SAFORMER: A CONDITIONAL SEQUENCE MODELING APPROACH TO OFFLINE SAFE REINFORCEMENT LEARNING

Qin Zhang[1]*, **Linrui Zhang**[1]*, **Li shen**[2], **Haoran Xu**[3], **Bowen Wang**[4], **Bo Yuan**[5]
Yongzhe Chang[1] **& Xueqian Wang** [1]
[1]Center for Artificial Intelligence and Robotics,Tsinghua University
[2]JD Explore Academy [3]Institute for Al Industry Research (AIR), Tsinghua University
[4]University of Wisconsin-Madison [5]Research Institute of Tsinghua University in Shenzhen
{zhangqin21,zlr20}@tsinghua.edu.cn
wang.xq@sz.tsinghua.edu.cn

ABSTRACT

Offline safe RL is of great practical relevance for deploying agents in real-world applications. However, acquiring constraint-satisfying policies from the fixed dataset is non-trivial for conventional approaches. Even worse, the learned constraints are stationary and may become invalid when the online safety requirement changes. In this paper, we present a novel offline safe RL approach referred to as SaFormer, which tackles the above issues via conditional sequence modeling. In contrast to existing sequence models, we propose cost-related tokens to restrict the action space and a posterior safety verification to enforce the constraint explicitly. Specifically, SaFormer performs a two-stage auto-regression conditioned by the maximum remaining cost to generate feasible candidates. It then filters out unsafe attempts and executes the optimal action with the highest expected return. Extensive experiments demonstrate the efficacy of SaFormer featuring (1) competitive returns with tightened constraint satisfaction; (2) adaptability to the in-range cost values of the offline data without retraining; (3) generalizability for constraints beyond the current dataset.

1 INTRODUCTION

Reinforcement learning (RL) agents have achieved significant success in simulated environments (Mnih et al., 2015; Silver et al., 2017; Vinyals et al., 2019) but may pose potential risks when deployed in the real world. Safe RL (a.k.a constrained RL), has gained traction in recent years as most safety-critical applications require agents to follow certain constraints. However, existing solutions often satisfy constraints through trial and error (Achiam et al., 2017; Yang et al., 2020), leading to a risk-filled training process. In such scenarios, it is more practical to learn constraint-satisfying policies from offline data (Levine et al., 2020), but this poses several challenges, including the identification of mixed, unsafe, or conflicting demonstrations, insufficient long-term cost return estimation, and unstable optimization.

In this paper, we tackle the above issues by leveraging sequence modeling, which is of independent interest in recent offline RL literature (Chen et al., 2021; Janner et al., 2021). This paradigm shows the promise for a reliable cost estimation for safety-critical tasks. However, applying the current art to offline safe RL is still intractable. Take Decision Transformer (DT) (Chen et al., 2021) as an example. It makes sense to increase the reward-to-go (RTG) manually for higher rewards, but that might be detrimental when considering that the agent demands more information to navigate the trade-off between performance and safety in the constrained action space.

We propose a novel architecture called SaFormer, which is the first sequence modeling approach to offline safe RL. We introduce two cost-related tokens, the cost limit token and the cost-to-go (CTG)

*denote the equal contribution in this paper

token, which are used to determine the target reward-to-go (RTG) and generate feasible actions. Besides, SaFormer does not rely on unidirectional sequence generation to achieve zero constraint violation, but instead samples a batch of candidates, filters out unsafe attempts, and executes the feasible action with the highest expected return. We follow a practical pattern in our experiments, using cost-agnostic offline data acquisition based on arbitrary behavior policies and relabeling the dataset with specific cost criteria to facilitate the reuse of offline data.

We conduct experiments on D4RL benchmarks (Fu et al., 2020) subjecting to constraints on the cumulative torque. Empirical results confirm the strengths of our approach:

1. SaFormer outperforms state-of-the-art baselines for competitive reward improvements and strict constraint satisfactions in both discounted and undiscounted settings.
2. SaFormer treats the constraint as contextual tokens and is adaptive to different cost limits in the range of existing offline data without retraining.
3. SaFormer supports online fine-tuning to satisfy the unexplored constraints beyond the dataset.

2 Related Work

Offline Safe Reinforcement Learning Offline safe RL is an intersection of safe RL and offline RL . Conventional methods Tessler et al. (2018); Wang et al. (2020) are less effective when dealing with constrained optimization and conservative learning simultaneously. Le et al. (2019) are the first to study the problem of batch policy learning under constraints. They simply utilize the off-policy evaluation for safety constraints, which limits their approach to the discrete action space. Based on the above work, Polosky et al. (2022) propose a algorithm that corrects the rewards-optimal policy back to the cost-feasible set through the Fenchel duality. Their analysis explicitly accounts for the distributional shift and offers non-asymptotic confidence bounds on the cost return. COptiDICE (Lee et al., 2021) instead directly estimates the stationary distribution corrections of the optimal policy and therefore yields a single solvable optimization objective. Orthogonal to existing approaches, Xu et al. (2022a) trains an independent safety critic to penalizes unsafe actions and disables policy updates on them. Their proposed CPQ algorithm achieves impressive performance on a set of constrained locomotion tasks in continuous spaces.

RL via Sequence Modeling Sequence modeling has been widely adopted in natural language processing and computer vision. Recently, casting offline RL as a sequence generation problem shows the promise as well (Hu et al., 2022). Trajectory Transformer (TT) (Janner et al., 2021) discretizes the state-action-reward sequence and maximizes the sampling probability via beam search. Moreover, the transformer-based architecture features more reliable long-horizon prediction ability than single-step models, which also enables a fake trajectory bootstrapping for offline data augmentation (Wang et al., 2022). Decision Transformer (DT) (Chen et al., 2021) utilizes the future return to condition the sequence generation. Their model-free method generally outperforms TT in terms of cumulative rewards. Xu et al. (2022b) pre-train DT on a large-scale dataset and realizes few-shot generalization by providing demonstrations. Zheng et al. (2022) propose a stochastic DT to overcome the gap between offline training and online fine-tuning. To our best knowledge, there is no existing work that relates sequence modeling to offline safe RL.

3 Preliminaries

Problem Formulation we are interested in acquiring the constraint-satisfying policy from a stationary offline dataset. The problem formulation follows the constrained Markov Decision Process denoted by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, \mathcal{R}, \mathcal{C})$. Here, $\mathcal{S}$ and $\mathcal{A}$ represent the state space and the action space, respectively. $\mathcal{P}(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$ accounts for the state transition probability . $\rho_0 : \mathcal{S} \mapsto [0,1]$ is the initial state distribution. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ and $\mathcal{C} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^+$ denote the reward and cost functions. A policy $\pi : S \mapsto P(A)$ maps the current state to a distribution over the action space. The discounted cumulative return of policy π can be calculated by $J_R(\pi) = \mathbb{E}_\tau \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, where τ is determined by $s_0 \sim \rho_0(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. we define the discounted cost return as $J_C(\pi) = \mathbb{E}_\tau \left[\sum_{t=0}^{\infty} \gamma_c^t r(s_t, a_t)\right]$. Formally, the constrained RL problem is given by

$$\max_{\pi} J_R(\pi) \quad \text{s.t. } J_C(\pi) \leq d. \tag{1}$$

The agent is excepted to limit the cost return down to the threshold d.In our offline setting, the agent is not accessible to the environment. We can only tackle the above constrained problem using a fixed dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, collected from a range of behaviour policies π_β.

