Constrained Decision Transformer for Offline Safe Reinforcement Learning

Zuxin Liu [*1] **Zijian Guo** [*1] **Yihang Yao** [1] **Zhepeng Cen** [1] **Wenhao Yu** [2] **Tingnan Zhang** [2] **Ding Zhao** [1]

Abstract

Safe reinforcement learning (RL) trains a constraint satisfaction policy by interacting with the environment. We aim to tackle a more challenging problem: learning a safe policy from an offline dataset. We study the offline safe RL problem from a novel multi-objective optimization perspective and propose the ϵ-reducible concept to characterize problem difficulties. The inherent trade-offs between safety and task performance inspire us to propose the constrained decision transformer (CDT) approach, which can dynamically adjust the trade-offs during deployment. Extensive experiments show the advantages of the proposed method in learning an adaptive, safe, robust, and high-reward policy. CDT outperforms its variants and strong offline safe RL baselines by a large margin with the same hyperparameters across all tasks, while keeping the zero-shot adaptation capability to different constraint thresholds, making our approach more suitable for real-world RL under constraints.

1. Introduction

Learning high-reward policies from offline datasets has been a prevalent topic in reinforcement learning (RL) and has shown great promise in broad applications (Fu et al., 2020; Prudencio et al., 2022). Various learning paradigms are proposed to extract as much information as possible from pre-collected trajectories while preventing the policy from overfitting (Kostrikov et al., 2021; Sinha et al., 2022). However, in the real world, many tasks can hardly be formulated by solely maximizing a scalar reward function, and the existence of various constraints restricts the domain of feasible solutions (Gulcehre et al., 2020). For example, though numerous self-driving datasets are collected (Sun et al., 2020), it is hard to define a single reward function to describe the task (Lu et al., 2022). The optimal driving policies should satisfy a set of constraints, such as traffic laws and physical dynamics. Simply maximizing the reward may cause constraint violations and catastrophic consequences in safety-critical applications (Chen et al., 2021a).

Safe reinforcement learning aims to obtain a reward-maximizing policy within a constrained manifold (Garcıa & Fernández, 2015; Brunke et al., 2021), showing advantages to satisfy the safety requirements in real-world applications (Ray et al., 2019; Gu et al., 2022). However, most deep safe RL approaches focus on the safety during deployment, i.e., after training, while ignoring the constraint violation costs during training (Xu et al., 2022b). The requirement of collecting online interaction samples brings challenges in ensuring training safety, because it is a non-trivial task to prevent the agent from executing unsafe behaviors during the learning process. Though carefully designed correction systems or even human interventions can be used as a safety guard to filter unsafe action in training (Saunders et al., 2017; Dalal et al., 2018; Wagener et al., 2021), it could be expensive to be applied due to the low sample efficiency of many RL approaches (Xie et al., 2021).

This paper studies the problem of learning constrained policies from offline datasets such that the safety requirements can be met both in training and deployment. Several recent works tackle the problem by bridging the ideas in offline RL and safe RL domains, such as using pessimistic estimations (Xu et al., 2022a) or the stationary distribution correction technique (Liu et al., 2020; Lee et al., 2022). A constrained optimization formulation and Lagrange multipliers are usually adopted when updating the policy, targeting to find the most rewarding policy while satisfying the constraints (Le et al., 2019). However, these approaches require setting a constant constraint threshold before training, and thus the trained agents can not be adapted to other constraint conditions. We believe *the capability of adapting the trained policy to different constraint thresholds is important* for many practical applications, because imposing stricter constraints is usually at the cost of sacrificing the task performance and inducing conservative behaviors (Liu et al., 2022b). Therefore, we aim to study a training scheme such that the trained agent can dynamically adjust its constraint threshold, such that we can control its deployment conservativeness without further fine-tuning or re-training.

