GAS: ENHANCING REWARD-COST BALANCE OF GENERATIVE MODEL-ASSISTED OFFLINE SAFE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline Safe Reinforcement Learning (OSRL) aims to learn a policy that achieves high performance in sequential decision-making while satisfying safety constraints, using only pre-collected datasets. Recent works, inspired by the strong capabilities of Generative Models (GMs), reformulate decision-making in OSRL as a conditional generative process, where GMs generate desirable actions conditioned on predefined reward and cost return-to-go values. However, GM-assisted methods face two major challenges in constrained settings: (1) they lack the ability to "stitch" optimal transitions from suboptimal trajectories within the dataset, and (2) they struggle to balance reward maximization with constraint satisfaction, particularly when tested with imbalanced human-specified reward-cost conditions. To address these issues, we propose Goal-Assisted Stitching (GAS), a novel algorithm designed to enhance stitching capabilities while effectively balancing reward maximization and constraint satisfaction. To enhance the stitching ability, GAS first augments and relabels the dataset at the transition level, enabling the construction of high-quality trajectories from suboptimal ones. GAS also introduces novel goal functions, which estimate the optimal achievable reward and cost goals from the dataset. These goal functions, trained using expectile regression on the relabeled and augmented dataset, allow GAS to accommodate a broader range of reward-cost return pairs and achieve a better tradeoff between reward maximization and constraint satisfaction compared to human-specified values. The estimated goals then guide policy training, ensuring robust performance under constrained settings. Furthermore, to improve training stability and efficiency, we reshape the dataset to achieve a more uniform reward-cost return distribution. Empirical results validate the effectiveness of GAS, demonstrating superior performance in balancing reward maximization and constraint satisfaction compared to existing methods.

1 INTRODUCTION

Significant progress has been achieved in Reinforcement Learning (RL) that learns policies to maximize rewards through constant interactions with the environment. Limited by the trial-and-error approach, standard RL often fails in scenarios with safety constraints, such as autonomous driving (Fang et al., 2022) and investment portfolios (Lee & Moon, 2023). It stems from two issues: (1) the exploration process in RL can be inherently risky, as random actions may lead to unsafe outcomes, and (2) the optimized policies often focus solely on maximizing cumulative rewards, neglecting safety constraints. To tackle these issues, Offline Safe RL (OSRL) has emerged as a promising paradigm that learns safe policies from a pre-collected dataset, eliminating the need for risky online exploration.

While OSRL mitigates the risks associated with online exploration by relying on static datasets, it introduces a new challenge: the Out-Of-Distribution (OOD) problem. Specifically, the Bellman backup may extrapolate actions beyond the dataset, leading to unpredictable or unsafe behaviors. Early OSRL methods primarily adapted RL techniques to constrained settings using approaches such as the Lagrange method (Stooke et al., 2020), constraint penalty (Xu et al., 2022), or the DICE-style techniques (Lee et al., 2022). To mitigate OOD-related issues, these methods incorporate strategies like distribution correction (Lee et al., 2022), regularization (Kostrikov et al., 2021), and OOD detection (Xu et al., 2022). However, they still face challenges in effectively addressing OOD problems and adapting to dynamic, real-world constraints.

More recently, Generative Model-assisted (GM) methods have emerged as an alternative to traditional OSRL approaches to address these limitations. In OSRL, GM methods reformulate the Constrained

Markov Decision Process (CMDP) as a goal-conditioned generating problem, where the generative model is trained to produce trajectories that match predefined reward and cost returns specified as inputs. This formulation provides two key benefits. First, GM methods essentially adopt a goal-conditioned behavior cloning scheme, where the model learns to imitate behavior from the dataset conditioned on the desired reward and cost targets. This formulation entirely bypasses the Bellman backup procedure, which is the primary source of the OOD problem in traditional OSRL approaches. Second, GM methods also offer greater flexibility compared to conventional methods, which typically operate under fixed safety constraints. Because the target reward and cost returns are provided as inputs, GM methods can seamlessly adapt to varying objectives at test time without re-training.

Nevertheless, GM methods present their own challenges. First, while GM methods bypass the Bellman backup procedure, they lack trajectory stitching capabilities (Badrinath et al., 2023; Wu et al., 2023; Kim et al., 2024b) — the ability to combine transitions from different trajectories to enhance performance. This limitation restricts their ability to fully utilize suboptimal datasets to improve performance. Second, GM methods lack an explicit mechanism to balance reward maximization and constraint satisfaction, potentially leading to unsafe or overly conservative policies. In this work, we propose a novel *Goal-Assisted Stitching (GAS)* method to address these challenges while retaining the advantages of GM methods. The key contributions are summarized as follows.

We propose to use goal functions as intermediate values to bridge the gap between human-specified (potentially suboptimal) reward-cost targets and the optimal achievable goals of the conditional policy instantiated by GM methods. To achieve this, we introduce three key innovations: **1) Temporal Segmented Return Augmentation and Transition-level Return Relabeling:** To enhance GAS's stitching capabilities, we restructure the offline dataset at the transition level and introduce temporal segmented return augmentation, which extracts richer information by considering reward and cost returns over varying timesteps rather than only at trajectory endpoints. Additionally, we ensure robustness to suboptimal human-specified target return-to-goes during testing by relabeling reward-cost return-to-goes at the transition level during training. **2) Goal Functions with Expectile Regressions:** We train the novel reward and cost goal functions using expectile regression to estimate the optimal achievable reward and cost goals without relying on Bellman backups. These goal functions guide the policy to stitch transitions effectively, achieving both reward maximization and constraint satisfaction for a wider range of given target reward-cost return pairs. **3) Dataset Reshaping:** To improve training stability and efficiency, we address the data imbalance issue by reshaping the dataset to create a more balanced reward-cost return distribution. **Through extensive experiments on 2 benchmarks with 12 scenarios and 8 baselines under various constraint thresholds, GAS shows superior safety ability under tight thresholds and 6% improvement in performance under loose thresholds.**

2 Related Work

RL Methods in Offline RL. Offline RL (Fujimoto et al., 2019) aims to find the policy to maximize the cumulative rewards from a pre-collected dataset. The primary challenge lies in the OOD problem during the Bellman backup, where the policy may select actions beyond the dataset. This issue arises because offline RL does not allow for environment exploration to gather additional data. To this end, most existing works try to constrain the target policy to stay close to the behavior policy with a KL regularization term (Jaques et al., 2020; Peng et al., 2019; Siegel et al., 2020) or Wasserstein distance (Wu et al., 2020) where the behavior policy is estimated by a generative model. For example, BCQ (Fujimoto et al., 2019) uses a variational autoencoder (VAE) to estimate the behavior policy and restrict the action space during the Bellman backup. Except for explicitly estimating the behavior policy, CQL (Kumar et al., 2020) proposes to learn Q-values conservatively, encouraging those within the dataset to act as a lower bound. Alternatively, IQL (Kostrikov et al., 2022) avoids the OOD problem entirely by employing expectile regression to learn Q-values without querying OOD actions.

GM Methods in Offline RL. More recently, some works formulate offline RL as a return-conditioned generation problem and address it using generative models, such as Decision Transformer (DT) (Wu et al., 2023; Chen et al., 2021; Zheng et al., 2022), and Decision Convformer (DC) (Kim et al., 2024a). These methods naturally avoid the OOD problem since the return-conditioned generation problem does not need the Bellman backup procedure. However, a significant limitation of these methods is their low stitching ability—i.e., the ability to identify and combine optimal transitions from suboptimal trajectories—particularly when the dataset consists primarily of suboptimal trajectories. To improve the stitching ability of GM methods, several works have been proposed. For instance, QDT (Yamagata et al., 2023a) relabels return-to-go in the dataset with Q-values derived from RL

methods. WT (Badrinath et al., 2023) proposes to use a sub-goal as the prompt to guide the DT policy to find a shorter path in navigation problems. Building on these ideas, ADT (Ma et al., 2024) proposes a hierarchical framework by replacing the return-to-go with a sub-goal learned from IQL. In contrast to improving stitching ability in the training procedure, EDT (Wu et al., 2023) boosts the stitching ability of DT at decision time by adaptively adjusting the history length of the attention module.

Offline Safe RL. OSRL integrates safety constraints into offline learning settings, addressing both safety requirements and limited online interaction with the environment. This emerging field has spawned two main approaches: RL methods and GM methods. For RL methods, CPQ (Xu et al., 2022) employs a conditional VAE model to estimate and penalize the OOD actions. COptiDICE (Lee et al., 2022) utilizes stationary distribution correction to mitigate the distributional shift problem. For GM methods, which are inspired by DT, CDT (Liu et al., 2023b) transforms OSRL into a goal-conditioned generative problem and inputs both target reward and cost return-to-go to the GPT structure. To handle the safety-critical cases, FISOR (Zheng et al., 2024) proposes a feasibility-guided diffusion model to ensure strict satisfaction of constraints.

Although previous works demonstrate great success in enhancing the stitching ability of GM methods in offline RL, they cannot be utilized in OSRL directly due to the fundamentally different objectives introduced by constrained settings. Extending GM methods to OSRL is challenging because, while the goal of maximizing rewards remains consistent, the requirements for constraints vary across scenarios. Our work addresses this critical gap by introducing GAS, a novel framework designed to enhance stitching capabilities while achieving a robust balance between reward maximization and constraint satisfaction under various constrained scenarios.

3 BACKGROUND

Constrained Markov Decision Process (CMDP). CMDP (Altman, 1998) is a standard framework for safe RL defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, c, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s,a)$ is the transition function, $r(s,a)$ is the reward function, $c(s,a)$ is the cost function, and γ is the discount factor. The goal of safe RL is to find a policy $\pi(a|s)$ that maximizes the cumulative rewards while satisfying the safety constraints. Given the constraint threshold L and the trajectory $\tau = \{(s_0, a_0, r_0, c_0), ..., (s_T, a_T, r_T, c_T)|a_t = \pi(a_t|s_t)\}$, with T being the trajectory length, the optimization problem of safe RL can be written as:

$$\begin{aligned} &\max_{\pi} \ V_r^{\pi}(\tau), \\ &s.t. \ V_c^{\pi}(\tau) \leq L, \end{aligned} \tag{1}$$

where $V_r^{\pi}(\tau) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r_t]$ denotes the reward value function, and $V_c^{\pi}(\tau) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t c_t]$ denotes the cost value function.

Offline Safe Reinforcement Learning (OSRL). OSRL maintains the same objectives as safe RL while additionally addressing the OOD problem due to the inability to explore. To mitigate the OOD problem, the target policy is constrained to remain close to the behavior policy, as formulated in eq. (2), which augments eq. (1) in the optimization.

$$\mathbb{D}(\pi, \mu) \leq \zeta, \tag{2}$$

where μ is the behavior policy used in the pre-collected dataset $\mathcal{D}$, $\mathbb{D}(.,.)$ is an arbitrary distance/divergence function, and ζ is a hyper-parameter.