We assume that our offline safe RL tasks feature the following properties. First, the reward and cost returns are at least weakly correlated, which is essential to restrict the expected return via cost tokens. This assumption is mild since risk usually comes with profit. Second, the offline data contains trajectories with heterogeneous cost returns. This assumption is a necessary condition for our safety identification and action proposal. Instead, our method may not be distinguished from imitation learning (IL) when learning from completely homogeneous demonstrations, for example, all the trajectories are cost-signal-free and the expected constraint threshold is equal to 0 as well.

Revisit Decision Transformer in Offline Safe RL Decision Transformer (DT) (Chen et al., 2021) brings a novel sequence modeling paradigm to offline RL. It employs 3 types of tokens, namely $\hat{R}_t, s_t, a_t$, and represent a trajectory τ in order as $\langle \hat{R}_0, s_0, a_0, \hat{R}_1, s_1, a_1, ..., \hat{R}_T, s_T, a_T \rangle$. At each timestep t, DT is fed with $3K$ tokens from the last K timesteps and then predicts the following deterministic action $a_t = \pi_{\text{DT}}(\{\hat{R}_i, s_i, a_i\}_{t-K}^{t-1} \cup \{\hat{R}_{t-k}, s_{t-k}\})$ via a casually masked Transformer. Here, $\{\hat{R}_i, s_i, a_i\}_{t-K}^{t-1}$ is as a shorthand for the subsequence $\langle \hat{R}_{\max\{0.t-K\}}, s_{\max\{0.t-K\}}, a_{\max\{0.t-K\}}, ..., \hat{R}_{t-1}, s_{t-1}, a_{t-1} \rangle$. The policy is learned by minimizing the ℓ_2 objective :

$$\mathbb{E}_{\mathcal{D}} \sum_{k=0}^{K} \left(a_{t-k} - \pi_{\text{DT}}(\{\hat{R}_i, s_i, a_i\}_{t-K}^{t-1-k} \cup \{\hat{R}_{t-k}, s_{t-k}\}) \right)^2. \tag{2}$$

DT is specified with the desired episodic return $\hat{R}_0$ at the initial state s_0 and then a_0.Once a_0 is performed, the agent observes the reward r_0, which gives $\hat{R}_1 = \hat{R}_0 - r_0$.

Chen et al. (2021) show that the eventual return strongly correlates with the initial RTG which can even be extrapolated to values beyond the maximum in the dataset. Nevertheless, tweaking RTG manually in offline safe RL is not straightforward since the optimal RTG is heavily dependent on the constraints. Raising RTG aggressively may incur infeasible actions that conflict with certain safety constraints. In summary, DT has to be conditioned via cost-reliant RTGs and other informative tokens to navigate the trade-off between performance and safety.

4 METHODOLOGY

In this section, we present SaFormer, which addresses the above issues with the conditional sequence generation and the posterior safety verification, as illustrated in fig. 1,The actor (top & middle) first generates the RTG distribution via cost-related tokens and then generates action candidates after sampling potential RTGs. The critic (bottom) predicts the long-term cost return of each candidate to filters out unsafe attempts and eventually selects the feasible action with the highest future reward designation.

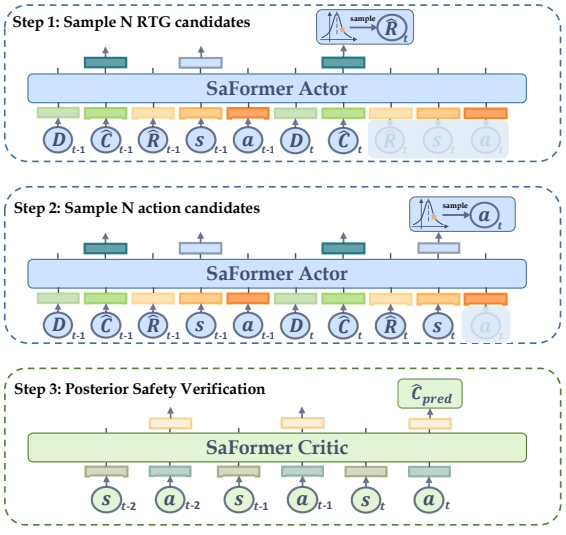


Figure 1: The schema of SaFormer.

Feasible Action Proposal Considering the cost constraints in offline safe RL, we propose two cost-related tokens, namely the cost-to-go (CTG) $\hat{C}_t = \sum_{t'=t}^{T} c_{t'}$ and the cost limit $D = d$.

The CTG accounts for residual costs that the agent can still afford from now on, which is crucial to condition the action. Even though the update of CTG is computable during the execution since the initial value is equal to the constraint threshold at the beginning of an episode, we still expect that predicting CTGs as an auxiliary task in the training process is beneficial for learning better

representations. The motivation comes from the fact that it is equivalent to learning the cost function considering that $c_t = \hat{C}_{t+1} - \hat{C}_t$.

The cost limit D is the constraint threshold and held constant throughout the sequence. We keep the token D instead of feeding CTG only since $D - \hat{C}_t$ indicates the cumulative cost incurred before. It contains past behavioral information that better conditions future sequence generation for a more general partially observable Markov decision process.

Under the assumption that the cost limit token D, the CTG token $\hat{C}_t$, and the past subsequence $\{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1}$ are sufficient to determine the current RTG target, SaFormer generates RTG distribution at timestep t as follows:

$$\hat{R}_t \sim \mathcal{N}(\mu_\theta, \Sigma_\theta | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1} \cup \{D, \hat{C}_t\}). \tag{3}$$

After sampling the target from the RTG distribution, we then generate the action distribution next based on the past sequence plus the future reward expectation $\hat{R}_t$ and the visiting state s_t as follows:

$$\hat{a}_t \sim \mathcal{N}(\mu'_\theta, \Sigma'_\theta | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1} \cup \{D, \hat{C}_t, \hat{R}_t, s_t\}). \tag{4}$$