[*]Equal contribution [1]Carnegie Mellon University [2]Google Deepmind. Correspondence to: Zuxin Liu <zuxinl@cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

We also observe that the taxonomy of offline safe RL datasets is not adequately discussed in the literature, while we believe the characterization of a dataset can significantly influence the problem difficulty. We provide a novel view of the offline safe RL problem using tools from the multi-objective optimization (MOO) domain, which unveils the inherent trade-off between safety performance and task reward. The trade-offs can be described by a function with respect to the dataset and the constraint threshold, which inspires us to propose the Constrained Decision Transformer (CDT) approach. CDT leverages the return-conditioned sequential modeling framework (Chen et al., 2021b) to achieve zero-shot adaptation to different constraint thresholds at deployment while maintaining safety and high reward. The main contributions are summarized as follows:

- We study the offline safe RL problem beyond a single pre-defined constraint threshold from a novel MOO perspective. The insights suggest the limitations of existing offline safe RL training paradigms and motivate us to propose CDT by leveraging the return-conditioned sequential modeling capability of Transformer.
- We propose three key techniques in CDT that are important in learning an adaptive and safe policy. To the best of our knowledge, CDT is the first successful offline safe RL approach that can achieve zero-shot adaptation to different safety requirements after training, without solving a constrained optimization.
- Extensive experiments show that CDT outperforms the baselines and its variants in terms of both safety and task performance by a large margin. CDT can generalize to different cost thresholds without re-training the policy, while all the prior methods fail.

2. Related Work

Safe RL. Constrained optimization techniques are usually adopted to solve safe RL problems (Garcıa & Fernández, 2015; Sootla et al., 2022; Yang et al., 2021; Flet-Berliac & Basu, 2022; Ji et al., 2023). Lagrangian-based methods use a multiplier to penalize constraint violations (Chow et al., 2017; Tessler et al., 2018; Stooke et al., 2020; Chen et al., 2021c). Correction-based approaches project unsafe actions to the safe set, aiming to incorporate domain knowledge of the problem to achieve safe exploration (Zhao et al., 2021; Luo & Ma, 2021). Another line of work performs policy optimization on surrogate policy spaces via low-order Taylor approximations (Achiam et al., 2017; Yang et al., 2020) or variational inference (Liu et al., 2022a). However, ensuring zero constraint violations during training is still a challenging problem.

Offline RL. Offline RL targets learning policies from collected data without further interaction with the environment (Ernst et al., 2005). Many regularization and constraint methods for offline RL are proposed to address the state-action distribution shift problem between the static dataset and physical world (Levine et al., 2020; Prudencio et al., 2022). One type of approach limits the discrepancy between learned policy and behavioral policy (Fujimoto et al., 2019; Kumar et al., 2019; Peng et al., 2019; Nair et al., 2020; Fujimoto & Gu, 2021). Another way is to use value regularization as implicit constraints(Wang et al., 2020), e.g., optimizing the policy based on a conservative value estimation (Kumar et al., 2020). In addition to the above pessimism mechanism, stationary distribution correction (DICE)-style methods train the policy by importance sampling, which reduces the estimation variance (Nachum et al., 2019b;a; Zhang et al., 2020a). Recent research also shows the great success of leveraging the power of Transformer to perform behavior cloning style policy optimization (Janner et al., 2021; Chen et al., 2021b; Furuta et al., 2022).

Offline RL with safety constraints. Several recent works study the offline safe RL problem, aiming to achieve zero constraint violations during training (Le et al., 2019). They utilize the ideas from both offline RL and safe RL, such as using the DICE-style technique to formulate the constrained optimization problem (Polosky et al., 2022; Lee et al., 2022). Lagrangian-based approaches are also explored due to their simplicity of combining with existing offline RL methods, and are shown to be effective when using conservative cost estimation (Xu et al., 2022a). However, how to adapt a trained safe policy to various constraint thresholds is rarely discussed in the literature.

3. Preliminaries

3.1. CMDP and Safe RL

Safe RL can be described under the Constrained Markov Decision Process (CMDP) framework (Altman, 1998). A finite horizon CMDP $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, c, \mu_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $\mu_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution. CMDP augments MDP with an additional element $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, C_{max}]$ to characterize the cost for violating the constraint, where C_{max} is the maximum cost. Note that this work can be directly applied to multiple constraints and partially observable settings, but we use CMDP with a single constraint for ease of demonstration.