Constrained Decision Transformer (CDT). CDT (Liu et al., 2023b) transfers the OSRL problem into a goal-conditioned generative problem by reformulating the dataset as:

$$\tau = (s_0, a_0, R_0, C_0, ..., s_T, a_T, R_T, C_T), \tag{3}$$

where $R_t = r_t + ... + r_T$ is the cumulative rewards from t to the trajectory end in the dataset, named reward return-to-go, and $C_t = c_t + ... + c_T$ is the cost return-to-go. Then CDT trains a policy π with the GPT structure, as shown in eq. (4), to predict the action given (s, R, C) at the current timestep and (s, a, R, C) from previous $K-1$ timesteps as conditions, where K is the memory length of GPT.

$$\pi(a_t|s_t, R_t, C_t, ..., a_{t-K+1}, s_{t-K+1}, R_{t-K+1}, C_{t-K+1}) \quad \text{a.k.a} \quad \pi(a_t|s_t, R_t, C_t, K), \tag{4}$$

During testing, users need to specify the target reward return $\hat{R}$ and cost return $\hat{C}$ as the input of the CDT policy, which then generates actions to approach these targets $\hat{R}$ and $\hat{C}$.

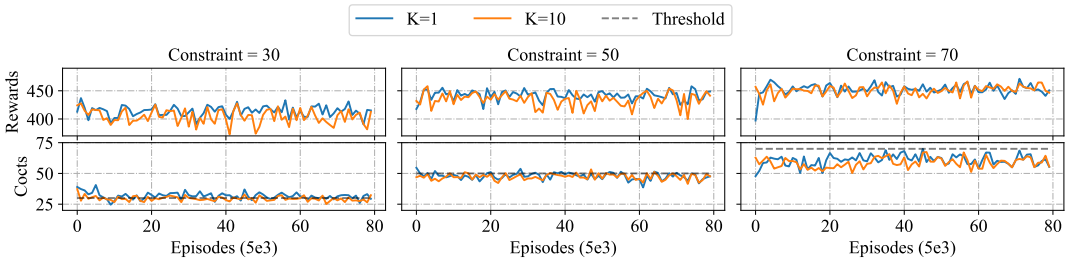

Figure 1: Training curve of CDT with different memory length K on task *CarCircle*.

4 MOTIVATION

In this section, we first analyze two critical limitations of current GM methods for OSRL: (1) insufficient trajectory stitching capabilities and (2) the inability to balance reward maximization and constraint satisfaction. We design our solution to specifically address these challenges.

4.1 INSUFFICIENT TRAJECTORY STITCHING CAPABILITIES

A key strength of traditional RL methods lies in their ability to stitch together different suboptimal trajectories, enabling effective generalization across diverse experiences. This stitching ability arises from the fact that RL only considers the current state as the condition and stitches the next optimal transition via Bellman backup. In contrast, GM methods sacrifice this crucial ability by additionally taking previous information as conditions to capture the temporal association among transitions. Recent studies (Badrinath et al., 2023; Ma et al., 2024; Xiao et al., 2023; Yamagata et al., 2023b) demonstrate that DT/CDT's trajectory-level training paradigm is essentially a form of goal-conditioned behavior cloning, which predicts the action based on information in previous timesteps via the temporal attention module. Given suboptimal information as the contextual condition, GM methods tend to memorize the suboptimal actions, resulting in suboptimal performance.

However, Kim et al. (2024b) reveals that the attention module designed for natural language processing often fails to characterize the temporal relations for sequential transitions of MDPs in DT (Chen et al., 2021). Specifically, while MDP policies should primarily focus on immediate previous states, DT's attention spans up to 20 timesteps backward, potentially diluting the focus on relevant temporal information and adversely affecting the performance of offline RL (Kim et al., 2024b). To validate this observation, we conducted empirical experiments comparing CDT variants with different memory lengths ($K = 1$ and $K = 10$) for attention modules while keeping other parameters constant. As shown in fig. 1, CDT's performance remains largely unchanged across these memory settings under three distinct constraints, suggesting that the attention mechanism fails to effectively leverage temporal information in OSRL.

4.2 INABILITY TO BALANCE REWARD MAXIMIZATION AND CONSTRAINT SATISFACTION

In GM-assisted OSRL, as opposed to goal-conditioned behavior cloning, the objective should be formulated as:

$$\begin{aligned} \max_{\pi} \ & \mathbb{R}(s_t, \pi(a_t|s_t, R_t, C_t, K)) \\ s.t. \ & \mathbb{C}(s_t, \pi(a_t|s_t, R_t, C_t, K)) \leq C_t, \\ & R_t = \hat{R} - (r_0 + ... + r_{t-1}), \\ & C_t = \hat{C} - (c_0 + ... + c_{t-1}), \end{aligned} \tag{5}$$

where $\mathbb{R}(.)$ and $\mathbb{C}(.)$ are the non-discounted reward and cost returns under the policy π. DT-family algorithms, due to their goal-conditioning nature (detailed in section 3), struggle to effectively balance the dual objectives of reward maximization and constraint satisfaction in OSRL. This limitation stems from two key challenges: **First**, without prior knowledge, it becomes problematic to determine appropriate target reward-cost pairs $(\hat{R}, \hat{C})$ as these objectives often require careful trade-offs. The GPT structure can only search for policies that satisfy given targets (Liu et al., 2023b) without validating their feasibility. Consequently, specifying overly ambitious return targets $\hat{R}$ alongside stringent constraints $\hat{C}$ may lead to policy degradation when such conditions prove unrealistic.

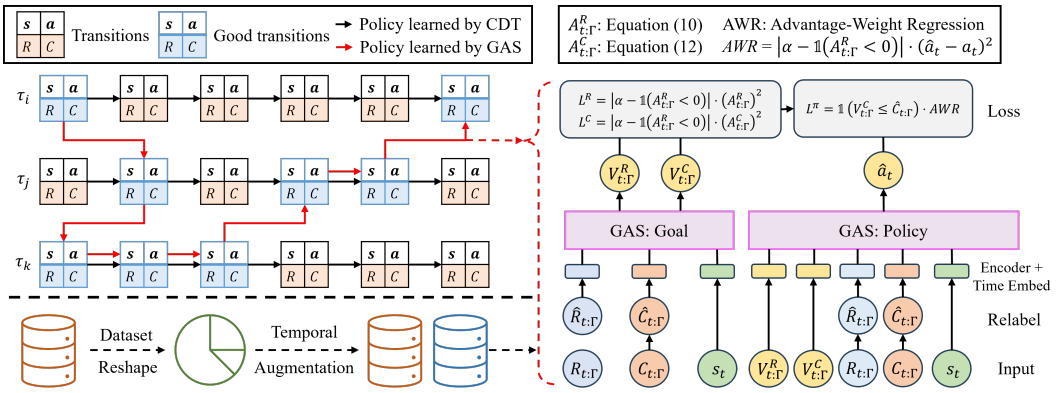

Figure 2: Overall view of GAS. Left: Comparison between CDT and GAS. CDT optimizes at a trajectory level, while GAS enables fine-grained trajectory stitching under the guidance of reward maximization and constraint satisfaction. Right: GAS's stitching mechanism, where the goal function learns the optimal reward and cost return-to-go within the constraint given any target. The policy aims to take actions to achieve optimal goals estimated by the goal function via constrained AWR.

Second, the current CDT architecture simply concatenates target return $\hat{R}$ and cost $\hat{C}$ as contextual conditions without any mechanism to balance their relative importance. This structural limitation prevents GM methods from properly prioritizing constraint satisfaction over reward maximization—a crucial requirement in OSRL. This misalignment between architectural design and OSRL objectives frequently causes performance degradation in practice.

5 Goal-Assisted Stitching

In section 4.1, we highlight that extending the temporal attention module to overly long time windows does not enhance memorization and instead hinders the stitching ability for GM methods in OSRL. This inspires us to trade the temporal attention for a more focused approach: stitching transitions in the dataset under the guidance of reward maximization and constraint satisfaction in a conditional manner. To achieve this goal, we propose Goal-Assisted Stitching (GAS), a novel algorithm that offers a more flexible balance between R_t and C_t and stitches high-quality transitions to achieve safe and best performance as shown in fig. 2. In particular, we first estimate the optimal achievable reward return-to-go satisfying the constraint and corresponding cost return-to-go in the dataset for a given pair of R_t, C_t with a reward-cost goal function through expectile regression without relying on the Bellman backup procedure. The estimated optimal reward and cost goals are then used to guide the policy optimization through a constrained policy optimization paradigm. To further enhance the stability and efficiency of the training procedure, we reshape the dataset in terms of R_t, C_t distribution, ensuring a more uniform reward-cost-return distribution to support GAS training. **Theoretical analysis and the pseudocode of GAS are presented in section B and section C.**

5.1 Optimal Achievable Goals

Conceptually, the maximum achievable return-to-go V^R_t and the corresponding cost return-to-go V^C_t for a given state s and the target reward/cost returns $\hat{R}$ and $\hat{C}$ in the dataset should follow:

$$V^R_t(s, \hat{R}, \hat{C}) = \max_{(s_t=s, a_t, R_t, C_t)\sim\mathcal{D}} R_t * \mathbb{1}(C_t \le \hat{C}). \tag{6}$$

$$V^C_t(s, \hat{R}, \hat{C}) = \arg_{C_t}\{\max_{(s_t=s, a_t, R_t, C_t)\sim\mathcal{D}} R_t * \mathbb{1}(C_t \le \hat{C})\}. \tag{7}$$

However, directly applying eqs. (6) and (7) to estimate the optimal goals raises three issues. **First**, the standard definitions of R_t and C_t are the cumulative rewards and costs from the current timestep to the trajectory end. However, good transitions often occur within shorter segments. This motivates our introduction of $R_{t:\Gamma}$ and $C_{t:\Gamma}$ in section 5.2, representing cumulative values over a shorter window from t to Γ. **Second**, in the training stage, the target reward/cost returns are derived from cumulative values along the trajectory, which may not consistently align with the predefined targets in the testing stage. To address this problem, we propose a transition-level return relabeling in the training stage in section 5.3. **Third**, rare "lucky" transitions with high rewards and low costs can lead to value

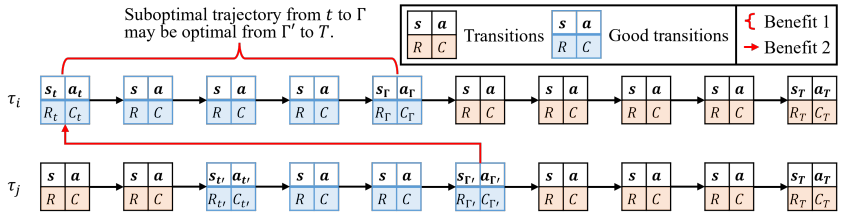

Figure 3: Necessity of temporal segmented return augmentation.

overestimation for maximum achievable goals due to transition function stochasticity. To address this problem, we design a goal function with expectile regression to estimate the expectation of the upper quantile of the return-to-go distribution in section 5.4.