The above probabilistic mapping is more reasonable than a deterministic model since the agent can perform different actions to achieve the same cost return, which yields a wide range of feasible RTGs under a definite CTG. Apart from that, it enables SaFormer to optimize CTGs, RTGs, and actions within a single auto-regressive model. Tweaking multi-objective ℓ_2 loss function in eq. (2) is cumbersome considering that the scales of the above tokens vary significantly among different scenarios. Instead, we keep them aligned via a multivariate independent Gaussian distribution $P(\hat{C}_t, \hat{R}_t, a_t | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1})$ and minimize the negative log-likelihood (NLL) objective $J(\theta)$ as follows:

$$\begin{aligned} J(\theta) = -\frac{1}{K+1}\mathbb{E}_{\mathcal{D}} \sum_{k=0}^{K} \big[& \log \pi_\theta\big(\hat{C}_{t-k} | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1-k} \cup D\big) \\ & + \log \pi_\theta\big(\hat{R}_{t-k} | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1-k} \cup \{D, \hat{C}_{t-k}\}\big) \\ & + \log \pi_\theta\big(a_{t-k} | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1-k} \cup \{D, \hat{C}_{t-k}, \hat{R}_{t-k}, s_t\}\big)\big]. \end{aligned} \tag{5}$$

Another attractive point of this design is that the stochastic action distribution naturally fits in with online exploration techniques (Zheng et al., 2022) and has the potential to transfer to unexplored constraints beyond the offline dataset, which we will discuss in section 4.

Posterior Safety Verification The cost-conditioned sequence generation is effective in producing near-optimal solutions. Nevertheless, we argue that it is impractical to solely rely on naive forward-computing to satisfy the hard constraint. Instead, the unidirectional auto-regression in our approach only nominates the potential candidates, and an additional transformer-based critic ζ_ϕ is proposed to enforce safety explicitly. Although the optimization of ζ_ϕ is irrelevant with Bellman backups in the conventional actor-critic architecture Fujimoto et al. (2019); Wang et al. (2020), we inherit the name "critic" since it evaluates the cost that the agent may incur in the future. The critic ζ_ϕ is updated via causal sequence modeling, and the self-supervised objective is equal to the CTG in the dataset:

$$\widetilde{J}(\phi) = \frac{1}{K+1}\mathbb{E}_{\mathcal{D}} \sum_{k=0}^{K} \big(\hat{C}_{t-k} - \zeta_\phi(\{s_i, a_i\}_{t-K}^{t-k})\big)^2. \tag{6}$$

In our implementation, we introduce the regularization terms into the objective function as:

$$J(\phi) = \widetilde{J}(\phi) + \lambda \cdot \sum_{k=0}^{K-1} [\zeta_\phi(\{s_i, a_i\}_{t-K}^{t-k}) - \zeta_\phi(\{s_i, a_i\}_{t-K}^{t-k-1})]^+. \tag{7}$$

The penalized objective leads to more reasonable representations since the cost signal c is always non-negative, and the sequence of CTGs is monotonically decreasing.

Consequently, the critic ζ_ϕ serves as the CTG estimator and conducts a posterior safety verification after the actor π_θ has proposed feasible action candidates. Details of online execution and offline training for SaFormer are summarized in appendix B Algorithms 1 and 2, respectively.

Online Fine-tuning for OOD Constraints SaFormer treats different cost limits as contextual tokens; thus, it is applicable to the in-range constraint values of the existing dataset without retraining. It also shows the promise for constraint satisfaction beyond the offline data distribution considering the impressive generalization ability of Transformer architecture (Vaswani et al., 2017). fig. 4 shows that the final cost return strongly correlates with the specified cost limit and monotonically decreases when the threshold is tightened. Nevertheless, pure offline training might be insufficient for out-of-distribution (OOD) constraints when the collected data is limited.

We further propose an online fine-tuning algorithm inspired by Zheng et al. (2022). Different from their purpose for boosting reward, we aim to fine-tune SaFormer for the tightened OOD constraints. Specifically, we reformulate the naive NLL objective in eq. (5) under the MaxEnt RL framework (Haarnoja et al., 2018) to benefit the exploration, which boils down to the following constrained optimization:

$$\min_{\theta} J(\theta) \quad \text{s.t. } H(\theta) \geq \beta. \tag{8}$$

$H(\theta)$ denotes the Shannon entropy of the action distribution , which is defined as:

$$H(\theta) = \frac{1}{K+1}\mathbb{E}_{\mathcal{D}} \sum_{k=0}^{K} H\left[\pi_\theta\left(a_{t-k}|\{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-k}\right)\right]. \tag{9}$$

Notably, we employ an attenuation factor α to decrease the cost limit every time SaFormer interacts with the environment. It enables the acquisition of heterogeneous samples to speed up the training, especially when the expected constraint is difficult to satisfy with the initial pre-trained SaFormer. Nevertheless, we use the actual cost return as the value of token D when we relabel the newly collected trajectory. Furthermore, we simply aggregate the dataset with new samples instead of discarding the earliest trajectory Zheng et al. (2022) in order to maintain the previously learned constraints. Details of online fine-tuning for SaFormer are summarized in appendix B Algorithm 3.

5 Experiments

5.1 Experiment Setup

Dataset We leverage the D4RL dataset Fu et al. (2020) over three Mujoco tasks, namely `Hopper`, `Walker2d`, and `HalfCheetah`. Note that the behaviour policies are arbitrary and cost-agnostic when collecting the samples; the existing offline data is relabeled hind-sightly according to certain cost criteria. This paradigm is of great practical relevance in data reuse and can be more flexible for varying online safety requirements compared with the prior work Xu et al. (2022a); Polosky et al. (2022). In previous study, the datasets are constructed by two types of behaviour policies: the unsafe one is trained with general RL methods; the safe one is trained via constrained RL adhering to a pre-defined constraint. Such the setting has several limitations: First, the cost distribution will center on two widely separated peaks, which is relatively easy to identify and thus can be simply solved by behavior cloning. Second, the agent can only learn a stationary constraint determined by the safe behavior policy. Most significantly, there would be a chicken-and-egg problem in practice for obtaining the constraint-satisfying policy in advance to construct the offline dataset.

Task Our safety consideration is to prolong the lifespan of motors. Therefore, we specify the cost as $c_t = \sum_{i=1}^{M} |a_t^i|$ standing for the total torque applied to the M joints, and then we limit the cumulative energy consumption by enforcing the episodic constraint $C_\pi = \sum_{t=0}^{T} \gamma^t c_t \leq d$. We conduct experiments with $\gamma_c = 0.99$ and $\gamma_c = 1$, respectively. The discounted setting is widely adopted in conventional TD-learning algorithms and also fits in with SaFomer if the cost is relabeled as $c_t = \gamma_c^t c_t$; the undiscounted setting is more challenging but practical since the torque applied in the first and last steps are equally weighed for the lifespan of motors. As for the assignment of d, we sort the trajectory-wise cost return C_{π_β} on each dataset and uniformly specify the $10\%, 20\%, 30\%, 50\%$ percentiles as different thresholds. Readers can refer to appendix A for the distributions of C_{π_β} and calculations of d regarding all the nine datasets.

