h(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t) = \sum_{i=0}^{d_{\boldsymbol{\tau}^*}(\boldsymbol{s}_t, \boldsymbol{z}_t)-1} \gamma^i r^{\boldsymbol{\tau}}_{t+i}, \tag{10}$$

where $\boldsymbol{\tau}^*(\boldsymbol{s}_t, \boldsymbol{z}_t)$ is the optimal trajectory to reach subgoal $\boldsymbol{z}_t$ from state $\boldsymbol{s}_t$, $d_{\boldsymbol{\tau}^*}$ denotes the length of the $\boldsymbol{\tau}^*$, $r^{\boldsymbol{\tau}}_{t+i}$ represents the i-th reward in the path $\boldsymbol{\tau}^*(\boldsymbol{s}_t, \boldsymbol{z}_t)$. Therefore, $R^h(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t)$ represents the sum of rewards along the optimal path $\boldsymbol{\tau}^*(\boldsymbol{s}_t, \boldsymbol{z}_t)$ from state $\boldsymbol{s}_t$ to subgoal $\boldsymbol{z}_t$. Thus, the optimal Q-function $Q^{h*}(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t)$ and V-function $V^{h*}(\boldsymbol{s}_t, \boldsymbol{g})$ are defined as follows:

$$\begin{aligned} Q^{h*}(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t) &= R^h(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t) + \Gamma(\boldsymbol{s}_t, \boldsymbol{z}_t) V^{h*}(\boldsymbol{z}_t, \boldsymbol{g}) \\ &= \max_{\boldsymbol{\tau}(\boldsymbol{s}_t, \boldsymbol{z}_t)} \left[\sum_{i}^{d_{\boldsymbol{\tau}}} \gamma^i \boldsymbol{r}^{\boldsymbol{\tau}}_{t+i} + \gamma^{d_{\boldsymbol{\tau}}} V^{h*}(\boldsymbol{z}_t, \boldsymbol{g}) \right] \end{aligned} \tag{11}$$

$$V^{h*}(\boldsymbol{s}_t, \boldsymbol{g}) = \max_{\boldsymbol{z}_t \in \mathcal{B}(\boldsymbol{s}_t, n)} Q^{h*}(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t), \tag{12}$$

where the discount factor $\Gamma(\boldsymbol{s}_t, \boldsymbol{z}_t) = \gamma^{d_{\boldsymbol{\tau}^*}}$. Due to the difficulty of directly obtaining rewards $R^h(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t)$ and discount factors $\Gamma(\boldsymbol{s}_t, \boldsymbol{z}_t)$ corresponding to the optimal transition trajectory, we use the *max* operator in Eq. (11) to represent them, extracting the corresponding values from the offline dataset. Using the aforementioned value function, we can select the optimal subgoal $\boldsymbol{z}_t$ within a sufficient large neighbor of current s_t to guide the agent, thereby decomposing the original goal with elastic subgoals.

Following Park et al. [24], given a subgoal $\boldsymbol{z}_t$ provided by the high-level policy π^h, we define the low-level MDP MDP^ℓ as an MDP with $\boldsymbol{z}_t$ as the goal. Although MDP^ℓ retains the same state space, action space, and dynamics as the original MDP, the decision-making process is simplified from a long-horizon problem to a short-horizon problem. This simplification allows the low-level policy to receive a clear learning signal, as it queries the value function with only nearby states, thereby reducing the overall task complexity.

4.2 Offline RL for Value Function

We now demonstrate that it is possible to simultaneously obtain the optimal value functions for both the high-level and low-level MDPs through offline RL. To achieve this, we first introduce the following proposition:

Proposition 1. *Given arbitrary $\boldsymbol{g} \in \mathcal{G}, \boldsymbol{s} \in \mathcal{S}$, let $V^{h*}(\boldsymbol{s}, \boldsymbol{g})$ be the optimal V-function in high-level MDP^h, $V^{\ell*}(\boldsymbol{s}, \boldsymbol{g})$ be the optimal V-function in low-level MDP^ℓ. If environment dynamics is deterministic [9], optimal V-function $V^{h*}(\boldsymbol{s}, \boldsymbol{g})$ and optimal V-function $V^{\ell*}(\boldsymbol{s}, \boldsymbol{g})$ are equivalent:*

$$V^{h*}(\boldsymbol{s}, \boldsymbol{g}) = V^{\ell*}(\boldsymbol{s}, \boldsymbol{g}) \quad \textit{for all} \quad \boldsymbol{s} \in \mathcal{S}, \quad \boldsymbol{g} \in \mathcal{G}$$