5.2 Temporal Segmented Return Augmentation

Processing the dataset in smaller temporal segments provides GM training with more abundant information. Specifically, we leverage such insights and augment the return information in the dataset according to:

$$(s_t, a_t, R_t, C_t) \rightarrow \{(s_t, a_t, R_{t:\Gamma}, C_{t:\Gamma}) | \Gamma = t, ..., T\} = \{(s_{t'}, a_{t'}, R_{t':T}, C_{t':T}) | t' = t + T - \Gamma\},$$

where $R_{t:\Gamma} = r_t + ... + r_\Gamma$ and $C_{t:\Gamma} = c_t + ... + c_\Gamma$. When sampling $(s_{t'}, a_{t'}, R_{t'}, C_{t'})$, GAS can seek better $R_{t:\Gamma} > R_{t'}$ & $C_{t:\Gamma} \leq C_{t'}$ from other transitions under augmentation as long as $\Gamma - t = T - t'$. As illustrated by fig. 3, this data augmentation scheme provides two benefits: (1) It substantially expands the training data by including transitions of varying temporal lengths; (2) It enables more flexible transition stitching across different timesteps by providing diverse time intervals.

5.3 Transition-level Return Relabeling

First observed in Liu et al. (2023b), the misalignment between human-specified reward-cost targets in the testing stage and the training inputs can lead to degraded performance for GM methods due to their nature of behavior cloning. Inspired by the trajectory-level labeling in Liu et al. (2023b), we propose a more fine-grained, transition-level return relabeling mechanism fitting in with transition-level stitching. For sampled transitions $\hat{\mathcal{D}} = \{(s_t, a_t, R_{t:\Gamma}, C_{t:\Gamma}, t' = t + T - \Gamma)\}$, we relabel $R_{t:\Gamma}$ and $C_{t:\Gamma}$ in eq. (8) and take relabeled values as input of goal functions $V^R_{t:\Gamma}(s_t, a_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t')$.

$$\begin{aligned} \hat{R}_{t:\Gamma} &= U((1-\delta)R_{t:\Gamma}, (1+\delta)R_{t:\Gamma}), \\ \hat{C}_{t:\Gamma} &= U(C_{t:\Gamma}, C^{\max}), \end{aligned} \tag{8}$$

where $U(a, b)$ is a uniform distribution between a and b, $\delta \in (0, 1)$ is a hyper-parameter and $C^{\max}$ is the maximum value of cost returns. In this way, goal functions can be trained under more imbalanced and comprehensive reward-cost targets. Notably, our proposed GAS does not directly update the policy guided by the relabeled values. Instead, we utilize them to assist training through intermediate optimal goals and update the policy based on these goals during optimization. As a result, GAS retains the robustness of the policy without affecting reward maximization and constraint satisfaction.

5.4 Goal Functions with Expectile Regressions

Since naively taking the maximum operator in the dataset can be prone to rare "lucky" samples, we adopt a distributional perspective and optimize goal functions that focus more on high return-to-go samples and less on low return-to-go samples. To this end, we employ expectile regression for iteratively updating the estimated goal functions.

The reward goal function should output the largest reward-to-go that satisfies the constraint. To formalize this, we first define the advantage function as:

$$A^R_{t:\Gamma} = \mathbb{1}(V^C_{t:\Gamma} < \hat{C}_{t:\Gamma}) \cdot R_{t:\Gamma} - V^R_{t:\Gamma}(s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t' = t + T - \Gamma), \tag{9}$$

where $\mathbb{1}(V^C_{t:\Gamma} < \hat{C}_{t:\Gamma})$ is an indicator function of constraint satisfaction.

In this way, transitions that violate the constraint or have low return $R_{t:\Gamma}$ are down-weighted during expectile regression. Then the loss function with expectile regression can be defined as :

$$L_R = \mathbb{E}_{\hat{\mathcal{D}}}[|\alpha - \mathbb{1}(A^R_{t:\Gamma} < 0)| \cdot (A^R_{t:\Gamma})^2]. \tag{10}$$

With this loss function, the reward goal function ($V^R_{t:\Gamma}$) converges to the expectile of the largest reward return-to-go that satisfies the constraint controlled by α.

The optimization objective for the cost goal function differs from that of the reward goal function. While the reward goal function aims to find the largest reward return-to-go in the dataset via expectile regression, the cost goal function seeks to estimate the cost value associated with the optimal reward goal. To address this, we modify the loss function of the cost goal function to eqs. (11) and (12), assigning higher weights to transitions with higher reward goals under the constraint.

$$A^C_{t:\Gamma} = C_{t:\Gamma} - V^C_{t:\Gamma}(s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t' = t + T - \Gamma). \tag{11}$$

$$L_C = \mathbb{E}_{\hat{\mathcal{D}}}[|\alpha - \mathbb{1}(A^R_{t:\Gamma} < 0)| \cdot (A^C_{t:\Gamma})^2]. \tag{12}$$

Goal-guided Policy Optimization: To ensure that the policy comprehends optimal targets under relabeled targets returns, we incorporate the optimal reward and cost goals obtained from the goal functions as inputs to the policy. Then we optimize the policy utilizing a constrained version of Advantage Weight Regression (AWR), as shown in:

$$L_\pi = \mathbb{E}_{\hat{\mathcal{D}}}[\mathbb{1}(V^C_{t:\Gamma} < \hat{C}_{t:\Gamma}) \cdot |\alpha - \mathbb{1}(A^R_{t:\Gamma} < 0)| \cdot (\pi(a|s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, V^R_{t:\Gamma}, V^C_{t:\Gamma}, t') - a_t)^2]. \tag{13}$$

5.5 Dataset Reshaping

Existing GM methods take reward and cost returns as input, while the reward-cost distribution in the dataset is often highly imbalanced, a problem referred to as data imbalance (Kang et al., 2021; Bagui & Li, 2021; Yang et al., 2021; Ren et al., 2022). The issue seriously affects the ability of RL-based methods on constraint satisfaction (Yao et al., 2024), as well as GM-assisted methods. We define transitions with low costs and low rewards as "**conservative transitions**", transitions with high costs and high rewards as "**aggressive transitions**", and transitions with low costs but high rewards as "**ideal transitions**". As shown in fig. 4, most transitions in the dataset are concentrated in regions where both cost and reward returns are extremely low. When training with uniform sampling, GM methods tend to learn predominantly from these conservative transitions, while under-representing ideal transitions that exhibit higher reward returns with lower cost returns. To mitigate this issue, we propose to reshape the dataset distribution during the training phase. In particular, we first estimate the reward-return distribution conditioned on cost returns from the offline dataset and then select all transitions that fall within the top $q\%$ reward returns for each cost return, thereby creating a new dataset $\mathcal{D}^q = \{(s, a, R, C) \sim \mathcal{D} | P^c(R|C) > 1 - q\}$, where P^c indicates cumulative distribution function. Throughout the training procedure, $\mathcal{D}^q$ will be sampled with a probability ϵ and the original dataset $\mathcal{D}$ will be sampled with a probability $1-\epsilon$. As shown in fig. 4, the reshaped dataset distribution is more balanced compared to the original one.

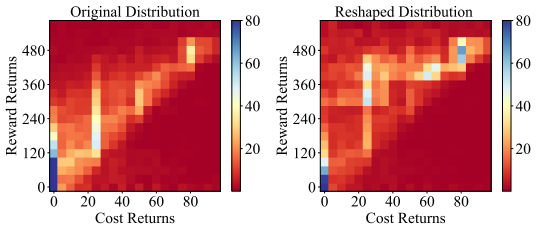

Figure 4: Original and reshaped dataset distribution.

6 Experiment

In this section, we aim to evaluate our proposed GAS and empirically answer three questions: **1)** Can GAS achieve both safe and better performance with improved stitching ability? **2)** Can GAS preserve the zero-shot adaptation ability to different constraint thresholds? **3)** Is GAS robust to imbalanced and human-specified target reward-cost return-to-goes? Accordingly, we design the following experimental setup to evaluate GAS.

Tasks: We evaluate GAS on the widely used **Bullet-Safety-Gym** (Gronauer, 2022) and **Safety-Gymnasium** (Ji et al., 2023) benchmarks. For all tasks, we use the dataset provided in DSRL (Liu et al., 2023a) as our offline dataset, following the D4RL (Fu et al., 2020) benchmark format.

Baselines: We compare our proposed GAS with the following baseline methods: (1) Constraint penalized method: CPQ (Xu et al., 2022); (2) Distribution correction estimation: COptiDICE (Lee et al., 2022); (3) Variational optimization with conservative estimation: VOCE (Guan et al., 2023); (4) Weighted safe actor-critic: WSAC (Wei et al., 2024); (5) Generative model assisted algorithms: CDT (Liu et al., 2023b) and FISOR (Zheng et al., 2024).

Table 1: Normalized evaluation results. The normalized cost threshold is set to 1. Values shown as "mean$_{\pm \text{std}}$" represent the mean and standard deviation. Each value represents the average performance over 10 evaluation episodes with 5 seeds and 3 thresholds. **Bold**/gray/**blue** indicate **safe**/unsafe/**safe and best-performing** results. ↑ (↓) indicates that higher (lower) values are better.