Baselines We compare our method with the three kinds of representative baselines:

- DT (Chen et al., 2021): Decision Transformer with manually assigned RTGs is regarded as the unconstrained counterpart. We set a constant initial RTG = $3600, 5000, 6000$ for each dataset of `Hopper`, `Walker2d`, and `HalfCheetah`, respectively.

- BCQ-L: BCQ-L extends BCQ (Fujimoto et al., 2019) with Lagrangian relaxation to enforce constraints. It is a naive combination of offline RL and safe RL.
- CPQ (Xu et al., 2022a): CPQ is the state-of-the-art algorithm in offline safe RL. It addresses a stationary constraint via conservative Q-Learning, instead of conditional sequence generation.

Implementation Details Due to the limited space, we place hyper-parameter lists in appendix B table 6. The source code is available in the supplementary material.

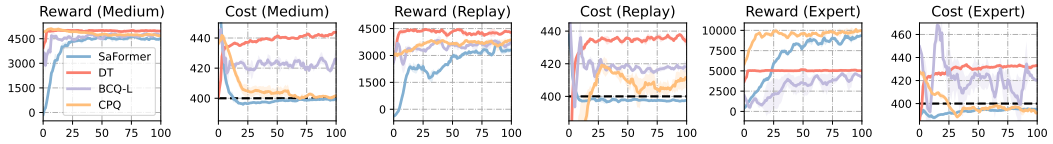


Figure 2: Online evaluation curves on HalfCheetah datasets under the discounted setting ($\gamma_c = 0.99$). The x-axis denotes the number of offline training epochs (5×10^3 iterations per epoch). The y-axis denotes the reward or cost return. The dashed line denotes the cost limit.

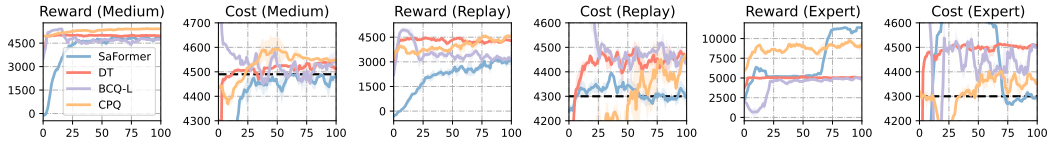


Figure 3: Online evaluation curves on HalfCheetah datasets under the undiscounted setting ($\gamma_c = 1$). The x-axis denotes the number of offline training epochs (5×10^3 iterations per epoch). The y-axis denotes the reward or cost return. The dashed line denotes the cost limit.

5.2 Baseline Comparisons

In this experiment, the broad applicability of our method is highlighted. Empirically, SaFomer is competitive with state-of-the-art baselines in the discounted setting, and still performs well in the undiscounted setting where all the baselines fail to satisfy the constraints, as shown in table 1.

Dataset		$\gamma_c = 0.99$				$\gamma_c = 1$			
		SaFormer	BCQ-L	CPQ	DT	SaFormer	BCQ-L	CPQ	DT
HalfCheetah Medium	Reward	4519.21 ± 52.75	4734.96 ± 98.13	**4727.55 ± 36.41**	4986.64 ± 5.56	**4661.20 ± 52.46**	4729.00 ± 59.52	5411.98 ± 55.20	4986.64 ± 5.56
	Cost	398.86 ± 0.62	421.68 ± 3.36	400.83 ± 0.79	441.89 ± 0.21	4487.2 ± 6.05	4529.94 ± 34.25	4548.34 ± 11.07	4531.89 ± 1.45
	Limit	400	400	400	–	4490	4490	4490	–
HalfCheetah Medium_Replay	Reward	**3321.55 ± 101.17**	3603.96 ± 144.22	3731.19 ± 163.03	4270.07 ± 16.11	**3019.17 ± 234.09**	3191.36 ± 101.62	4317.85 ± 192.30	4270.07 ± 16.11
	Cost	397.41 ± 0.43	417.86 ± 1.98	409.59 ± 5.93	436.16 ± 1.87	4289.54 ± 6.31	4443.87 ± 49.28	4408.20 ± 51.61	4448.84 ± 6.8
	Limit	400	400	400	–	4300	4300	4300	–
HalfCheetah Medium_Expert	Reward	9124.51 ± 78.08	4642.31 ± 234.34	**9870.24 ± 169.63**	5053.89 ± 30.17	**11000.37 ± 141.93**	4902.81 ± 131.39	9221.09 ± 229.64	5053.89 ± 30.17
	Cost	395.17 ± 0.79	425.13 ± 9.84	393.91 ± 1.57	429.75 ± 0.37	4296.87 ± 9.92	4468.96 ± 50.47	4369.97 ± 18.63	4478.89 ± 3.52
	Limit	400	400	400	–	4300	4300	4300	–

Table 1: Baseline comparison results on HalfCheetah datasets. The costs marked in satisfy the constraint; the costs marked in violate the constraint and the corresponding policies are deemed infeasible. We use the same notation in the rest part of the paper.

We first evaluate their performances under the discounted setting with threshold $d = 400$ across the different Halfcheetah datasets.The results show that SaFormer adheres to the cost limit more strictly than others and achieves competitive reward returns simultaneously. On the contrary, BCQ-L suffers from oscillation in the learning process due to the changing Lagrangian multipliers and hardly satisfies the constraints. Despite one case of constraint violation, CPQ outperforms SaFormer in terms of cumulative rewards due to the explicit policy optimization based on Q functions.

We then perform a comparative evaluation under the undiscounted setting and apply a uniform threshold $d = 4300$ across different tasks. Note that, we change the cost limit to 4490 in the HalfCheeta_medium dataset since the original value is completely out of the offline data distribution, and we will present the experiments in this circumstance later. The results show that only the proposed SaFormer still adheres to the constraints. By contrast, BCQ-L and CPQ are extremely unstable and yield infeasible policies when $\gamma_c = 1$, which is the inherent issue of Q-Learning.

It's worth noting that DT converges to a sub-optimal solution on the HalfCheetah_medium_expert dataset, while SaFormer can avoid the tramp and achieve significantly better performance with a

lower cost return. To some extent, it reveals that SaFormer is not a simple behaviour cloning conditioned by cost but searches for optimal actions in the constrained space.