The proof can be found in the supplementary Appendix **??**. According to Proposition 1, we replace the value function $V^h(\boldsymbol{s}_t, \boldsymbol{z}_t)$ in Eq. (11) with the value function $V^\ell(\boldsymbol{s}_t, \boldsymbol{z}_t)$, resulting in Eq. (13).

$$Q^{h*}(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t) = \max_{\boldsymbol{\tau}(\boldsymbol{s}_t, \boldsymbol{z}_t)} \left[\sum_{i}^{d_{\boldsymbol{\tau}}} \gamma^i \boldsymbol{r}^{\boldsymbol{\tau}}_{t+i} + \gamma^{d_{\boldsymbol{\tau}}} V^{\ell*}(\boldsymbol{z}_t, \boldsymbol{g}) \right]. \tag{13}$$

Referring to IQL, we employ the expectile function to represent the additional *max* operator in Eq. (13), leading to Eq. (14). Moreover, we leverage the action-free variant of IQL to learn the value function $V^\ell(\boldsymbol{s}_t, \boldsymbol{z}_t)$ [9, 32]. Consequently, we can combine the learning processes for the value functions in both the high-level and the low-level MDPs into the following loss function:

$$\mathcal{L}_{Q^h}(\phi) = \mathbb{E}_{(\boldsymbol{\tau}(\boldsymbol{s}_t, \boldsymbol{z}_t), \boldsymbol{g})} \left[L_2^{\boldsymbol{\lambda}} \left(\chi^\ell - Q^h_\phi(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t) \right) \right] \tag{14}$$

$$\mathcal{L}_{V^\ell}(\psi) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{s}_{t+1})} \left[L_2^{\boldsymbol{\lambda}} (\boldsymbol{r}_t + \gamma \overline{V}^\ell_\psi(\boldsymbol{s}_{t+1}, \boldsymbol{g}) - V^\ell_\psi(\boldsymbol{s}_t, \boldsymbol{g})) \right], \tag{15}$$

where χ^ℓ equals $\sum_{i=0}^{d_{\boldsymbol{\tau}}} \gamma^i \boldsymbol{r}^{\boldsymbol{\tau}}_{t+i} + \gamma^{d_{\boldsymbol{\tau}}} \overline{V}^\ell_\psi(\boldsymbol{z}_t, \boldsymbol{g})$. During training, the goal $\boldsymbol{g}$ is chosen with a probability of 0.3 for a random state, 0.5 for a terminal state, and 0.2 for the current state [1, 9]. Note that we do not learn a Q^ℓ here. This is because, under the deterministic environment assumption in Proposition 1, we can unbiasedly replace Q^ℓ with $\boldsymbol{r}_t + \gamma \overline{V}^\ell_\psi(\boldsymbol{s}_{t+1}, \boldsymbol{z}_t)$ [9], thereby reducing the number of neural networks required and improving training efficiency. It should be noted that in non-deterministic environments, we need to additionally learn two networks, V^h (Appendix **??**) and Q^ℓ (using Eqs. (2) and (3)), to obtain more accurate results.

4.3 Diffusion Policy

Following AWR [16, 25, 26], we can directly extract the policy from behavior policy π_β and Q-function with a weighted distribution form:

$$\pi^h(\boldsymbol{z}_t | \boldsymbol{s}_t, \boldsymbol{g}) \propto \pi_\beta(\boldsymbol{z}_t | \boldsymbol{s}_t, \boldsymbol{g}) w(Q^h(\boldsymbol{s}_t, \boldsymbol{g}, \boldsymbol{z}_t)), \tag{16}$$

$$\pi^{\ell}(\boldsymbol{a}_t|\boldsymbol{s}_t,\boldsymbol{z}_t) \propto \pi_{\beta}(\boldsymbol{a}_t|\boldsymbol{s}_t,\boldsymbol{z}_t\ )w(Q^{\ell}(\boldsymbol{s}_t,\boldsymbol{z}_t,\boldsymbol{a}_t\ )), \tag{17}$$

where w is a monotonic increasing function (e.g. advantage-weighted function) used to assign a weight to each action. This can be abstracted as a classifier-guided method [3] in diffusion model:

$$p(\boldsymbol{x}) \propto q(\boldsymbol{x})f(\boldsymbol{x}), \tag{18}$$

where $\boldsymbol{x}$ represents $\boldsymbol{a}_t$ or $\boldsymbol{z}_t$. However, if directly using the weight function $w(Q)$ as a classifier f to guide behavior policy $q(\boldsymbol{x}) = \pi_{\beta}$ during denoising process [14]:

$$p_{\theta}(\boldsymbol{x}^{k-1}|\boldsymbol{x}^{k}) = \mathcal{N}(\boldsymbol{x}^{k-1}|\boldsymbol{\mu}_{\theta} + \eta\mathcal{E}^{k}\nabla_{\boldsymbol{x}^{k}}\log f(\boldsymbol{x}^{k}), \mathcal{E}^{k}), \tag{19}$$

we will face severe OOD problems. This is because the noisy data distribution during the denoising process (i.e., $p_{\theta}(\boldsymbol{x}^{k}), k = [1, K]$) differs significantly from the data distribution $q(\boldsymbol{x})$ generated by the behavior policy. Therefore, the gradient $\nabla_{\boldsymbol{x}^{k}}\log f(\boldsymbol{x}^{k})$ estimated by the learned value function for these OOD noisy data is inaccurate.

To address this issue, we employ an advantage-weighted noise objective to directly learn the original distribution $q(\boldsymbol{x})f(\boldsymbol{x})$ during training:

$$\mathbb{E}_{k,q,\boldsymbol{\epsilon}}\left[f(\boldsymbol{x})||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\boldsymbol{x}^{k}, k)||^{2}\right], \tag{20}$$

where we use the advantage function as the weight function f. The proof can be found in supplementary Appendix **??**. Through the above training objective, we can achieve the desired target policy using only Eq. (8) during sampling, without relying on classifier guidance. The sampling process can be reformulated as:

$$\boldsymbol{x}^{k-1} = \frac{1}{\sqrt{\alpha^{k}}}(\boldsymbol{x}^{k} - \frac{\beta^{k}}{\sqrt{1-\overline{\alpha}^{k}}}\boldsymbol{\epsilon}_{\theta}(\boldsymbol{x}^{k}, k)) + \sqrt{\beta^{k}}\boldsymbol{\epsilon}. \tag{21}$$

Therefore, during inference, we can discard the classifier to avoid computing gradients on noisy data, thereby preventing the OOD issues caused by using value function (or advantage function)-based classifier estimations on noisy data. Meanwhile, this approach avoids the introduction of time-consuming gradient backpropagation operations, thereby improving denoising efficiency.

Consequently, the loss function to learn policy $\pi^{h}_{\theta_h}(\boldsymbol{z}_t|\boldsymbol{s}_t,\boldsymbol{g})$ and $\pi^{\ell}_{\theta_\ell}(\boldsymbol{a}_t|\boldsymbol{s}_t,\boldsymbol{z}_t)$ using the diffusion model is as follow:

$$\mathcal{L}_{\pi^{h}}(\theta_h) = \mathbb{E}_{k,(\boldsymbol{s}_t,\boldsymbol{g},\boldsymbol{z}_t),\boldsymbol{\epsilon}}\left[\exp(\beta^{h}\cdot A^{h}(\boldsymbol{s}_t,\boldsymbol{g},\boldsymbol{z}_t))||\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_{z}||^{2}\right] \tag{22}$$

$$\mathcal{L}_{\pi^{\ell}}(\theta_\ell) = \mathbb{E}_{k,(\boldsymbol{s}_t,\boldsymbol{z}_t,\boldsymbol{a}_t),\boldsymbol{\epsilon}}\left[\exp(\beta^{\ell}\cdot A^{\ell}(\boldsymbol{s}_t,\boldsymbol{z}_t,\boldsymbol{a}_t))||\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_{a}||^{2}\right] \tag{23}$$