Methods	CPQ		COptiDICE		WSAC		VOCE		CDT		FISOR		GAS	
Tasks	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓
Tight Constraint Threshold: Average results on thresholds with 10%, 20%, and 30% of the maximium costs for each task.														
AntRun	**0.02**$_{\pm 0.01}$	**0.00**$_{\pm 0.00}$	**0.60**$_{\pm 0.03}$	**0.45**$_{\pm 0.22}$	**0.29**$_{\pm 0.04}$	**0.30**$_{\pm 0.28}$	**0.23**$_{\pm 0.05}$	**0.87**$_{\pm 0.36}$	**0.72**$_{\pm 0.03}$	**0.93**$_{\pm 0.46}$	**0.29**$_{\pm 0.03}$	**0.00**$_{\pm 0.00}$	**0.72**$_{\pm 0.02}$	**0.70**$_{\pm 0.30}$
BallRun	0.32$_{\pm 0.14}$	1.53$_{\pm 1.07}$	0.58$_{\pm 0.01}$	4.54$_{\pm 0.33}$	1.10$_{\pm 0.30}$	7.10$_{\pm 0.94}$	1.08$_{\pm 0.38}$	7.10$_{\pm 0.11}$	0.33$_{\pm 0.01}$	1.16$_{\pm 0.15}$	**0.23**$_{\pm 0.01}$	**0.00**$_{\pm 0.00}$	**0.33**$_{\pm 0.01}$	**0.62**$_{\pm 0.15}$
CarRun	**0.95**$_{\pm 0.14}$	**0.83**$_{\pm 0.15}$	**0.96**$_{\pm 0.03}$	**0.00**$_{\pm 0.00}$	**0.96**$_{\pm 0.09}$	**0.31**$_{\pm 0.22}$	0.95$_{\pm 0.26}$	8.09$_{\pm 0.03}$	**0.99**$_{\pm 0.01}$	**0.99**$_{\pm 0.15}$	**0.82**$_{\pm 0.01}$	**0.00**$_{\pm 0.00}$	**0.99**$_{\pm 0.01}$	**0.19**$_{\pm 0.05}$
DroneRun	0.41$_{\pm 0.21}$	4.47$_{\pm 1.40}$	0.74$_{\pm 0.16}$	3.42$_{\pm 0.27}$	**0.38**$_{\pm 0.12}$	**0.41**$_{\pm 0.13}$	0.67$_{\pm 0.20}$	4.21$_{\pm 1.21}$	0.61$_{\pm 0.01}$	1.01$_{\pm 0.15}$	**0.37**$_{\pm 0.05}$	**0.41**$_{\pm 0.17}$	**0.60**$_{\pm 0.02}$	**0.22**$_{\pm 0.13}$
AntCircle	**0.02**$_{\pm 0.02}$	**0.00**$_{\pm 0.00}$	0.23$_{\pm 0.03}$	2.14$_{\pm 0.24}$	**0.26**$_{\pm 0.08}$	**0.61**$_{\pm 0.33}$	**0.17**$_{\pm 0.06}$	**0.83**$_{\pm 0.24}$	0.54$_{\pm 0.02}$	1.46$_{\pm 0.08}$	**0.13**$_{\pm 0.03}$	**0.00**$_{\pm 0.00}$	**0.52**$_{\pm 0.02}$	**0.96**$_{\pm 0.10}$
BallCircle	**0.66**$_{\pm 0.23}$	**0.61**$_{\pm 0.44}$	0.70$_{\pm 0.02}$	3.53$_{\pm 0.00}$	**0.73**$_{\pm 0.08}$	**0.30**$_{\pm 0.08}$	0.74$_{\pm 0.07}$	1.10$_{\pm 0.33}$	0.73$_{\pm 0.00}$	1.23$_{\pm 0.18}$	**0.28**$_{\pm 0.02}$	**0.20**$_{\pm 0.03}$	**0.71**$_{\pm 0.00}$	**0.84**$_{\pm 0.20}$
CarCircle	0.72$_{\pm 0.02}$	1.22$_{\pm 0.60}$	0.48$_{\pm 0.03}$	2.78$_{\pm 0.34}$	**0.64**$_{\pm 0.14}$	**0.21**$_{\pm 0.12}$	0.66$_{\pm 0.13}$	1.21$_{\pm 0.40}$	**0.75**$_{\pm 0.02}$	**0.97**$_{\pm 0.10}$	**0.24**$_{\pm 0.05}$	**0.00**$_{\pm 0.00}$	**0.70**$_{\pm 0.03}$	**0.84**$_{\pm 0.13}$
DroneCircle	0.05$_{\pm 0.02}$	2.68$_{\pm 1.33}$	0.41$_{\pm 0.02}$	1.24$_{\pm 0.24}$	**0.02**$_{\pm 0.01}$	**0.66**$_{\pm 0.38}$	0.05$_{\pm 0.02}$	1.41$_{\pm 0.43}$	0.69$_{\pm 0.01}$	1.19$_{\pm 0.28}$	**0.49**$_{\pm 0.03}$	**0.00**$_{\pm 0.00}$	**0.68**$_{\pm 0.01}$	**0.96**$_{\pm 0.27}$
Average	0.39$_{\pm 0.10}$	1.42$_{\pm 0.62}$	0.59$_{\pm 0.04}$	2.26$_{\pm 0.21}$	0.55$_{\pm 0.11}$	1.24$_{\pm 0.31}$	0.57$_{\pm 0.15}$	3.10$_{\pm 0.39}$	0.67$_{\pm 0.01}$	1.12$_{\pm 0.19}$	**0.36**$_{\pm 0.03}$	**0.03**$_{\pm 0.01}$	**0.66**$_{\pm 0.02}$	**0.67**$_{\pm 0.17}$
Loose Constraint Threshold: Average results on thresholds with 70%, 80%, and 90% of the maximium costs for each task.														
AntRun	**0.06**$_{\pm 0.02}$	**0.00**$_{\pm 0.00}$	**0.60**$_{\pm 0.02}$	**0.12**$_{\pm 0.11}$	**0.52**$_{\pm 0.09}$	**0.62**$_{\pm 0.14}$	**0.40**$_{\pm 0.04}$	**0.75**$_{\pm 0.13}$	**0.79**$_{\pm 0.03}$	**0.77**$_{\pm 0.02}$	**0.32**$_{\pm 0.04}$	**0.00**$_{\pm 0.00}$	**0.84**$_{\pm 0.03}$	**0.93**$_{\pm 0.05}$
BallRun	1.20$_{\pm 0.00}$	1.45$_{\pm 0.00}$	**0.57**$_{\pm 0.01}$	**0.88**$_{\pm 0.08}$	1.21$_{\pm 0.33}$	1.46$_{\pm 0.38}$	1.14$_{\pm 0.01}$	1.45$_{\pm 0.01}$	**0.72**$_{\pm 0.02}$	**0.94**$_{\pm 0.08}$	**0.24**$_{\pm 0.01}$	**0.00**$_{\pm 0.00}$	**0.76**$_{\pm 0.00}$	**0.96**$_{\pm 0.04}$
CarRun	**0.95**$_{\pm 0.05}$	**0.73**$_{\pm 0.32}$	**0.96**$_{\pm 0.04}$	**0.15**$_{\pm 0.14}$	**0.94**$_{\pm 0.08}$	**0.31**$_{\pm 0.08}$	0.95$_{\pm 0.09}$	2.00$_{\pm 0.02}$	**0.99**$_{\pm 0.02}$	**0.87**$_{\pm 0.04}$	**0.82**$_{\pm 0.00}$	**0.01**$_{\pm 0.00}$	**0.99**$_{\pm 0.00}$	**0.69**$_{\pm 0.05}$
DroneRun	**0.27**$_{\pm 0.08}$	**0.59**$_{\pm 0.25}$	**0.80**$_{\pm 0.19}$	**0.67**$_{\pm 0.08}$	1.00$_{\pm 0.25}$	1.22$_{\pm 0.20}$	0.92$_{\pm 0.27}$	1.28$_{\pm 0.38}$	**0.70**$_{\pm 0.04}$	**0.48**$_{\pm 0.02}$	**0.29**$_{\pm 0.03}$	**0.22**$_{\pm 0.09}$	**0.89**$_{\pm 0.03}$	**0.92**$_{\pm 0.01}$
AntCircle	**0.10**$_{\pm 0.05}$	**0.18**$_{\pm 0.16}$	**0.25**$_{\pm 0.03}$	**0.50**$_{\pm 0.06}$	**0.62**$_{\pm 0.05}$	**0.83**$_{\pm 0.12}$	**0.05**$_{\pm 0.02}$	**0.32**$_{\pm 0.27}$	**0.65**$_{\pm 0.02}$	**0.55**$_{\pm 0.03}$	**0.15**$_{\pm 0.05}$	**0.00**$_{\pm 0.00}$	**0.77**$_{\pm 0.02}$	**0.74**$_{\pm 0.01}$
BallCircle	**0.77**$_{\pm 0.07}$	**0.92**$_{\pm 0.12}$	**0.94**$_{\pm 0.02}$	**0.75**$_{\pm 0.00}$	**0.81**$_{\pm 0.07}$	**0.82**$_{\pm 0.15}$	**0.97**$_{\pm 0.01}$	**0.94**$_{\pm 0.02}$	**0.92**$_{\pm 0.00}$	**0.92**$_{\pm 0.06}$	**0.29**$_{\pm 0.02}$	**0.00**$_{\pm 0.00}$	**0.92**$_{\pm 0.00}$	**0.90**$_{\pm 0.08}$
CarCircle	**0.81**$_{\pm 0.05}$	**0.84**$_{\pm 0.09}$	**0.47**$_{\pm 0.02}$	**0.63**$_{\pm 0.07}$	0.81$_{\pm 0.06}$	1.30$_{\pm 0.14}$	0.79$_{\pm 0.19}$	1.26$_{\pm 0.35}$	**0.85**$_{\pm 0.02}$	**0.80**$_{\pm 0.04}$	**0.29**$_{\pm 0.02}$	**0.00**$_{\pm 0.00}$	**0.87**$_{\pm 0.02}$	**0.93**$_{\pm 0.03}$
DroneCircle	**0.02**$_{\pm 0.00}$	**0.11**$_{\pm 0.04}$	**0.41**$_{\pm 0.02}$	**0.27**$_{\pm 0.06}$	**0.03**$_{\pm 0.01}$	**0.61**$_{\pm 0.38}$	**0.05**$_{\pm 0.01}$	**0.51**$_{\pm 0.27}$	**0.79**$_{\pm 0.02}$	**0.78**$_{\pm 0.06}$	**0.48**$_{\pm 0.04}$	**0.01**$_{\pm 0.00}$	**0.87**$_{\pm 0.03}$	**0.89**$_{\pm 0.09}$
Average	**0.52**$_{\pm 0.04}$	**0.60**$_{\pm 0.12}$	**0.63**$_{\pm 0.04}$	**0.50**$_{\pm 0.08}$	**0.74**$_{\pm 0.12}$	**0.90**$_{\pm 0.20}$	0.66$_{\pm 0.08}$	1.06$_{\pm 0.18}$	**0.80**$_{\pm 0.02}$	**0.76**$_{\pm 0.04}$	**0.36**$_{\pm 0.03}$	**0.03**$_{\pm 0.01}$	**0.86**$_{\pm 0.02}$	**0.87**$_{\pm 0.05}$

Metrics: We evaluate performance using normalized reward and cost returns. The normalized reward return is defined by $R_{\text{norm}} = \frac{R_\pi}{R_{\max}}$, where R_π is the reward return achieved by policy π and and $R_{\max}$ is the maximum reward return in the dataset. The normalized cost return is defined by $C_{\text{norm}} = \frac{C_\pi}{L}$, where C_π is the cost return achieved by policy π and L is the selected threshold. To provide better interpretation to decouple the two objectives in OSRL compared to traditional fixed thresholds, we use percentage-based thresholds calibrated to each task's cost range. Specifically, we define tight constraints as 10%, 20%, and 30% of the maximum cost to emphasize constraint satisfaction, and loose constraints as 70%, 80%, and 90% of the maximum cost to focus on reward maximization.