5.3 Offline Constraint Adaptation

Dataset		10%Percentile	20%Percentile	30%Percentile	50%Percentile	DT(Reference)
HalfCheetah Medium	Reward	4661.2 ± 52.46	4713.52 ± 26.83	4718.44 ± 22.09	4724.89 ± 27.18	4986.64 ± 5.56
	Cost	4487.2 ± 6.05	4498.22 ± 2.57	4500.52 ± 1.65	4508.83 ± 2.79	4531.89 ± 1.45
	Limit	4490	4503	4511	4526	–
HalfCheetah Medium_Replay	Reward	2202.69 ± 148.39	2208.41 ± 90.14	3363.11 ± 165.89	3712.08 ± 104.74	4270.07 ± 16.11
	Cost	4203.0 ± 17.46	4234.15 ± 13.36	4381.42 ± 16.94	4411.84 ± 9.29	4448.84 ± 6.8
	Limit	3928	4257	4422	4493	–
HalfCheetah Medium_Expert	Reward	9016.82 ± 124.45	9121.95 ± 397.5	9161.5 ± 385.65	9296.29 ± 249.52	5053.89 ± 30.17
	Cost	4181.25 ± 16.63	4203.95 ± 2.36	4211.91 ± 9.08	4229.4 ± 17.35	4478.89 ± 3.52
	Limit	4201	4215	4224	4266	–

Table 2: Experiments on constraint adaptation. We train SaFormer on certain dataset and exploit different cost limit token in the execution. A full version of the results on all datasets can be found in appendix C table 7.

In this experiment, we demonstrate that SaFormer is robust and flexible against varying in-range constraint thresholds of the offline data without retraining, which is of great attraction to real-world applications.

We train SaFormer on the fixed dataset and use the $10\%, 20\%, 30\%, 50\%$ percentiles of the cost return of the offline trajectories as contextual tokens to evaluate its applicability toward different safety requirements. table 2 reports its constraint adaptation performance when pre-trained on HacfCheetah datasets. SaFormer satisfies 11 of 12 cost limits in the experiment, which holds a 91.6% constraint satisfaction rate. In general, SaFormer can identify different cost limits and is able to yield feasible solutions accordingly.

The only instance of failure in all 12 cases is when using the 10% percentile of HalfCheetah_medium_replay dataset as the threshold ($d = 3928$). Similar situations also occur in the other two environments. A possible reason is that the samples in the replay dataset (202 rollouts) are very limited compared with the medium and expert datasets (> 1000 rollouts). Thus, the constrained action space might be under-explored for our Transformer-based architecture.

5.4 Online Constraint Fine-tuning

In this experiment, we apply algorithm 3 to deal with the under-explored constraints in the aforementioned dataset and show the efficacy of the online fine-tuning technique against those out-of-distribution (OOD) constraints.

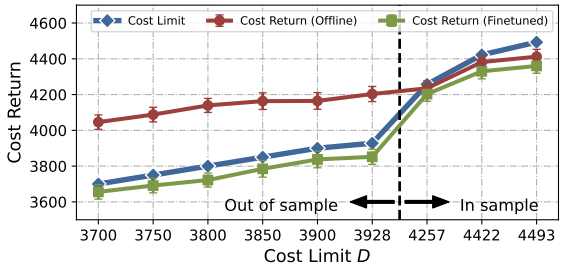


Figure 4: Cost returns of SaFormer after offline pre-training (red) and online fine-tuning (green) with respect to a range of expected cost limits (blue) of the HalfCheetah_medium_replay dataset.

At first, we discard samples whose cost returns are less than the previous unmet 10% percentile ($d = 3928$) and regard such thresholds as OOD constraints. Then, we compare the generalizablity of SaFormer via offline pre-training and online fine-tuning. table 3 shows the mean cost return of the above two types of SaFormer with respect to both in-sample and out-of-sample constraints. We decrease the out-of-sample constraint threshold at equal intervals and witness a drop in the final cost return achieved by offline SaFormer. Nevertheless, the OOD constraints are hardly satisfied via pure offline training. By contrast, the online SaFormer shrinks the expected cost limits to generate new conservative samples and reduces the overall cost below the corresponding thresholds after fine-tuning. The online sample consumption is acceptable and we insert 200 new trajectories to satisfy all the constraints listed in table 3.

COST LIMIT D	OUT OF SAMPLE					IN SAMPLE			
	3700	3750	3800	3850	3900	3928	4257	4422	4493
AVG COST (OFFLINE)	4045.92	4088.0	4139.33	4163.35	4164.42	4203.0	4234.15	4381.42	4411.84
AVG COST (FINETUNED)	3655.93	3691.42	3722.03	3784.85	3837.36	3852.83	4202.85	4330.46	4360.21

Table 3: Experiments on constraint generalizability. We report the mean cost return of SaFormer after offline pre-trained on the HalfCheetah_medium_replay dataset and online fine-tuned with respect to a range of expected cost limits.

DATASET		20%PERCENTILE		30%PERCENTILE		50%PERCENTILE	
		ζ_ϕ (✓)	ζ_ϕ (✗)	ζ_ϕ (✓)	ζ_ϕ (✗)	ζ_ϕ (✓)	ζ_ϕ (✗)
HALFCHEETAH MEDIUM_EXPERT	REWARD	9121.95 ± 397.5	11441.5 ± 12.35	9161.5 ± 385.65	11445.21 ± 27.3	9296.29 ± 249.52	11460.68 ± 21.44
	COST	4203.95 ± 2.36	4304.72 ± 2.18	4211.91 ± 9.08	4308.48 ± 1.58	4229.4 ± 17.35	4305.78 ± 2.87
	LIMIT	4215	4215	4224	4224	4266	4266
WALKER2D MEDIUM_EXPERT	RETURN	3076.96 ± 101.41	3597.44 ± 179.95	4278.38 ± 302.14	3913.53 ± 302.42	4479.84 ± 323.36	3989.87 ± 145.14
	COST	2989.44 ± 75.12	3518.32 ± 159.18	3567.39 ± 174.72	3576.96 ± 214.27	3665.75 ± 231.09	3667.21 ± 90.22
	LIMIT	3154	3154	3745	3745	3778	3778
HOPPER MEDIUM_EXPERT	REWARD	1191.74 ± 30.2	1671.21 ± 41.19	1312.42 ± 25.91	1663.75 ± 30.31	1467.81 ± 41.64	1677.25 ± 50.03
	COST	639.62 ± 15.19	756.71 ± 18.93	696.03 ± 14.93	761.01 ± 12.35	732.89 ± 21.21	777.59 ± 25.22
	LIMIT	683	683	742	742	856	856

Table 4: Ablation study on ζ_ϕ. We report the performance of SaFormer with and without the posterior safety verification at three different thresholds (20%,30%,50% percentiles, respectively). A full version of the results on all datasets can be found in appendix C table 8.

5.5 ABLATION STUDY

Posterior safety verification We first investigate the necessity of the critic ζ_ϕ, which evaluates the long-term cost return and filters out unsafe actions. table 4 reports the performance of SaFormer at 20%, 30% and 50% percentiles of in-range thresholds with and without the posterior safety verification, respectively. The results show that SaFormer fulfills the hard constraints well when it is equipped with ζ_ϕ. On the contrary, the naive actor π_θ will only lead to near constraint-satisfying policies and still holds the 55.6% of the proportion that violates the constraint. This phenomenon is more pronounced when the constraint is tightened. Furthermore, even if the actor can directly yield feasible trajectories under certain constraints, their cumulative rewards are inferior to SaFormer with posterior safety verification. The reason is that SaFormer samples a batch of candidates and executes the one with the highest RTG by leveraging ζ_ϕ, which may boost the reward performance while preserving the feasibility.