where $\hat{\boldsymbol{\epsilon}}_{z}$ denotes $\boldsymbol{\epsilon}_{\theta_h}(\boldsymbol{z}^{k}_{t}, k; (\boldsymbol{s}_t,\boldsymbol{g}))$, $\hat{\boldsymbol{\epsilon}}_{a}$ denotes $\boldsymbol{\epsilon}_{\theta_\ell}(\boldsymbol{a}^{k}_{t}, k; (\boldsymbol{s}_t,\boldsymbol{z}_t))$, β^{h} and β^{ℓ} denote the temperature parameter for high-level and low-level policies respectively. During training, we use the advantage function $A^{h}(\boldsymbol{s}_t,\boldsymbol{g},\boldsymbol{z}_t) = Q^{h}_{\phi}(\boldsymbol{s}_t,\boldsymbol{g},\boldsymbol{z}_t) - V^{\ell}_{\psi}(\boldsymbol{s}_t,\boldsymbol{g})$ and $A^{\ell}(\boldsymbol{s}_t,\boldsymbol{z}_t,\boldsymbol{a}_t) = \boldsymbol{r}_t + \gamma\overline{V}^{\ell}_{\psi}(\boldsymbol{s}_{t+1},\boldsymbol{z}_t) - V^{\ell}_{\psi}(\boldsymbol{s}_t,\boldsymbol{z}_t)$ instead of the Q-function in Eq.(16) and (17) as the weight function f for smaller variance. This will make the learning process of the diffusion model more stable.

So far, we have explained all modules utilized within the ESD (Fig. 3). Our algorithm consists of two stages. First, we learn the goal-conditioned value function of both the high-level MDP and the low-level MDP together, performing a number of gradient updates alternating between Eqs. (14) and (15). Second, we perform stochastic gradient descent on Eqs. (22) and (23) to learn the goal-conditioned diffusion policy. The entire training process is summarized in Algorithm 1.

Algorithm 1 Training Stage of the ESD

Require:
High-level policy model $\pi^{h}_{\theta_h}$, Low-level policy model $\pi^{\ell}_{\theta_\ell}$, Q-function Q^{h}_{ϕ}, V-function V^{ℓ}_{ψ}, Offline dataset $\mathcal{D}$

Initialize $\pi^{h}_{\theta_h}$, $\pi^{\ell}_{\theta_\ell}$, Q^{h}_{ϕ}, V^{ℓ}_{ψ}

#learn value function
For step $j \leftarrow 1$ to M do:
  Update ϕ by minimizing $\mathcal{L}_{Q^{h}}(\phi)$ with $(\boldsymbol{\tau}(\boldsymbol{s}_t,\boldsymbol{z}_t),\boldsymbol{g}) \sim \mathcal{D}$
  Update ψ by minimizing $\mathcal{L}_{V^{\ell}}(\psi)$ with $(\boldsymbol{s}_t,\boldsymbol{g},\boldsymbol{s}_{t+1}) \sim \mathcal{D}$
End for

#learn hierarchical policy
For step $j \leftarrow 1$ to M do:
  Update θ_h by minimizing $\mathcal{L}_{\pi^{h}}(\theta_h)$ with $(\boldsymbol{s}_t,\boldsymbol{g},\boldsymbol{z}_t) \sim \mathcal{D}$
  Update θ_ℓ by minimizing $\mathcal{L}_{\pi^{\ell}}(\theta_\ell)$ with $(\boldsymbol{s}_t,\boldsymbol{z}_t,\boldsymbol{a}_t) \sim \mathcal{D}$
End for

5 EXPERIMENT

In this section, we conduct a series of experiments to answer the following questions. 1) Does ESD exhibit performance advantages over previous offline GCRL algorithms? 2) Does the diffusion model outperform the unimodal model when used as policy models? 3) Can ESD better mitigate policy errors caused by noisy value functions? 4) How do the modules of ESD influence its performance?

5.1 Does ESD Exhibit Performance Advantages over Previous Offline GCRL Algorithms?

Baselines. We compare various goal-conditioned offline decision algorithms with the ESD. The compared baselines include imitation learning like goal-conditioned behavioral cloning (GCBC) [4], RvS-G [5], and hierarchical goal-conditioned behavioral cloning (HGCBC) [23]; sequence models like the Trajectory Transformer (TT) [15]; and offline GCRL methods such as the goal-conditioned variant of IQL [16] (GC-IQL) and CRL [6]. Additionally, we include hierarchical offline GCRL methods like HIQL (which uses the inelastic subgoal) [24] and goal-conditioned variant of POR [32] (GC-POR), which employs hierarchy but utilizes a shorter inelastic subgoal (i.e. predict the immediate next state as a subgoal).