6.1 Can GAS achieve both safe and better performance with improved stitching ability?

The experiment results of different baselines under various tasks on tight and loose constraints are presented in table 1. In tight constraint settings, only GAS achieves the best and safe performance in all tasks. Traditional RL methods, such as CPQ, COptiDICE, WSAC, and VOCE, suffer from severe constraint violations both on average and for each task. Even among GM methods, CDT, while outperforming RL-assisted methods, still fails to ensure constraint satisfaction in multiple scenarios where GAS succeeds. FISOR maintains safety in most tasks but at a substantial cost to performance, with rewards significantly lower than GAS. In such tight constraint settings, the superiority of GAS over CDT on constraint satisfaction comes from the stitching ability, where GAS can stitch safe transitions among different timesteps and trajectories together. In loose constraint settings, although most baselines exhibit safe performance, GAS achieves the best performance on reward maximization. This performance gap between GAS and CDT highlights GAS's advanced reward maximization capabilities, leveraging its innovative transition stitching approach to combine high-reward segments while maintaining robust safety guarantees. Additional results on more tasks and baselines are provided in section E.3.

6.2 Can GAS preserve the zero-shot adaptation ability to different constraint thresholds?

A critical advantage of GM methods is their zero-shot adaptation ability in handling different constraint thresholds without retraining. To validate this ability in GAS, we compare our method with CDT and test thresholds from 10% to 90% of the maximum costs, increased by 10% each time. The results are shown in fig. 5. Compared with CDT, the cumulative costs of GAS are consistently below the constraint thresholds, whereas CDT performs unsafely in some cases, especially when the constraint thresholds are smaller than 30% of the maximum costs. In addition, as the constraint threshold increases, GAS flexibly adjusts its behavior and achieves progressively higher rewards

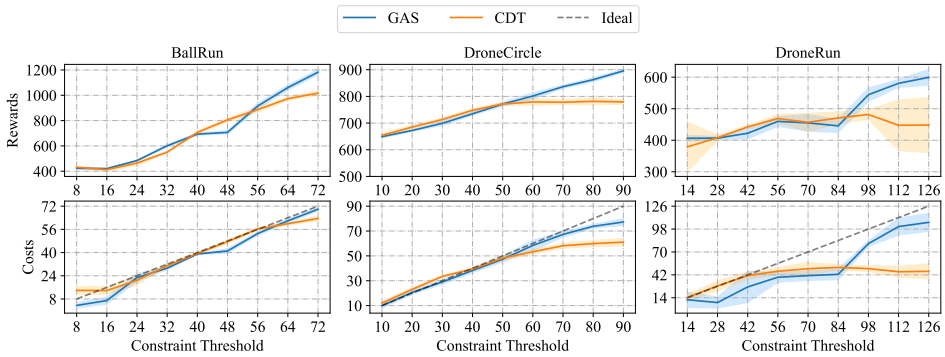

Figure 5: Evaluation results on zero-short adaptation. The x-axis indicates different selected thresholds and the y-axis indicates corresponding performance on cumulative rewards and costs. "Ideal" line indicates the case when the cumulative costs are equal to the constraint thresholds.

while ensuring safety, but CDT tends to be overly conservative. Results on more tasks are provided in section E.4.

6.3 Is GAS robust to imbalanced and human-specified target reward-cost return-to-goes?

To demonstrate GAS's robustness against imbalanced target reward-cost return-to-goes, we compare GAS and a variant without transition-level return relabeling (denoted as "GAS w/o Relabel") in settings with the constraint threshold 20% of the maximum costs and varying target reward return-to-goes. As shown in fig. 6, GAS consistently achieves safe performance as the target reward return-to-go increases. In contrast, GAS w/o Relabel can only achieve safe performance for a narrow band of reward targets. This highlights one of the GAS's key innovations: the transition-level return relabeling method in enhancing robustness. Ablation studies on other components of GAS are provided in section E.5 and section E.6.

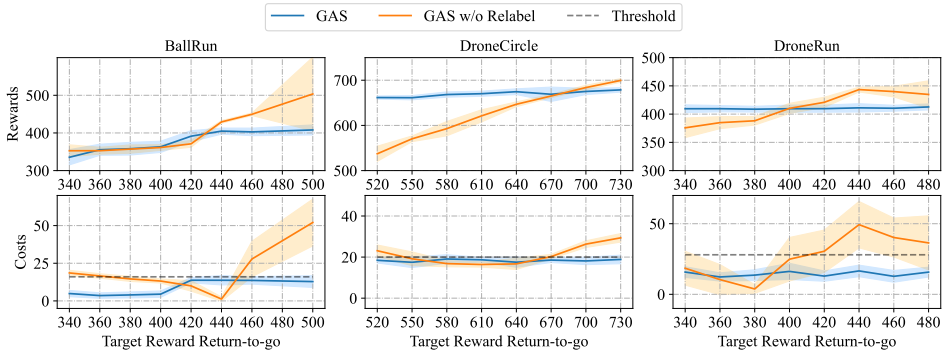

Figure 6: Evaluation results on robustness of imbalanced target reward-cost return-to-goes. The x-axis indicates different target reward return-to-go and the y-axis indicates corresponding performance on cumulative rewards and costs. "Threshold" line indicates constraint thresholds.

7 Conclusion

Aiming to address the limitations of existing GM-assisted OSRL methods, we propose a novel algorithm named GAS that concentrates on enhancing the stitching ability for a better balance of reward maximization and constraint satisfaction. GAS focuses on achieving better reward maximization and constraint satisfaction by training specialized reward and cost goal functions via expectile regression. These goal functions estimate the optimal achievable reward and cost returns and are used to guide the policy in effectively stitching transitions under relabeled reward-cost return-to-goes. Experiment results demonstrate GAS's superiority on reward maximization, constraint satisfaction, zero-shot adaptation, and robustness on imbalanced target reward-cost return-to-goes. A potential weakness of GAS is the trade-off between memory capability and stitching ability, as the current attention module struggles to fully capture true temporal dependencies in CMDPs. Future work could address this limitation by designing more advanced memory mechanisms into GAS to further enhance the robustness and performance of GM-assisted policies.

REPRODUCIBILITY STATEMENT

This paper introduces a new algorithm, named GAS. A clear theoretical explanation of GAS is illustrated in sections A and B. The clear pseudo code is shown in algorithm 1. Detailed experiment settings, including hyperparameters, are shown in section E. The experimental code will be released publicly after the review process if accepted.

REFERENCES

Eitan Altman. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. Mathematical methods of operations research, 48:387–417, 1998.

Anirudhan Badrinath, Yannis Flet-Berliac, Allen Nie, and Emma Brunskill. Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), Advances in Neural Information Processing Systems, volume 36, pp. 78006–78027. Curran Associates, Inc., 2023.

Sikha Bagui and Kunqi Li. Resampling imbalanced data for network intrusion detection datasets. Journal of Big Data, 8(1):6, 2021.

Yassine Chemingui, Aryan Deshwal, Honghao Wei, Alan Fern, and Jana Doppa. Constraint-adaptive policy switching for offline safe reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pp. 15722–15730, 2025.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems, 34:15084–15097, 2021.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

Xing Fang, Qichao Zhang, Yinfeng Gao, and Dongbin Zhao. Offline reinforcement learning for autonomous driving with real world driving data. In 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), pp. 3417–3422. IEEE, 2022.

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

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 2052–2062. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Sven Gronauer. Bullet-safety-gym: A framework for constrained reinforcement learning. Technical report, TUM Department of Electrical and Computer Engineering, Jan 2022.

Jiayi Guan, Guang Chen, Jiaming Ji, Long Yang, ao zhou, Zhijun Li, and changjun jiang. Voce: Variational optimization with conservative estimation for offline safe reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), Advances in Neural Information Processing Systems, volume 36, pp. 33758–33780. Curran Associates, Inc., 2023.

Zijian Guo, Weichao Zhou, Shengao Wang, and Wenchao Li. Constraint-conditioned actor-critic for offline safe reinforcement learning. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=nrRkAAAufl.

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

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of human preferences in dialog, 2020. URL https://openreview.net/forum?id=rJl5rRVFvH.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. Advances in Neural Information Processing Systems, 36:18964–18993, 2023.

Haeyong Kang, Thang Vu, and Chang D Yoo. Learning imbalanced datasets with maximum margin loss. In 2021 IEEE International Conference on Image Processing (ICIP), pp. 1269–1273. IEEE, 2021.

Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision convformer: Local filtering in metaformer is sufficient for decision making. In International Conference on Learning Representations, 2024a.

Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision convformer: Local filtering in metaformer is sufficient for decision making. In The Twelfth International Conference on Learning Representations, 2024b. URL https://openreview.net/forum?id=af2c8EaKl8.

Roger Koenker and Kevin F Hallock. Quantile regression. Journal of economic perspectives, 15(4): 143–156, 2001.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In Marina Meila and Tong Zhang (eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 5774–5783. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/kostrikov21a.html.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. COptiDICE: Offline constrained reinforcement learning via stationary distribution correction estimation. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=FLA55mBee6Q.

Namyeong Lee and Jun Moon. Offline reinforcement learning for automated stock trading. IEEE Access, 11:112577–112589, 2023.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, et al. Datasets and benchmarks for offline safe reinforcement learning. arXiv preprint arXiv:2306.09303, 2023a.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In International Conference on Machine Learning, pp. 21611–21630. PMLR, 2023b.

Yi Ma, Jianye Hao, Hebin Liang, and Chenjun Xiao. Rethinking decision transformer via hierarchical reinforcement learning. In Proceedings of the 41st International Conference on Machine Learning, pp. 33730–33745, 2024.

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

Jiawei Ren, Mingyuan Zhang, Cunjun Yu, and Ziwei Liu. Balanced mse for imbalanced visual regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7926–7935, 2022.

Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=rke7geHtwH.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In International Conference on Machine Learning, pp. 9133–9143. PMLR, 2020.

Honghao Wei, Xiyue Peng, Arnob Ghosh, and Xin Liu. Adversarially trained weighted actor-critic for safe offline reinforcement learning. Advances in Neural Information Processing Systems, 37: 52806–52835, 2024.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2020. URL https://openreview.net/forum?id=BJg9hTNKPH.

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. Advances in neural information processing systems, 36:18532–18550, 2023.

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=u-RuvyDYqCM.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence, 36(8): 8753–8760, Jun. 2022. doi: 10.1609/aaai.v36i8.20855. URL https://ojs.aaai.org/index.php/AAAI/article/view/20855.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In International Conference on Machine Learning, pp. 38989–39007. PMLR, 2023a.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 38989–39007. PMLR, 23–29 Jul 2023b.

Yuzhe Yang, Kaiwen Zha, Yingcong Chen, Hao Wang, and Dina Katabi. Delving into deep imbalanced regression. In International conference on machine learning, pp. 11842–11851. PMLR, 2021.

Yihang Yao, Zhepeng Cen, Wenhao Ding, Haohong Lin, Shiqi Liu, Tingnan Zhang, Wenhao Yu, and Ding Zhao. Oasis: Conditional distribution shaping for offline safe reinforcement learning. Advances in Neural Information Processing Systems, 37:78451–78478, 2024.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In international conference on machine learning, pp. 27042–27059. PMLR, 2022.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. arXiv preprint arXiv:2401.10700, 2024.