Proposal batchsize we study the effect of RTG and action proposal batchsize N. We evaluate the constraint satisfaction at 20%, 30%, and 50% percentiles of in-range thresholds ($d = 683, 742, 856$, respectively) on the Hopper_medium_expert dataset with respect to different N. fig. 5 confirms that the cost return is steady and constraint-satisfying when the actor proposes a large batch of candidates. By contrast, SaFormer may exceed the cost limit during the execution when the batch size N is relatively small. It is concluded that the mapping from cost to reward is not a one-to-one function and requires a wide range of samples to search for the optimal solution. The eventual returns converge if the batchsize is sufficiently large.

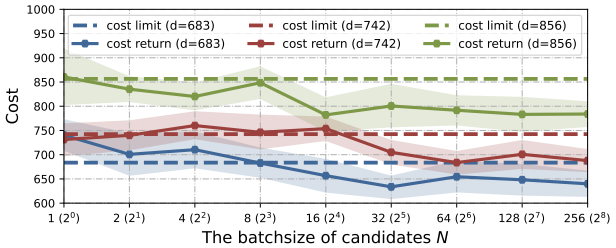


Figure 5: Sensitivity study on RTG and action proposal batchsize N. The curve denotes the eventual cost return with respect to different N. The dashed line in the corresponding color denotes the cost limit of the Hopper_medium_expert dataset.

6 CONCLUSION

We present SaFormer in this paper, which, to the best of our knowledge, is the first sequence modeling approach to offline safe RL. Casting constraints as contextual tokens, SaFormer is competitive with state-of-the art algorithms in terms of reward performance, but more robust and flexible toward varying safety requirements. We believe such the properties is of great practical relevance in real-world problems. As the future work, more complicated offline datasets and tasks are required to better evaluate the proposed approach. Besides, we are also dedicated to overcome the fundamental assumptions in the problem setup to enhance SaFormer's risk-awareness and extend SaFormer to multi-constraint scenarios to reduce its limitations in safety-critical tasks.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

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.

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 *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. On transforming reinforcement learning by transformer: The development trajectory. *arXiv preprint arXiv:2212.14164*, 2022.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

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*, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Nicholas Polosky, Bruno C Da Silva, Madalina Fiterau, and Jithin Jagannath. Constrained offline policy optimization. In *International Conference on Machine Learning*, pp. 17801–17810. PMLR, 2022.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. *arXiv preprint arXiv:2206.08569*, 2022.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

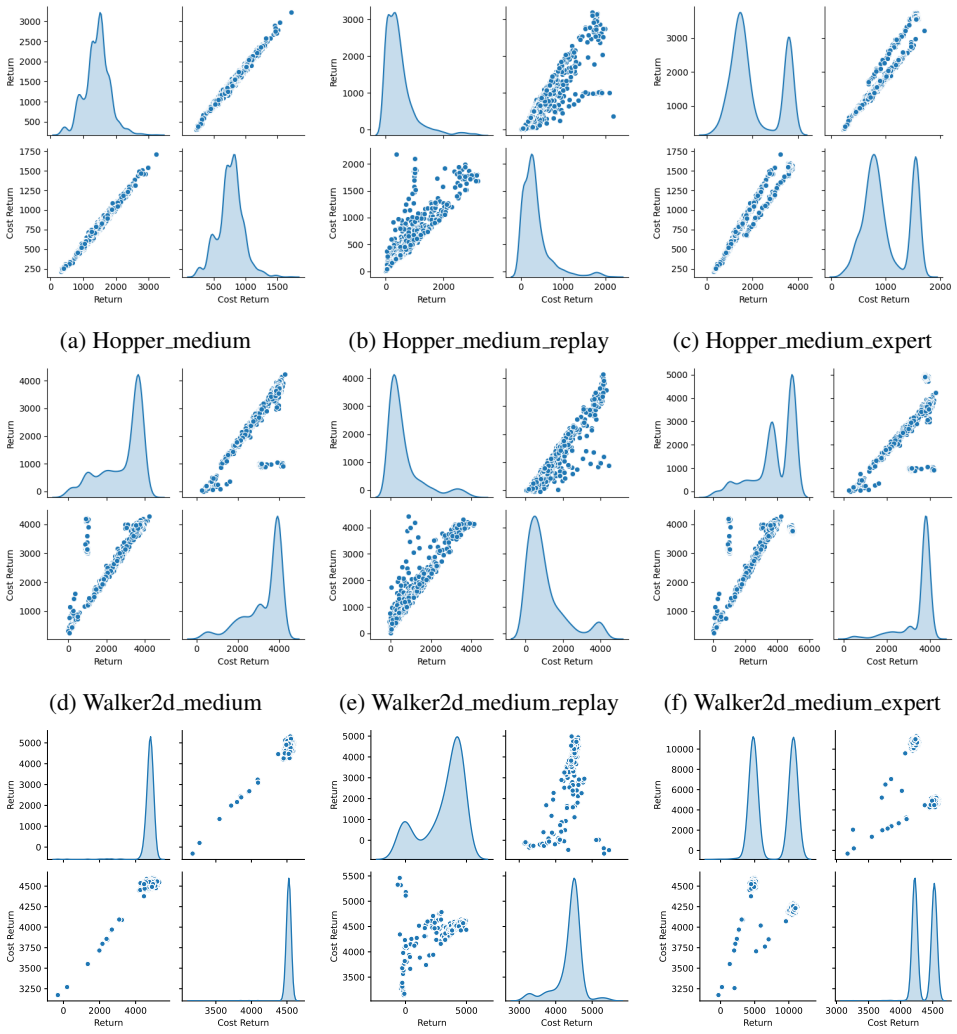
Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022a.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pp. 24631–24645. PMLR, 2022b.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *International Conference on Machine Learning*, pp. 27042–27059. PMLR, 2022.

A DATASET AND TASK VISUALIZATIONS



(a) Hopper_medium (b) Hopper_medium_replay (c) Hopper_medium_expert

(d) Walker2d_medium (e) Walker2d_medium_replay (f) Walker2d_medium_expert

(g) HalfCheetah_medium (h) HalfCheetah_medium_replay (i) HalfCheetah_medium_expert

Figure 6: Pair-plots of reward and cost returns on the D4RL datasets.