We evaluate ESD on various benchmarks (for a detailed description of the benchmarks, please refer to supplementary Appendix **??**) and present all test results in Table 1. In AntMaze environments, ESD outperforms all other baselines. In Kitchen environments, ESD performs competitively with the best performance of prior methods. On the more challenging AntSoccer, CALVIN and Procgen environment, ESD outperforms all other baselines by a large margin. In all these environments, ESD surpasses the hierarchical algorithms HQIL and POR, which use a fixed length of $m(> 1)$ and 1 for inelastic subgoal selection. We conjecture that the success in all those datasets should be attributed to the effective planning capability

Table 1: Performance on the offline goal-condtioned benchmarks. We show that ESD either matches or outperforms the current offline GCRL algorithm across various environments. The normalized scores are reported using a multiplier of 25 for Kitchen and CALVIN tasks, and 100 for all other tasks. We present the mean and the standard error over 8 random seeds. The top two scores per task are highlighted in dark blue and light blue, respectively.

Dataset	GCBC	HGCBC	GC-IQL	GC-POR	TT	CRL	RvS-G	HIQL	ESD-Gauss	ESD
AntMaze-Umaze	56.7	57.5	89.2	90.4	100.0	-	65.4	83.3	97.5±3.2	97.1±2.6
AntMaze-Umaze-Diverse	57.1	53.8	68.5	68.2	-	-	60.9	85.4	93.3±5.8	92.9±4.2
AntMaze-Medium-Play	67.3	71.6	63.5	74.8	100.0	-	58.1	86.8	89.2±3.2	90.8±6.4
AntMaze-Medium-Diverse	71.9	66.3	70.9	71.4	93.3	-	57.3	84.1	85.0±6.0	88.3±6.0
AntMaze-Large-Play	20.2	63.9	50.7	49.0	60.0	-	32.4	88.2	89.6±4.2	88.8±6.0
AntMaze-Large-Diverse	23.1	64.7	56.5	63.2	66.7	-	36.9	86.1	84.6±5.3	87.9±5.0
AntMaze-Ultra-Play	14.4	39.4	21.6	29.8	33.3	-	-	52.9	51.7±19.3	56.7±9.1
AntMaze-Ultra-Diverse	20.7	38.2	29.8	31.0	20.0	-	-	39.2	50.6±24.1	55.8±11.3
Average	41.4	56.9	56.3	59.6	-	-	-	75.7	80.2	82.3
Kitchen-Partial	38.5	32.0	39.2	18.4	-	51.4	-	65.0	66.2±5.6	69.8±2.1
Kitchen-Mixed	46.7	46.8	51.3	27.9	-	60.3	-	67.7	68.7±7.3	67.1±5.0
Average	42.6	39.4	45.3	23.2	-	54.0	-	66.4	67.5	68.5
AntSoccer-Arena-Navigate	5.0	4.2	50.0	61.7	-	23.0	-	58.0	62.5±12.7	77.5±8.3
AntSoccer-Arena-Stitch	2.0	0	7.0	9.6	-	1.0	-	13.0	17.1±2.6	39.6±10.3
Average	3.5	2.1	28.5	35.7	-	12.0	-	35.5	39.8	58.6
Scene-Play	5.0	6.3	51.0	50.0	-	19.0	-	38.0	60.0±7.8	64.2±8.9
Scene-Noisy	1.0	1.7	26.0	17.5	-	1.0	-	25.0	29.6±6.8	47.1±7.7
Average	3.0	4.0	38.5	33.8	-	10.0	-	31.5	44.8	55.7
Procgen-Maze-500-Train	16.8	14.3	72.5	75.8	-	-	-	82.5	80.8±6.4	84.2±5.0
Procgen-Maze-500-Test	14.5	11.2	49.5	53.8	-	-	-	64.5	70.8±10.0	72.2±8.5
Procgen-Maze-1000-Train	27.2	15.0	78.2	82	-	-	-	87.0	85.4±9.1	86.3±14.8
Procgen-Maze-1000-Test	12.0	14.5	60	69.8	-	-	-	78.2	78.8±6.2	87.5±10.8
Average	14.4	13.3	60.7	66.5	-	-	-	75.1	79.0	82.6
CALVIN	17.3	3.1	7.8	12.4	-	-	-	43.8	50.5±20.9	63.9±27.9
Average	17.3	3.1	7.8	12.4	-	-	-	43.8	50.5	63.9

of elastic subgoals. Additionally, the strong representational capability of the diffusion model has also had a significant impact on the accurate representation of the subgoals. Therefore, we conduct a detailed analysis of the contributions of the diffusion model and the elastic subgoal in Section 5.2 and Section 5.4, respectively.