A EXPECTILE REGRESSION

Expectile regression (Koenker & Hallock, 2001) is a statistical modeling technique that generalizes traditional mean regression to obtain the weighted means across different parts of the distribution. Given a set of samples $\{x_i|i = 1, ..., N\}$ under a distribution, the expectile regression aims to achieve eq. (14) where α is the expectile level and $u = x - \bar{x}^\alpha$.

$$\min_{\bar{x}^\alpha} \mathbb{E}[|\alpha - \mathbb{1}(u < 0)| \cdot u^2] \rightarrow \min_{\bar{x}^\alpha} \mathbb{E}[|\alpha - \mathbb{1}((x - \bar{x}^\alpha) < 0)| \cdot (x - \bar{x}^\alpha)^2], \tag{14}$$

Property 1 (Kostrikov et al., 2022):

As α increases from 0.5 to 1, $\bar{x}^\alpha$ increases from the mean value to the largest value of a distribution:

$$\lim_{\alpha \rightarrow 1} \bar{x}^\alpha = \max_{x_i}\{x_i|i = 1, ..., N\} \tag{15}$$

For example, if $\alpha = 0.5$, it is the same as MSE since $|0.5 - \mathbb{1}(.)| = 0.5$; if $\alpha = 0.9$, it gives a weight of 0.9 to samples $x \geq \bar{x}^\alpha$ but only gives a weight of 0.1 to samples $x < \bar{x}^\alpha$.

This property makes expectile regression widely used in the estimation of the optimal value function in reinforcement learning (Kostrikov et al., 2021; Dabney et al., 2018) since the optimal value function is naturally defined as the largest reward return in a task.

Property 2 (Hansen-Estruch et al., 2023):

Let us represent the expectile regression as $f(u) = |\alpha - \mathbb{1}(u < 0)| \cdot u^2$ and $f'(u) = \frac{df(u)}{du}$ for easy and clear writing. We have:

$$f'(u) = |f'(u)| \cdot \frac{u}{|u|} \tag{16}$$

where $f(u)$ is convex and $f'(0) = 0$.

B THEORETICAL ANALYSIS OF GAS

B.1 ALGORITHM DERIVATION

Fowllowing the eq. (15) in **property 1**, eq. (9), and eq. (10), we can directly have:

$$\lim_{\alpha \rightarrow 1} V^R_{t:\Gamma}(s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t' = t + T - \Gamma) = \max_{\{(s_t, a_t, R_{t:\Gamma}, C_{t:\Gamma}, t' = t+T-\Gamma)\}} (R_{t:\Gamma}|V^C_{t:\Gamma} < \hat{C}_{t:\Gamma}) \tag{17}$$

This indicates that the reward goal function finally converges to the largest reward return-to-go within the constraint.

Then consider L_R in eq. (10) as $f(A^R_{t:\Gamma})$. When the reward goal function converges, we have

$$\begin{aligned}
\frac{\partial L_R}{\partial V^R_{t:\Gamma}} &= -\int_a \mu(a|s) \cdot |f'(A^R_{t:\Gamma})| \cdot \frac{A^R_{t:\Gamma}}{|A^R_{t:\Gamma}|} \\
&= -\int_a \mu(a|s) \cdot \frac{|f'(A^R_{t:\Gamma})|}{|A^R_{t:\Gamma}|} \cdot A^R_{t:\Gamma} \\
&= -\int_a \pi(a|s) \cdot A^R_{t:\Gamma} \\
&= 0
\end{aligned} \tag{18}$$

where $\pi(a|s) = \mu(a|s) \cdot \frac{|f'(A^R_{t:\Gamma})|}{N^{scale}|A^R_{t:\Gamma}|}$ is the target policy.

With the target policy, we can define the loss function of the cost goal function as $L_C = E_\pi[(C_{t:\Gamma} - V^C_{t:\Gamma})^2]$. When it converges, we have:

$$
\begin{aligned}
0 &= -2E_\pi[C_{t:\Gamma} - V^C_{t:\Gamma}] \\
&= -2\int_a \pi(a|s)\cdot(C_{t:\Gamma} - V^C_{t:\Gamma}) \\
&= -2/N^{scale}\cdot\int_a \mu(a|s)\cdot|\alpha - \mathbb{1}(A^R_{t:\Gamma}<0)|\cdot(C_{t:\Gamma} - V^C_{t:\Gamma})
\end{aligned} \tag{19}
$$

Thus, we can obtain the loss function of the cost goal function as eq. (12).

During the policy extraction, the target policy is supposed to be $\pi(a|s) = \mu(a|s)\cdot\frac{|f'(A^R_{t:\Gamma})|}{N^{scale}|A^R_{t:\Gamma}|}$. However, to further ensure safety, we add the constraint to the policy extraction:

$$
\pi(a|s) = 1/N^{scale}\cdot\mathbb{1}(V^C_{t:\Gamma}\le\hat{C}_{t:\Gamma})\cdot|\alpha - \mathbb{1}(A^R_{t:\Gamma}<0)|\cdot\mu(a|s) \tag{20}
$$

Thus, the loss function of the target policy is eq. (13).

B.2 Ablation Analysis

This ablation analysis aims to study the contribution of each separate component of GAS theoretically.

B.2.1 Stitching Ability (Expectile Regression)

Based on **property 1** in section A, it is easy to prove that:

$$
\lim_{\alpha=0.5} V^R_{t:\Gamma}(s_t,\hat{R}_{t:\Gamma},\hat{C}_{t:\Gamma},t') < \lim_{\alpha\to1} V^R_{t:\Gamma}(s_t,\hat{R}_{t:\Gamma},\hat{C}_{t:\Gamma},t') = \max_{\{(s_t,a_t,R_{t:\Gamma},C_{t:\Gamma},t')\}}(R_{t:\Gamma}|V^C_{t:\Gamma}<\hat{C}_{t:\Gamma}) \tag{21}
$$

Thus the optimality guarantee is lost without expectile regression.

B.2.2 Temporal Segmented Return Augmentation

We can partition the dataset after augmentation as $D = D^o\cup D^a$, where D^o is the original dataset and D^a is the augmented part.

$$
\begin{aligned}
\lim_{\alpha\to1} E_D[V^R_{t:\Gamma}(s_t,\hat{R}_{t:\Gamma},\hat{C}_{t:\Gamma},t')] &= \max_{\{(s_t,a_t,R_{t:\Gamma},C_{t:\Gamma},t')\sim D\}}(R_{t:\Gamma}|V^C_{t:\Gamma}<\hat{C}_{t:\Gamma}) \\
&= \max[\max_{\{D^o\}}(R_{t:\Gamma}|V^C_{t:\Gamma}<\hat{C}_{t:\Gamma}),\ \max_{\{D^a\}}(R_{t:\Gamma}|V^C_{t:\Gamma}<\hat{C}_{t:\Gamma})] \\
&\ge \max_{\{(s_t,a_t,R_{t:\Gamma},C_{t:\Gamma},t')\sim D^o\}}(R_{t:\Gamma}|V^C_{t:\Gamma}<\hat{C}_{t:\Gamma})
\end{aligned} \tag{22}
$$

This result indicates that the optimality guarantee in the original dataset is just a lower bound of that after augmentation. With augmentation, the optimality guarantee is always larger than that in the original dataset, which further improves the stitching ability.

B.2.3 Transition-level Return Relabeling

Since the purpose of this method is to make GAS more robust, this paper will discuss it from the perspective of the input and output. For simplicity, the influence on the reward goal function is analyzed as an example, considering that it has no difference from that on the cost goal function and the policy.

Without Relabeling, both inputs and loss functions utilize $(R,\ C)$ pairs following the dataset distribution. This indicates that only with the appropriate R and C, the goal functions and policy can achieve the optimal value:

$$
\lim_{\alpha\to1} V^R_{t:\Gamma}(s_t,R_{t:\Gamma},C_{t:\Gamma},t') = \max_{\{(s_t,a_t,R_{t:\Gamma},C_{t:\Gamma},t')\}}(R_{t:\Gamma}|V^C_{t:\Gamma}<C_{t:\Gamma}) \tag{23}
$$

However, during the test stage, the targets $\hat{R}$ and $\hat{C}$ cannot be accurately known and need to be specified by users. Thus it will suffers from inaccurate R and C signals and cannot determine which target should be prioritized when R and C conflict with each other.

With Relabeling, the targets are relabeled to consider a more robust input distribution while loss functions still utilize true $(R,\ C)$ pairs to update the outputs with expectile regression. This indicates that even with inaccurate R and C, the goal functions can still obtain the optimal value, and the policy can provide the corresponding action:

$$\lim_{\alpha\to 1} V^R_{t:\Gamma}(s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t') = \max_{\{(s_t, a_t, R_{t:\Gamma}, C_{t:\Gamma}, t')\}} (R_{t:\Gamma} | V^C_{t:\Gamma} < \hat{C}_{t:\Gamma}) \tag{24}$$

C Algorithm Details

We present the full algorithm of our method in algorithm 1.

Algorithm 1 Goal-Assisted Stitching (GAS)

Network: Initialize two goal functions $V^R_{t:\Gamma}(s, R, C, t' = t+T-\Gamma)$, $V^C_{t:\Gamma}(s, R, C, t' = t+T-\Gamma)$, and policy $\pi(a|s, R, C, V^R, V^C, t' = t + T - \Gamma)$.
for iteration$= 0, ..., N$ **do**
  Sample transitions $\hat{\mathcal{D}} = \{(s, a, R_{t:\Gamma}, C_{t:\Gamma})\} \sim \mathcal{D}$ with $1-\epsilon$ and $\mathcal{D}^q$ with ϵ probability.
  Get augmented return and cost return:
    $\hat{R}_{t:\Gamma} = U((1-\delta)R_{t:\Gamma}, (1+\delta)R_{t:\Gamma})$.
    $\hat{C}_{t:\Gamma} = U(C_{t:\Gamma}, C^{\max})$.
  Get advantage function:
    $A^R_{t:\Gamma} = \mathbb{1}(V^C_{t:\Gamma} < \hat{C}_{t:\Gamma}) \cdot R_{t:\Gamma} - V^R_{t:\Gamma}(s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t' = t + T - \Gamma)$.
    $A^C_{t:\Gamma} = C_{t:\Gamma} - V^C_{t:\Gamma}(s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, t' = t + T - \Gamma)$.
  Update reward goal function:
    $L_R = \mathbb{E}_{\hat{\mathcal{D}}}[|\alpha - \mathbb{1}(A^R_{t:\Gamma} < 0)| \cdot (A^R_{t:\Gamma})^2]$.
  Update cost goal function:
    $L_C = \mathbb{E}_{\hat{\mathcal{D}}}[|\alpha - \mathbb{1}(A^R_{t:\Gamma} < 0)| \cdot (A^C_{t:\Gamma})^2]$.
  Policy Extraction:
    $L_\pi = \mathbb{E}_{\hat{\mathcal{D}}}[\mathbb{1}(V^C_{t:\Gamma} < \hat{C}_{t:\Gamma}) \cdot |\alpha - \mathbb{1}(A^R_{t:\Gamma} < 0)| \cdot (\pi(a|s_t, \hat{R}_{t:\Gamma}, \hat{C}_{t:\Gamma}, V^R_{t:\Gamma}, V^C_{t:\Gamma}, t') - a_t)^2]$.