DATASET	MIN	10%PERCENTILE	20%PERCENTILE	30%PERCENTILE	50%PERCENTILE	MAX
HALFCHEETAH_MEDIUM	3172	4490	4503	4511	4526	4600
HALFCHEETAH_MEDIUM_REPLAY	3145	3928	4257	4422	4493	5461
HALFCHEETAH_MEDIUM_EXPERT	3172	4201	4215	4224	4266	4600
WALKER2D_MEDIUM	246	1877	2450	3043	3839	4283
WALKER2D_MEDIUM_REPLAY	8	75	251	432	705	4432
WALKER2D_MEDIUM_EXPERT	246	2343	3154	3745	3778	4283
HOPPER_MEDIUM	218	483	618	685	772	1710
HOPPER_MEDIUM_REPLAY	2	35	90	184	268	2186
HOPPER_MEDIUM_EXPERT	218	546	683	742	856	1710

Table 5: Threshold settings (10%,20%,30%,50% percentiles) on the D4RL datasets.

B REPRODUCIBLITY

The hyper-parameters are listed below.

HYPER-PARAMETERS	SAFORMER	DT	BCQ-L	CPQ
SUBSEQUENCE LENGTH K	20	20	/	/
NUMBER OF ATTENTION BLOCKS	3	3	/	/
EMBEDDING DIMENSION	128	128	/	/
HIDDEN LAYER	128	128	(256,256)	(256,256)
DROPOUT RATIO	0.1	0.1	0.1	0.1
TRAINING BATCHSIZE	128	128	128	128
LEARNING RATE	10E-4	10E-4	10E-4	10E-4
LEARNING RATE DECAY	10E-4	10E-4	10E-4	10E-4
PROPOSAL BATCHSIZE N	128	/	/	/
PENALTY FACTOR λ	0.25	/	/	/
ATTENUATION FACTOR α	0.95	/	/	/
INITIAL LAGRANGIAN MULTIPLIER	/	/	0.1	/
LAGRANGIAN MULTIPLIER LEARNING RATE	/	/	3E-4	/
POLYAK AVERAGING FACTOR τ	/	/	0.05	0.05
OOD PENALTY α XU ET AL. (2022A)	/	/	/	5.0
VAE PENALTY β XU ET AL. (2022A)	/	/	/	1.5

Table 6: Hyper-parameter lists for baseline comparisons.

C EMPIRICAL DETAILS

DATASET		10%PERCENTILE	20%PERCENTILE	30%PERCENTILE	50%PERCENTILE	DT(REFERENCE)
HALFCHEETAH MEDIUM	REWARD	4661.2 ± 52.46	4713.52 ± 26.83	4718.44 ± 22.09	4724.89 ± 27.18	4986.64 ± 5.56
	COST	4487.2 ± 6.05	4498.22 ± 2.57	4500.52 ± 1.65	4508.83 ± 2.79	4531.89 ± 1.45
	LIMIT	4490	4503	4511	4526	–
HALFCHEETAH MEDIUM_REPLAY	REWARD	2202.69 ± 148.39	2208.41 ± 90.14	3363.11 ± 165.89	3712.08 ± 104.74	4270.07 ± 16.11
	COST	4203.0 ± 17.46	4234.15 ± 13.36	4381.42 ± 16.94	4411.84 ± 9.29	4448.84 ± 6.8
	LIMIT	3928	4257	4422	4493	–
HALFCHEETAH MEDIUM_EXPERT	REWARD	9016.82 ± 124.45	9121.95 ± 397.5	9161.5 ± 385.65	9296.29 ± 249.52	5053.89 ± 30.17
	COST	4181.25 ± 16.63	4203.95 ± 2.36	4211.91 ± 9.08	4229.4 ± 17.35	4478.89 ± 3.52
	LIMIT	4201	4215	4224	4266	–
WALKER2D MEDIUM	REWARD	2654.84 ± 210.72	2752.82 ± 161.99	2896.36 ± 72.88	3016.77 ± 139.35	3325.07 ± 94.88
	COST	3038.95 ± 222.47	3191.76 ± 138.93	3224.36 ± 67.41	3383.85 ± 121.79	3360.02 ± 99.78
	LIMIT	1877	2450	3043	3839	–
WALKER2D MEDIUM_REPLAY	REWARD	32.54 ± 20.95	155.82 ± 36.2	238.5 ± 20.42	318.8 ± 42.47	2194.73 ± 176.7
	COST	124.55 ± 39.66	417.45 ± 105.91	585.19 ± 28.27	682.98 ± 31.45	2259.91 ± 198.37
	LIMIT	75	251	432	705	–
WALKER2D MEDIUM_EXPERT	REWARD	2938.5 ± 140.71	3076.96 ± 101.41	4278.38 ± 302.14	4479.84 ± 323.36	4972.82 ± 1.01
	COST	3160.32 ± 130.84	2989.44 ± 75.12	3567.39 ± 174.72	3665.75 ± 231.09	3569.08 ± 0.41
	LIMIT	2343	3154	3745	3778	–
HOPPER MEDIUM	REWARD	862.21 ± 17.59	1037.53 ± 21.69	1090.57 ± 23.52	1245.57 ± 25.04	2018.02 ± 79.05
	COST	477.39 ± 9.12	568.83 ± 12.04	599.08 ± 10.59	671.1 ± 13.18	907.24 ± 38.19
	LIMIT	483	618	685	772	–
HOPPER MEDIUM_REPLAY	REWARD	39.28 ± 6.35	71.26 ± 14.0	244.93 ± 32.71	351.12 ± 59.37	904.97 ± 193.08
	COST	50.03 ± 3.95	89.52 ± 14.47	207.42 ± 13.25	259.67 ± 11.71	346.6 ± 73.01
	LIMIT	35	90	184	268	-
HOPPER MEDIUM_EXPERT	REWARD	986.29 ± 29.72	1191.74 ± 30.2	1312.42 ± 25.91	1467.81 ± 41.64	3274.14 ± 176.99
	COST	535.13 ± 18.88	639.62 ± 15.19	696.03 ± 14.93	732.89 ± 21.21	1113.92 ± 65.25
	LIMIT	546	683	742	856	–

Table 7: Full experiment results of constraint adaptation.