A safe RL problem is specified by a CMDP and a constraint threshold $\kappa \to [0, +\infty)$. Let $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denote the policy and $\tau = \{s_1, a_1, r_1, c_1 ..., s_T, a_T, r_T, c_T\}$ denote the trajectory, where $T = |\tau|$ is the maximum episode length. We denote $R(\tau) = \sum_{t=0}^{T-1} r_t$ as the reward return of the trajectory τ and $C(\tau) = \sum_{t=0}^{T-1} c_t$ as the cost return. The goal of safe RL is to find the policy that maximizes the re-

ward return while limiting the cost incurred from constraint violations to the threshold κ:

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi}\big[R(\tau)\big], \quad s.t. \quad \mathbb{E}_{\tau\sim\pi}\big[C(\tau)\big] \leq \kappa. \tag{1}$$

In the offline setting, the agent can not collect more data by interaction but only access pre-collected trajectories from arbitrary and unknown policies, which brings challenges to solving this constrained optimization problem.

3.2. Decision Transformer for Offline RL

Decision Transformer (DT) (Chen et al., 2021b) is a type of sequential modeling technique to solve offline RL problems, without considering the constraint in Eq. (1). Unlike classical offline RL approaches that parametrize a single state-conditioned policy $\pi(a|s)$, DT takes in a sequence of reward returns, states, and actions as input tokens, and outputs the same length of predicted actions. Given a trajectory τ of length T, the reward return at timestep t is computed by $R_t = \sum_{t'=t}^{T} r_{t'}$, then we obtain 3 types of tokens for DT: reward returns $\mathbf{R} = \{R_1, ..., R_T\}$, states $\mathbf{s} = \{s_1, ..., s_T\}$, and actions $\mathbf{a} = \{a_1, ..., a_T\}$. The input sequence for DT at timestep t is specified by a context length $K \in \{1, ..., t-1\}$, and the tokens are $\mathbf{R}_{-K:t} = \{R_K, ..., R_t\}$, $\mathbf{s}_{-K:t} = \{s_K, ..., s_t\}$ and $\mathbf{a}_{-K:t-1} = \{a_K, ..., a_{t-1}\}$. The DT policy is parametrized by the GPT architecture (Radford et al., 2018) with a causal self-attention mask, such that the action sequences are generated in an autoregressive manner. Namely, DT generates a deterministic action at timestep t by $\hat{a}_t = \pi_{\text{DT}}(\mathbf{R}_{-K:t}, \mathbf{s}_{-K:t}, \mathbf{a}_{-K:t-1})$. Then the policy can be trained by minimizing the loss between the predicted actions and the ground-truth actions in a sampled batch of data. Typically, DT uses the cross-entropy loss for discrete action spaces and the ℓ_2 loss for continuous action spaces.

4. Method

4.1. The Offline Safe RL Problem

In this section, we revisit the offline safe RL problem and investigate its taxonomy based on collected datasets' cost threshold and properties. Denote $\mathcal{T} = \{\tau_1, \tau_2, ...\}$ as a dataset of trajectories. For the sake of subsequent analysis, we make the assumption that the dataset is both *clean* and *reproducible*, meaning that any trajectory in the dataset can be reliably reproduced by a policy. This is an important precondition, as characterizing noisy datasets that contain outliers in highly stochastic environments is challenging and lies beyond the discussion scope of this paper.

To describe a dataset $\mathcal{T}$ with both reward and cost metrics, we introduce the Pareto Frontier (PF), Inverse Pareto Frontier (IPF), and the Reward Frontier (RF) functions that are inspired by the MOO domain. The PF of a dataset $\mathcal{T}$ is computed by the maximum reward of trajectories under cost threshold $\kappa \in [0, \infty)$:

$$\text{PF}(\kappa, \mathcal{T}) = \max_{\tau\in\mathcal{T}} R(\tau), \quad s.t. \quad C(\tau) \leq \kappa.$$

Similarly, the IPF of a dataset $\mathcal{T}$ is defined by the maximum reward beyond cost threshold $\kappa \in [0, \infty)$:

$$\text{IPF}(\kappa, \mathcal{T}) = \max_{\tau\in\mathcal{T}} R(\tau), \quad s.t. \quad C(\tau) \geq \kappa.$$

The RF is defined by the maximum reward with cost $\kappa \in \mathbb{C}$, where $\mathbb{C} := \{C(\tau) : \tau \in \mathcal{T}\}$ is the set of all the possible episodic cost in $\mathcal{T}$:

$$\text{RF}(\kappa, \mathcal{T}) = \max_{\tau\in\mathcal{T}} R(\tau), \quad s.t. \quad C(\tau) = \kappa.$$