5.2 Does the Diffusion Model Outperform the Unimodal Model as a Policy model?

To demonstrate the importance of using the diffusion model as a policy model, we also test a version of ESD with the Gaussian policy model, called ESD-Gauss (details in supplementary Appendix **??**). As shown in Table 1, we observe that on average across all six environments, ESD outperforms ESD-Gauss. Notably, in the challenging CALVIN and AntSoccer environments, the diffusion model provides a substantial improvement over the unimodal Gaussian model. In sophisticated image-based environments (Procgen), the diffusion model also shows a significant advantage. This indicates that the diffusion model is a better choice for representing policies with multimodal distributions.

5.3 Can ESD Better Mitigate Policy Errors Caused by Noisy Value Functions?

In general, when the dataset contains less data and has lower coverage, the discrepancy between the learned value function and the optimal value function becomes larger, resulting in greater noise in the value function. Therefore, we train multiple algorithms on datasets with varying sizes to assess the ability of ESD to resist noise in the value function. As shown in Fig 6, ESD performs relatively stable across datasets with different sizes. In contrast, HIQL exhibits a noticeable decline on the AntMaze-Umaze-Diverse. In the AntMaze-Medium-Diverse, HIQL is more sensitive to the dataset size than ESD, showing greater fluctuations. In the AntMaze-Large-Diverse, both ESD and HIQL experience a significant drop in performance when the dataset size decreases to around $0.2M$. However, before the dataset size drops to $0.2M$, HIQL's performance decline is more

pronounced than that of ESD. Therefore, ESD is more effective in mitigating the impact of noise than the other algorithms.

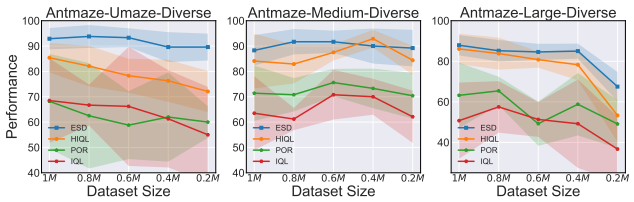

Figure 6: Analysis of performance under varying dataset sizes. As the dataset size decreases, the performance drop of ESD is relatively smaller compared to other algorithms.

5.4 Ablation Study

Next, we conduct an ablation study to verify whether the advantage-weighted noise objective can better train the policy model based on diffusion model, compared to the classifier-guided method (Table 2). We use both the Q-function and the advantage-weighted function $exp(\beta \cdot A)$ as classifiers to guide the sampling process of the diffusion model under the classifier-guided method, resulting in *ESD-QClf* and *ESD-AClf*, respectively. From Table 2, it can be observed that the model trained using the classifier-guided method performs worse than the model trained with the advantage-weighted noise objective. Moreover, the performance of ESD-QClf and ESD-AClf is roughly the same as *ESD-BC*, which does not apply any guidance to the diffusion policy model (i.e., employing imitation learning in both the high-level and low-level policies based on elastic subgoals). Therefore, we conjecture that advantage-weighted noise objective can better train the policy model based on diffusion models.

Table 2: Ablation study of advantage-weighted noise objective. Due to the impact of OOD problems, the performance of ESD-QClf and ESD-AClf is lower than ESD, which is trained by the advantage-weighted noise objective.

Dataset	ESD-BC	ESD-QClf	ESD-AClf	ESD
AntMaze-Umaze	73.3±9.0	90.6±4.9	92.7±3.9	97.1±2.6
AntMaze-Medium-Play	17.4±9.0	71.3±5.4	84.0±3.9	90.8±6.4
AntMaze-Large-Play	18.7±7.5	51.3±9.8	80±10.5	88.8±6.0
AntMaze-Ultra-Play	32.0±4.5	40.7±9.0	61.3±12.7	56.7±9.1
Average	35.4	63.5	79.5	83.4
Kitchen-Partial	57.2±6.3	53.8±8.7	59.4±10.6	69.8±2.1
Kitchen-Mixed	59.3±6.5	49.0±11.4	57.1±7.1	67.1±5.0
Average	58.3	51.4	58.25	68.5