D Return Relabeling Technique Details

The misalignment problem between the training and test phases is first proposed in Liu et al. (2023b), which indicates that users may select target reward-cost pairs different from the training inputs. This misalignment problem has become serious when users select imbalanced target reward-cost pairs, such as extremely large target rewards and extremely small target costs. To address this issue, CDT (Liu et al., 2023b) proposes a trajectory-level return relabel method as CDT only learns policy within the trajectory. This method, although it improves the robustness of CDT, degrade the ability to maximize rewards and satisfy constraints, as the relabeled returns provide wrong information about the trajectory. Inspired by the trajectory-level return labeling in Liu et al. (2023b), we propose a more fine-grained, transition-level return relabeling mechanism fitting in with transition-level stitching. Different from CDT, our proposed GAS does not directly update the policy guided by only the relabeled values. Instead, we utilize them to assist training through intermediate optimal goals and update the policy based on these goals in the loss function during optimization. As a result, GAS retains the robustness of the policy without affecting reward maximization and constraint satisfaction.

E Experiment Details

E.1 Benchmark and Tasks

Bullet-safety-gym (Gronauer, 2022) and Safety-gamnasium (Ji et al., 2023) are utilized for experiments. We consider 8 cases in Bullet-safety-gym and 4 cases in Safety-gamnasium, involving two tasks (*Run* and *Circle*) and multiple types of robots (*Ant*, *Ball*, *Car*, *Drone*, and *Point*). The tasks *AntCircle*, *PointCircle1*, *PointCircle2*, *CarCircle1*, and *CarCircle2* are considered as complex tasks with episode length $T = 500$ and maximum cost $C^{\max} \geq 200$, while other tasks are regarded as

Table 2: Benchmark details. The cost range indicates the maximum cumulative costs among the offline trajectories. Offline trajectories indicate the number of trajectories in the offline dataset.

Benchmarks	Task	State Space	Action Space	Cost Range	Episode Length (T)	Offline Trajectories
Bullet-safety-gym	AntRun	33	8	150	200	1816
	BallRun	7	2	80	100	940
	CarRun	7	2	40	200	651
	DroneRun	17	4	140	200	1990
	AntCircle	34	8	200	500	5728
	BallCircle	8	2	80	200	886
	CarCircle	8	2	100	300	1450
	DroneCircle	18	4	100	300	1923
Safety-gymnasium	PointCircle1	28	2	200	500	1098
	PointCircle2	28	2	300	500	895
	CarCircle1	40	2	250	500	1271
	CarCircle2	40	2	400	500	940

simpler tasks with episode length less than 301. Details of the parameter settings for each tasks are shown in table 2, which is part of DSRL (Liu et al., 2023a). As for the cost definition and reward definition, we follow the same standard with DSRL.

E.2 HYPERPARAMETER

Table 3: Hyperparameters of GAS.

Parameter	Value	Parameter	Value
Number of layers	7	Hidden size	128
Embedding size	64	Batch Size	2048
Learning rate	0.0001	Adam betas	(0.9,0.999)
Grad norm clip	0.25	Weight decay	0.0001
Expectile level α	0.8	Reward relabel level δ	0.1
Dataset reshape threshold $q\%$	10%	Sample probability ϵ	0.5

The table 3 shows the detailed hyperparameter for GAS used in section 6 and table 4 shows the target reward and cost return-to-go pairs used in the testing stage. Notably, GAS is not sensitive to the target reward-cost return-to-go pairs as we demonstrate in section 6.3.

Table 4: Target reward and cost return-to-go pairs for GAS in testing stage.

Benchmark	Task	Cost range	10%	20%	30%	70%	80%	90%
Bullet-safety-gym	AntRun	150	(690, 15)	(690, 30)	(700, 45)	(750, 105)	(800, 120)	(820, 135)
	BallRun	80	(420, 8)	(420, 16)	(500, 24)	(900, 56)	(1000, 64)	(1200, 72)
	CarRun	40	(572, 4)	(572, 8)	(572, 12)	(572, 28)	(572, 32)	(572, 36)
	DroneRun	140	(400, 14)	(420, 28)	(45, 42)	(600, 98)	(620, 112)	(640, 126)
	AntCircle	200	(160, 20)	(200, 40)	(240, 60)	(320, 140)	(350, 160)	(400, 180)
	BallCircle	80	(500, 8)	(600, 16)	(690, 24)	(800, 56)	(810, 64)	(820, 72)
	CarCircle	100	(370, 10)	(390, 20)	(410, 30)	(480, 70)	(480, 80)	(480, 90)
	DroneCircle	100	(600, 10)	(650, 20)	(700, 25)	(830, 70)	(850, 80)	(870, 90)
Safety-gymnasium	PointCircle1	200	(43,20)	(45,40)	(46,60)	(52,140)	(54,160)	(58,180)
	PointCircle2	300	(36,30)	(41,60)	(42,90)	(45,210)	(47,240)	(48,270)
	CarCircle1	250	(4,25)	(8,50)	(10,75)	(13,175)	(15,200)	(18,225)
	CarCircle2	400	(8,40)	(14,80)	(15,120)	(22,280)	(23,320)	(27,360)

E.3 MORE EXPERIMENTS ON IMPROVED STITCHING ABILITY

The table 5 is an extension of table 1 on four tasks of Safety-gymnasium: *PointCircle1* and *PointCircle2*, *CarCircle1* and *CarCircle2*. Furthermore, we also compare GAS with two recent RL-based algorithms that incorporate zero-shot adaptation ability, including CAPS (Chemingui et al., 2025) and CCAC (Guo et al., 2025) in table 6. Similar to table 1, GAS exhibits superiority on constraint satisfaction in tight constraint settings and reward maximization in loose constraint settings, which benefit from the improved stitching ability.

Table 5: Additional normalized evaluation results on more tasks.

Methods	CPQ		COptiDICE		WSAC		VOCE		CDT		FISOR		GAS	
Tasks	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓
Tight Constraint Threshold: Average results on thresholds with 10%, 20%, and 30% of the maximium costs for each task.														
PointCircle1	0.66±0.09	1.80±0.91	0.91±0.02	5.76±0.30	0.40±0.12	3.02±1.44	0.54±0.10	4.83±0.80	**0.70±0.01**	**0.63±0.17**	0.73±0.03	1.23±0.39	**0.69±0.01**	**0.38±0.14**
PointCircle2	0.74±0.11	4.53±1.22	0.91±0.02	5.92±0.26	0.55±0.05	1.64±0.65	0.85±0.09	5.31±0.65	0.77±0.01	1.18±0.13	0.84±0.02	1.42±0.23	**0.75±0.02**	**0.99±0.2**
CarCircle1	0.45±0.16	3.00±1.44	0.79±0.03	4.95±0.45	0.48±0.10	5.12±2.37	0.14±0.06	9.03±0.70	0.71±0.04	2.25±0.44	0.75±0.03	2.15±0.68	**0.56±0.03**	**0.62±0.30**
CarCircle2	0.65±0.07	2.32±0.90	0.79±0.04	3.82±0.38	0.58±0.05	1.29±0.44	0.57±0.06	4.72±0.46	0.72±0.03	2.26±0.34	**0.57±0.03**	**0.40±0.27**	**0.50±0.01**	**0.74±0.20**
Loose Constraint Threshold: Average results on thresholds with 70%, 80%, and 90% of the maximium costs for each task.														
PointCircle1	**0.66±0.17**	**0.34±0.26**	0.90±0.01	1.18±0.07	0.88±0.03	1.31±0.08	0.59±0.18	1.34±0.25	**0.73±0.01**	**0.31±0.06**	**0.73±0.04**	**0.92±0.17**	**0.88±0.01**	**0.97±0.03**
PointCircle2	**0.61±0.17**	**0.98±0.29**	0.91±0.02	1.23±0.06	0.90±0.03	1.08±0.12	0.83±0.07	1.11±0.07	**0.76±0.01**	**0.18±0.05**	**0.84±0.03**	**0.74±0.15**	**0.88±0.01**	**0.95±0.03**
CarCircle1	0.54±0.15	1.01±0.26	0.80±0.03	1.01±0.08	0.63±0.06	1.28±0.16	0.29±0.09	1.52±0.22	**0.71±0.04**	**0.57±0.11**	**0.75±0.03**	**0.48±0.13**	**0.82±0.02**	**0.89±0.07**
CarCircle2	0.76±0.05	1.20±0.06	**0.79±0.04**	**0.79±0.08**	**0.71±0.05**	**0.75±0.13**	0.53±0.05	1.09±0.07	**0.73±0.04**	**0.58±0.12**	**0.58±0.03**	**0.09±0.05**	**0.80±0.03**	**0.82±0.08**

Table 6: Additional normalized evaluation results on more baselines.