Note that their constraints and domains of κ are different. All the functions characterize the shape of the dataset. RF is "local" since it represents the highest reward of a cost and is only defined on reachable cost values in dataset $\mathcal{T}$. On the other hand, PF and IPF are "global", since PF/IPF is the supremum of all the RF values w.r.t costs smaller/larger than a cost threshold. They are both defined on a continuous space of κ. It is also easy to observe that the Pareto frontier $\text{PF}(\kappa, \mathcal{T})$ is a non-deceasing function of κ, which suggests the trade-offs between safety and task performance: finding a policy with a small cost return usually needs to sacrifice the reward. Based on the definition of PF and IPF, we introduce ϵ-reducible to characterize the property of the dataset.

Definition 1 (ϵ-reducible)**.** An offline safe RL dataset $\mathcal{T}$ is ϵ-reducible w.r.t. threshold κ if: $\text{PF}(\kappa, \mathcal{T}) = \text{IPF}(\kappa, \mathcal{T}) + \epsilon$.

It is worth noticing that $\epsilon \in \mathbb{R}$ rather than $\mathbb{R}_{\geq 0}$. A positive ϵ means that there does not exist any trajectory $\kappa \in \mathcal{T}$ that can achieve a higher reward than $\text{PF}(\kappa, \mathcal{T})$ even if removing the safety constraint, so the optimal policy is more likely to be an interior point within the safety boundary. A negative ϵ indicates that the reward of most rewarding trajectories in $\mathcal{T}$ is upper bounded by $\text{PF}(\kappa, \mathcal{T}) - \epsilon$, and thus the agent has a high chance for violating safety constraint if the policy greedily maximizes the reward. In this case, the optimal policy will likely be on the safety constraint boundary.

Fig. 1 shows an example of the cost-reward return plots of two datasets $\mathcal{T}_1$ and $\mathcal{T}_2$. Note that although $\mathcal{T}_1$ and $\mathcal{T}_2$ are collected in the same environment, $(\kappa, \mathcal{T}_1)$ and $(\kappa, \mathcal{T}_2)$ denote two different offline safe RL problems. We observe that the ϵ-reducible property can characterize the task difficulty. For instance, problem $(\kappa, \mathcal{T}_2)$ is usually easier to solve than $(\kappa, \mathcal{T}_1)$, because $(\kappa, \mathcal{T}_2)$ could be *reduced* to an offline RL problem by simply maximizing the reward without considering the cost constraint. We have the following conjecture regarding the task difficulty:

Suppose problem $(\kappa, \mathcal{T})$ is ϵ-reducible, then the smaller ϵ, the more difficult to find the optimal solution.

We empirically validate the hypothesis by experiments over different ϵ-reducible problems, and interestingly, we find that **standard offline RL algorithms** can achieve safe performance with high-reward in large-ϵ-reducible problems without solving constrained optimizations. However, their performance deteriorates when the problem $(\kappa, \mathcal{T})$ possesses a smaller ϵ value. Detailed results and discussions can be found in Appendix A.

Remark 1 (Applicability). It is important to note that the comparison of ϵ-reducible values is only valid within a single task under the same CMDP. Comparisons across different tasks are not meaningful as a smaller ϵ value in one task does not necessarily imply that this dataset is more challenging than another dataset related to a different task with a larger ϵ.

Remark 2 (Limitations). The concept of ϵ-reducibility may fall short when characterizing the complexity of noisy datasets or datasets related to highly-stochastic tasks. This is primarily because noisy datasets may include outlier trajectories with improbable high rewards and low costs. Similarly, in a highly-stochastic environment, the initial state distribution can significantly influence the final reward and cost. As such, the dataset is also likely to contain high-reward, low-cost trajectories due to "lucky" initial conditions.

Remark 3 (Relation to Temptation). The concept of reducibility aligns with the temptation definition in safe RL literature (Liu et al., 2022b), as both describe the reward and cost trade-offs. However, they differ in their operational domains. Temptation focuses on the policy space and its expected returns, flagging a problem as tempting if it has high-reward but unsafe policies. In contrast, ϵ-reducibility evaluates the dataset in the trajectory space, labeling a dataset as small ϵ-reducible if it includes high-reward, high-cost trajectories. Hence, they offer complementary perspectives to understand the challenges of safe RL.