Moreover, to further demonstrate the advantages of elastic subgoals compared to inelastic subgoals, we compared ESD with two inelastic-subgoal hierarchical policies, HIQL and POR, as shown in Table 3. These two algorithms utilize fixed lengths of $m(>1)$ and 1 for inelastic subgoal selection, respectively. For fairness, we used diffusion models as the basis for the policy models, employing the same network architecture across all comparisons. The results show that elastic subgoals consistently exhibit significant advantages across various environments. Moreover, when using a Gaussian model as the policy model, ESD-Gauss also demonstrates significant advantages compared to these algorithms (see Table 1). This indicates that elastic subgoals are better at mitigating the impact of noise in the value function, thereby enabling the learning of better policies.

Table 3: Ablation study of elastic subgoal under diffusion policy model. Elastic subgoal (ESD) significantly improves performance in both the Antmaze and CALVIN environments. ○ represents non-hierarchical policy, △ represents inelastic-subgoal hierarchical policy, ★ represents elastic-subgoal hierarchical policy.

Dataset	GC-IQL-D$^{\circ}$	GC-POR-D$^{\triangle}$	HIQL-D$^{\triangle}$	ESD$^{\bigstar}$
AntMaze-Umaze	89.5±5.8	93.3±6.5	88.4±8.3	97.1±2.6
AntMaze-Medium-Play	73.3±13.2	79.2±12.6	85.6±4.2	90.8±6.4
AntMaze-Large-Play	44.3±17.5	56.7±22.8	86.7±5.6	88.8±6.0
AntMaze-Ultra-Play	28.6±13.7	36.7±22.4	57.5±14.9	56.7±9.1
Average	58.9	66.5	79.6	83.4
CALVIN	18.9±16.6	9.3±9.1	46.2±16.7	63.9±27.9
Average	18.9	9.3	46.2	63.9
Scene-Play	49.5±9.8	41.7±3.7	57.1±10.3	64.2±8.9
Scene-Noisy	45.0±6.4	36.7±8.3	31.3±10.3	47.1±7.7
Average	47.3	39.2	44.2	55.7

We refer to supplementary Appendix **??** for further experimental analyses, including subgoal visualizations, comparisons of the number of parameters and inference times, and an ablation study on the maximum subgoal time steps.

6 CONCLUSION

We propose ESD as an effective hierarchical algorithm for offline GCRL. We introduce the idea of elastic subgoal in the high-level policy to decompose goals into subgoals, which is better at suppressing the impact of noise compared to inelastic approaches. In addition, to handle the multimodal distribution of subgoals and actions, we use the diffusion model to learn such multimodal policies and employ an implicit guidance method to guide the denoising process. Experimental results show that ESD achieves competitive performance and outperforms other algorithms on various benchmarks. Since ESD employs a diffusion model with powerful representation capacity as the policy model, it is suitable for scaling to large, real-world applications with multimodal data. Future work can apply ESD to larger architectures and explore learning policies on large-scale reward-free datasets, such as driving data and internet videos.

ACKNOWLEDGMENTS

This work was supported in part by Beijing Natural Science Foundation under Grant 4232056, in part by National Natural Science Foundation of China under Grants 62293541 and 62136008, and in part by Beijing Nova Program under Grant 20240484514.

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in neural information processing systems* 30 (2017), 12623 – 12634.

[2] Yuhui Chen, Haoran Li, and Dongbin Zhao. 2024. Boosting Continuous Control with Consistency Policy. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, Auckland, New Zealand, 335–344.

[3] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems* 34 (2021), 8780–8794.

[4] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. 2019. Goal-conditioned Imitation Learning. *Advances in Neural Information Processing Systems* 32 (2019), 15298–15309.

[5] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. 2022. RVS: WHAT IS ESSENTIAL FOR OFFLINE RL VIA SUPERVISED LEARNING?. In *The 10th International Conference on Learning Representations*. OpenReview.net, Virtual Conference, 1–14.

[6] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. 2022. Contrastive Learning as Goal-Conditioned Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (2022), 35603 – 35620.

[7] Xing Fang, Qichao Zhang, Yinfeng Gao, and Dongbin Zhao. 2022. Offline reinforcement learning for autonomous driving with real world driving data. In *The 25th IEEE Conference on Intelligent Transportation Systems*. IEEE, Macau, China, 1–7.