DATASET		20%PERCENTILE		30%PERCENTILE		50%PERCENTILE	
		ζ_ϕ (✓)	ζ_ϕ (✗)	ζ_ϕ (✓)	ζ_ϕ (✗)	ζ_ϕ (✓)	ζ_ϕ (✗)
HALFCHEETAH MEDIUM	REWARD	4611.15 ± 74.75	4884.18 ± 166.87	4604.47 ± 110.94	4883.92 ± 170.97	4724.89 ± 27.18	4968.35 ± 167.89
	COST	4478.43 ± 12.5	4590.93 ± 75.71	4477.93 ± 20.45	4580.49 ± 93.37	4508.83 ± 2.79	4614.53 ± 74.26
	LIMIT	4503	4503	4511	4511	4526	4526
HALFCHEETAH MEDIUM_REPLAY	REWARD	2208.41 ± 90.14	3807.5 ± 72.0	3363.11 ± 165.89	3646.36 ± 42.99	3712.08 ± 104.74	3470.89 ± 185.94
	COST	4234.15 ± 13.36	4578.56 ± 6.62	4381.42 ± 16.94	4596.31 ± 11.27	4411.84 ± 9.29	4579.14 ± 57.62
	LIMIT	4257	4257	4422	4422	4493	4493
HALFCHEETAH MEDIUM_EXPERT	REWARD	9121.95 ± 397.5	11441.5 ± 12.35	9161.5 ± 385.65	11445.21 ± 27.3	9296.29 ± 249.52	11460.68 ± 21.44
	COST	4203.95 ± 2.36	4304.72 ± 2.18	4211.91 ± 9.08	4308.48 ± 1.58	4229.4 ± 17.35	4305.78 ± 12.87
	LIMIT	4215	4215	4224	4224	4266	4266
WALKER2D MEDIUM	REWARD	2752.82 ± 161.99	3304.39 ± 150.29	2896.36 ± 72.88	3416.02 ± 248.65	3016.77 ± 139.35	3367.41 ± 159.17
	COST	3191.76 ± 138.93	3376.99 ± 143.7	3224.36 ± 67.41	3508.21 ± 240.09	3383.85 ± 121.79	3439.88 ± 163.23
	LIMIT	2450	2450	3043	3043	3839	3839
WALKER2D MEDIUM_REPLAY	REWARD	155.82 ± 36.2	192.46 ± 14.19	238.5 ± 20.42	314.85 ± 51.67	318.8 ± 42.47	535.65 ± 64.69
	COST	417.45 ± 105.91	482.54 ± 27.56	585.19 ± 28.27	618.22 ± 88.42	682.98 ± 31.45	873.69 ± 174.75
	LIMIT	251	251	432	432	705	705
WALKER2D MEDIUM_EXPERT	REWARD	3076.96 ± 101.41	3597.44 ± 179.95	4278.38 ± 302.14	3913.53 ± 302.42	4479.84 ± 323.36	3989.87 ± 145.14
	COST	2989.44 ± 75.12	3518.32 ± 159.18	3567.39 ± 174.72	3576.96 ± 214.27	3665.75 ± 231.09	3667.21 ± 90.22
	LIMIT	3154	3154	3745	3745	3778	3778
HOPPER MEDIUM	REWARD	1037.53 ± 21.69	1504.02 ± 31.34	1090.57 ± 23.52	1584.71 ± 60.46	1245.57 ± 25.04	1659.57 ± 65.07
	COST	568.83 ± 12.04	682.61 ± 17.72	599.08 ± 10.59	718.98 ± 28.18	671.1 ± 13.18	757.86 ± 31.93
	LIMIT	618	618	685	685	772	772
HOPPER MEDIUM_REPLAY	REWARD	71.26 ± 14.0	244.24 ± 36.01	244.93 ± 32.71	340.14 ± 21.47	351.12 ± 59.37	514.83 ± 108.93
	COST	89.52 ± 14.47	241.02 ± 25.91	207.42 ± 13.25	281.49 ± 12.8	259.67 ± 11.71	290.89 ± 46.99
	LIMIT	90	90	184	184	268	268
HOPPER MEDIUM_EXPERT	REWARD	1191.74 ± 30.2	1671.21 ± 41.19	1312.42 ± 25.91	1663.75 ± 30.31	1467.81 ± 41.64	1677.25 ± 50.03
	COST	639.62 ± 15.19	756.71 ± 18.93	696.03 ± 14.93	761.01 ± 12.35	732.89 ± 21.21	777.59 ± 25.22
	LIMIT	683	683	742	742	856	856

Table 8: Full experiment results of ablation study.

D ALGORITHMS

Algorithm 1 SaFormer Policy Execution

Require: Online Env, Actor π_θ, Critic ζ_ϕ, cost limit d.
Initialize $D = \hat{C}_0 = d$.
for $t = 0$ **to** maximum episode horizon T **do**
  `#Feasible Action Proposal`
  SaFormer actor π_θ generates the RTG distribution $\mathcal{N}(\mu_\theta, \Sigma_\theta | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1} \cup \{D, \hat{C}_t\})$.
  Sample N RTG candidates $\hat{R}_t^n, n = 1, 2, ..., N$.
  SaFormer actor π_θ generates N action distributions $\mathcal{N}(\mu_\theta, \Sigma_\theta | \{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^{t-1} \cup \{D, \hat{C}_t, \hat{R}_t^n, s_t\})$.
  Sample N action candidates $a_t^n, n = 1, 2, ..., N$.
  `#Posterior Safety Verification`
  SaFormer critic ζ_ϕ predicts the long-term cost-return of each pair $< \hat{R}_t^n, a_t^n >, n = 1, 2, ..., N$.
  if $\exists n, \zeta_\phi(\{s_i, a_i^n\}_{t-K}^t) \leq \hat{C}_t$ **then**
    $a_t = \arg\max(\hat{R}_t^n | \zeta_\phi(\{s_i, a_i^n\}_{t-K}^t) \leq \hat{C}_t)$
  else
    Reject and re-sample $< \hat{R}_t^n, a_t^n >, n = 1, 2, ..., N$.
  end if
  Update $s_t, r_t, c_t = Env.step(a_t)$ and $\hat{C}_t = \hat{C}_t - c_t$.
end for

Algorithm 2 SaFormer Offline Training

Require: Offline dataset $\mathcal{D}$, Actor π_θ, Critic ζ_ϕ
repeat
  for $iter = 0$ **to** maximum iteration M **do**
    Sample a mini-batch $\{D, \hat{C}_i, \hat{R}_i, s_i, a_i\}_{t-K}^t \sim \mathcal{D}$.
    Optimize SaFormer actor π_θ via minimizing Eq equation 5.
    Optimize SaFormer critic ζ_ϕ via minimizing Eq equation 7.
  end for
  Policy evaluation using algorithm 1.
until the offline training terminates.

Algorithm 3 SaFormer Online Fine-tuing

Require: Online Env, Offline dataset $\mathcal{D}$, Actor π_θ, Critic ζ_ϕ, OOD cost limit d. Attenuation factor $\alpha \in (0, 1)$.
repeat
  Set $d = \alpha d$ as the target cost limit.
  Rollout trajectory τ using algorithm 1.
  Relabel τ as $\hat{R}_t = \sum_{t'=t}^T r_{t'}, \hat{C}_t = \sum_{t'=t}^T c_{t'}, D = \hat{C}_0$.
  Aggregate the offline data $\mathcal{D} = \mathcal{D} \cup \tau$.
  Fine-tune SaFormer π_θ and ζ_θ using algorithm 2.
until the online fine-tuning terminates.