Methods	CAPS		CCAC		GAS	
Tasks	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓
Tight Constraint Threshold.						
AntRun	**0.64±0.04**	**0.96±0.27**	**0.14±0.01**	**0.01±0.01**	**0.72±0.02**	**0.70±0.30**
BallRun	0.15±0.04	1.04±0.25	0.77±0.00	3.65±0.10	**0.33±0.01**	**0.62±0.15**
CarRun	**0.98±0.00**	**0.38±0.42**	**0.96±0.00**	**0.47±0.72**	**0.99±0.01**	**0.19±0.05**
DroneRun	0.51±0.04	1.11±1.41	0.40±0.05	3.07±0.80	**0.60±0.02**	**0.22±0.13**
AntCircle	**0.47±0.06**	**0.31±0.11**	**0.22±0.03**	**0.75±0.33**	**0.52±0.02**	**0.96±0.10**
BallCircle	**0.63±0.02**	**0.47±0.05**	**0.77±0.01**	**0.35±0.01**	**0.71±0.00**	**0.84±0.20**
CarCircle	**0.64±0.04**	**0.66±0.06**	0.76±0.01	1.42±0.38	**0.70±0.03**	**0.84±0.13**
DroneCircle	**0.64±0.02**	**0.62±0.06**	**0.35±0.04**	**0.78±0.15**	**0.68±0.01**	**0.96±0.27**
PointCircle1	0.62±0.06	1.37±0.62	0.71±0.02	1.54±0.37	**0.69±0.01**	**0.38±0.14**
PointCircle2	**0.71±0.04**	**0.90±0.26**	0.10±0.12	3.87±2.21	**0.75±0.02**	**0.99±0.2**
CarCircle1	0.69±0.03	1.62±0.43	0.52±0.08	3.96±1.43	**0.56±0.03**	**0.62±0.30**
CarCircle2	**0.56±0.06**	**0.82±0.37**	0.57±0.02	2.44±1.08	**0.50±0.01**	**0.74±0.20**
Loose Constraint Threshold.						
AntRun	**0.85±0.03**	**1.00±0.11**	**0.08±0.02**	**0.00±0.00**	**0.84±0.03**	**0.93±0.05**
BallRun	0.47±0.10	1.18±0.03	1.21±0.00	1.44±0.00	**0.76±0.00**	**0.96±0.04**
CarRun	**0.99±0.00**	**0.28±0.35**	**0.86±0.05**	**0.19±0.20**	**0.99±0.00**	**0.69±0.05**
DroneRun	**0.53±0.05**	**0.22±0.28**	**0.60±0.10**	**0.81±0.13**	**0.89±0.03**	**0.92±0.01**
AntCircle	**0.70±0.04**	**0.61±0.06**	**0.19±0.03**	**0.40±0.08**	**0.77±0.02**	**0.74±0.01**
BallCircle	**0.92±0.01**	**0.84±0.02**	**0.92±0.01**	**0.83±0.01**	**0.92±0.00**	**0.90±0.08**
CarCircle	**0.86±0.01**	**0.93±0.01**	**0.81±0.03**	**0.93±0.08**	**0.87±0.02**	**0.93±0.03**
DroneCircle	**0.85±0.03**	**0.94±0.01**	**0.21±0.03**	**0.57±0.12**	**0.87±0.03**	**0.89±0.09**
PointCircle1	**0.76±0.06**	**0.68±0.21**	**0.75±0.03**	**0.97±0.19**	**0.88±0.01**	**0.97±0.03**
PointCircle2	**0.80±0.04**	**0.70±0.16**	**0.06±0.05**	**0.09±0.17**	**0.88±0.01**	**0.95±0.03**
CarCircle1	**0.80±0.03**	**0.80±0.06**	0.42±0.11	1.04±0.19	**0.82±0.02**	**0.89±0.07**
CarCircle2	**0.72±0.05**	**0.54±0.09**	0.49±0.06	1.21±0.05	**0.80±0.03**	**0.82±0.08**

E.4 More Experiments on the Zero-shot Adaptation Ability

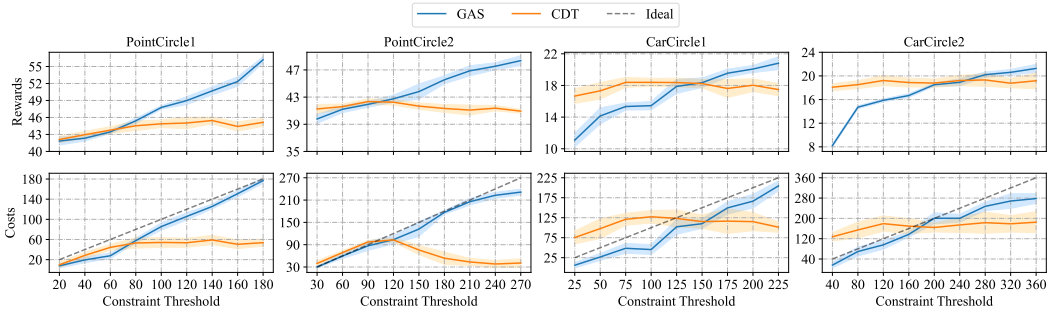

Figure 7: Evaluation results on zero-shot adaptation with complex tasks. The x-axis indicates different selected thresholds and the y-axis indicates corresponding performance on cumulative rewards and costs. "Ideal" line indicates the case when the cumulative costs are equal to the constraint thresholds.

The fig. 7 demonstrates the zero-shot adaptation ability comparison of GAS and CDT on more complex tasks. Even on more complex tasks, GAS can preserve the reward maximization and constraint satisfaction ability for all cost ranges. CDT, in contrast, suffers from constraint violation in tight constraint settings or is overly conservative as the threshold increases.

E.5 Ablation Study on Stitching Ability and TSRA

To verify the theoretical improvement for GAS's component in section B, we compare GAS to a variant without expectile regression (denoted as "GAS w/o Stitching") and a variant without Temporal Segmented Return Augmentation (denoted as "GAS w/o TSRA"). As shown in fig. 8, without expectile regression, GAS w/o Stitching degrades to conditional behavior cloning ($\alpha = 0.5$), which has similar performance and problems to CDT. This result is also consistent with section 4.1, where both GAS w/o Stitching and CDT suffer from insufficient stitching ability. Besides, the performance of GAS w/o TSRA is almost completely inferior to GAS. This result validates the theoretical analysis result, where the optimality without augmentation is just a lower bound of that with augmentation. Furthermore, GAS w/o TSRA suffers from constraint violation for tight constraint thresholds compared to GAS. This result indicates that the flexible sub-trajectories augmented by TSRA provide additional benefits when original trajectories fail to satisfy the tight constraint thresholds.

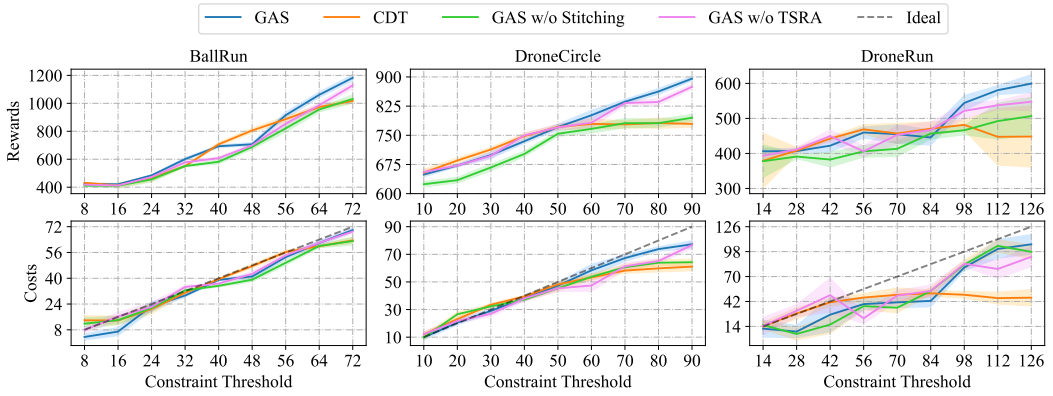

Figure 8: Ablation studies on the stitching ability (expectile regression) and temporal segmented return augmentation.

E.6 Ablation Study on Dataset Reshaping Method

To assess the importance of the dataset reshaping on the training stability and efficiency, we compare GAS and a variant without dataset reshaping (denoted as "GAS w/o DR") in *DroneCircle* and *CarCircle* tasks under different constraint thresholds. As shown in figs. 9 and 10, the dataset reshaping method can significantly improve the training efficiency of GAS while reducing the training variance.

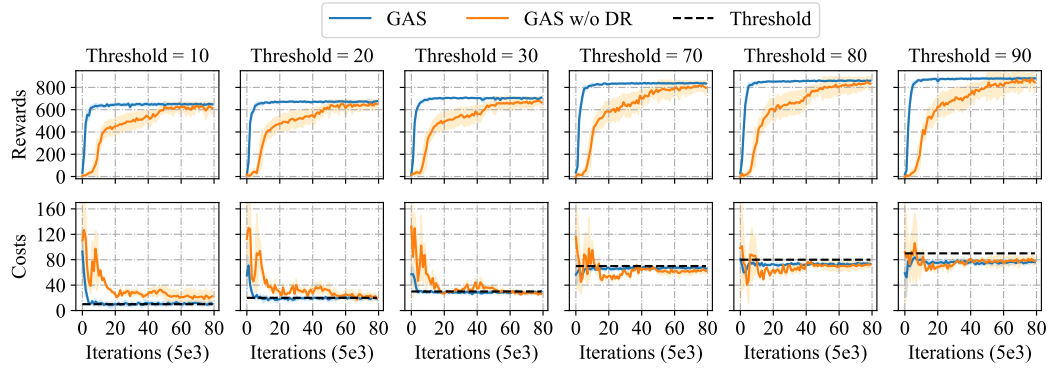

Figure 9: Ablation study of dataset distribution reshaping on task *DroneCircle*.

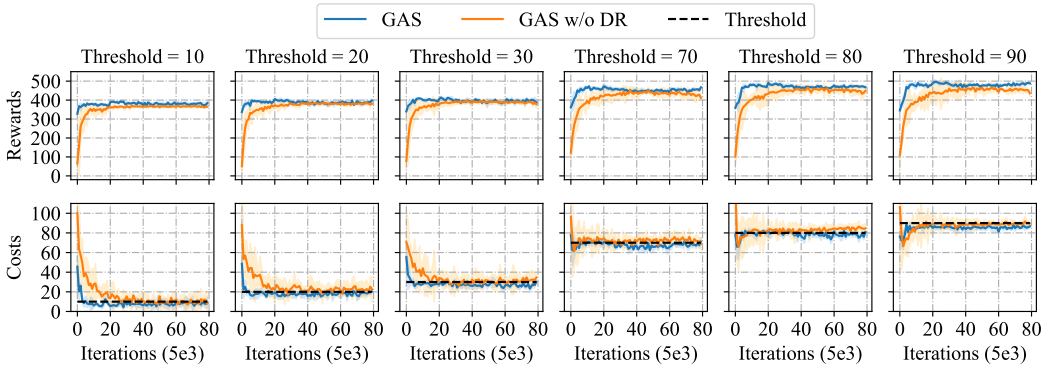

Figure 10: Ablation study of dataset distribution reshaping on task *CarCircle*.

E.7 Infrastructure and Time Cost

We run all the experiments and all the baselines with:

1. GPT version: NVIDIA GeForce RTX 3080.
2. CPU version: Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz.

Table 7: Training iterations and time for GAS and baselines on task *CarCircle*.

Method	CPQ	COptiDICE	WSAC	VOCE	CDT	FISOR	GAS
Iteration times	4e5	4e5	3e4	4e4	4e5	4e5	4e5
Time	2h19min	2h43min	3h10min	3h38min	11h32min	1h44min	8h20min

We also test the training time of GAS and baselines on task *CarCircle*, as shown in table 7. For most baselines, we follow the same iteration times. As for WSAC and VOCE, we follow the training standard of the original paper and make sure their convergence since training $4e5$ times is much too long for them. Notably, both GAS and CDT only need to be trained once for all thresholds, even though the training time is much longer, while other baselines need to be re-trained for each threshold. Besides, GAS is faster than CDT as GAS only utilizes the MLP architecture, but CDT utilize the Transformer architecture.

F The Use of Large Language Models (LLMs)

In this paper, LLMs are utilized to polish writing (e.g., grammar, spelling, word choice).