[8] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International conference on machine learning*. PMLR, Long Beach, California, 2052–2062.

[9] Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. 2023. Reinforcement learning from passive data via latent intentions. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Honolulu, Hawaii, 11321–11339.

[10] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. 2019. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In *The 7th International Conference on Learning Representations*. OpenReview.net, New Orleans,Louisiana, 1–13.

[11] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. 2023. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573* (2023).

[12] Joey Hejna, Jensen Gao, and Dorsa Sadigh. 2023. Distance Weighted Supervised Learning for Offline Interaction Data. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Honolulu, Hawaii, 12882–12906.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

[14] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, Baltimore, Maryland, 9902–9915.

[15] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems* 34 (2021), 1273–1286.

[16] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *The 10th International Conference on Learning Representations*. OpenReview.net, Virtual Conference, 1–11.

[17] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems* 29 (2016), 3682 – 3690.

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

[19] Haoran Li, Zhennan Jiang, Yuhui Chen, and Dongbin Zhao. 2024. Generalizing consistency policy to visual RL with prioritized proximal experience regularization. *Advances in Neural Information Processing Systems* 38 (2024), 1 – 29.

[20] Haoran Li, Yaocheng Zhang, Haowei Wen, Yunaheng Zhu, and Dongbin Zhao. 2024. Stabilizing diffusion model for robotic control with dynamic programming and transition feasibility. *IEEE Transactions on Artificial Intelligence* 5 (2024), 4585–4594.

[21] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning.. In *International Conference on Learning Representations*. Conference Track Proceedings, San Juan, Puerto Rico, 1–14.

[22] Xueyi Liu, Qichao Zhang, Yinfeng Gao, and Zhongpu Xia. 2023. PnP: Integrated Prediction and Planning for Interactive Lane Change in Dense Traffic. In *The 30th International Conference on Neural Information Processing*. Springer-Verlag, Berlin, Heidelberg, 303–316.

[23] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2019. Learning Latent Plans from Play. In *Proceedings of the 3th Conference on Robot Learning*. PMLR, Osaka, Japan, 1–10.

[24] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. 2023. HIQL: Offline Goal-Conditioned RL with Latent States as Actions. *Advances in Neural Information Processing Systems* 36 (2023), 34866–34891.

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

[26] Jan Peters and Stefan Schaal. 2007. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*. PMLR, Helsinki, Finland, 745–750.

[27] Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems* 32 (2019), 11895–11907.

[28] Zhentao Tang, Yuanheng Zhu, Dongbin Zhao, and Simon Lucas. 2023. Enhanced rolling horizon evolution algorithm with opponent model learning: results for the fighting game AI competition. *IEEE Transactions on Games* 1.5 (2023), 5–16.

[29] Songjun Tu, Jingbo Sun, Qichao Zhang, Yaocheng Zhang, Jia Liu, Ke Chen, and Dongbin Zhao. 2024. In-Dataset Trajectory Return Regularization for Offline Preference-based Reinforcement Learning. *arXiv preprint arXiv:2412.09104* (2024).

[30] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International conference on machine learning*. PMLR, Stockholm, Sweden, 3540–3549.

[31] Mingdong Wu, Fangwei Zhong, Yulong Xia, and Hao Dong. 2022. Targf: Learning target gradient field to rearrange objects without explicit goal specification. *Advances in Neural Information Processing Systems* 35 (2022), 31986–31999.

[32] Haoran Xu, Li Jiang, Li Jianxiong, and Xianyuan Zhan. 2022. A policy-guided imitation approach for offline reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 4085–4098.

[33] Qichao Zhang, Xing Fang, Kaixuan Xu, Haoran Li, and Dongbin Zhao. 2024. High-quality synthetic data is efficient for model-based offline reinforcement learning. In *2024 International Joint Conference on Neural Networks*. IEEE, Yokohama, Japan, 1–7.

[34] Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. 2024. Safe Offline Reinforcement Learning with Feasibility-Guided Diffusion Model. In *The 12th International Conference on Learning Representations*. OpenReview.net, Vienna Austr, 1–30.

[35] Yuanheng Zhu, Shangjing Huang, Binbin Zuo, Dongbin Zhao, and Changyin Sun. 2024. Multi-Task Multi-Agent Reinforcement Learning With Task-Entity Transformers and Value Decomposition Training. *IEEE Transactions on Automation Science and Engineering* to be published (2024), 1–14.