The proposed ϵ-reducible can serve as a measure of the offline safe RL problem difficulties: larger ϵ means the problem is more likely to be **reduced** to standard offline RL, though more rigorous analysis remains to be explored in future work. In this work, we are more interested in small ϵ-reducible problems, because these problems can hardly be solved by standard offline RL methods. The cost-reward return plots and their RF curves of the datasets in our experiments are also presented in Appendix B.2.3.

4.2. Offline Safe RL Beyond a Single Threshold

Most existing offline safe RL approaches train policies by solving a constrained optimization problem, where learnable dual variables are updated based on the estimation of constraint violation cost and a target threshold (Xu et al., 2022a; Lee et al., 2022; Polosky et al., 2022). The cost return estimation is of the form $C_\pi = \mathbb{E}_{\tau\sim\pi}[C(\tau)]$ and the

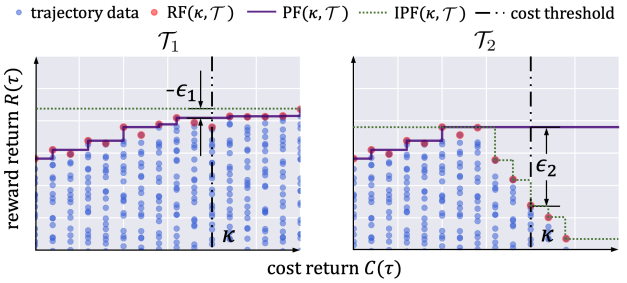


Figure 1. Cost-reward return plot for two collected datasets $\mathcal{T}_1, \mathcal{T}_2$. Each point represents trajectories with corresponding episodic cost and reward values in the dataset.

reward return is $R_\pi = \mathbb{E}_{\tau\sim\pi}[R(\tau)]$. The learning algorithm aims to train a policy π to maximize the reward return R_π while satisfying the constraint $C_\pi \leq \kappa$. The constrained optimization training scheme works well in online safe RL settings (Stooke et al., 2020), however, two main challenges arise for the offline setting. We detail them as follows.

First, **the trained policy tends to be either unsafe or overly conservative** due to biased and inaccurate estimation in the offline setting. This is because the trajectories τ from the dataset $\mathcal{T}$ are sampled from various unknown behavior policies rather than the optimized policy. In RL, biased estimation of the reward return R_π may not affect the results because the maximization operation over R_π is invariant to the bias and scale, However, in safe RL, a biased estimation of cost C_π could cause significantly wrong dual variables, because its absolute value is compared against a fixed threshold κ. A small negative bias can lead to unsafe behaviors, and a positive bias can induce a conservative policy. This problem is challenging for off-policy safe RL (Liu et al., 2022a) and is more difficult to address in the offline setting, as we will show empirically in the experiment section 5.1.

Second, **the trained policy cannot be easily adapted to different constraint thresholds without re-training**. The cost threshold needs to be pre-selected and kept fixed throughout training because otherwise, the dual variables for penalizing constraint violations could be unstable when solving the constrained optimization and thus diverge the learning process. Therefore, adapting the policy to different constraint conditions requires re-training with new thresholds.

The second challenge corresponds to the problem of learning a safe policy from the offline dataset beyond a single constraint threshold. Formally speaking, given a dataset $\mathcal{T}$, the trained agent $\pi(a|s,\kappa)$ is expected to be generalized to arbitrary cost thresholds $\forall\kappa \in [C_{\text{min}}, C_{\text{max}}]$, where $C_{\text{min}}, C_{\text{max}}$ are the minimum and maximum of the cost return of the trajectories in the dataset. The best reward return of $\pi(a|s,\kappa)$ is lower-bounded by the Pareto frontier value of the dataset with threshold κ: $\text{PF}(\kappa, \mathcal{T})$.

The limitations of the constrained optimization-based train-

ing paradigm motivate us to think about other learning schemes. We find that sequential modeling techniques, such as Decision Transformer (DT) (Chen et al., 2021b), have great potential to achieve zero-shot adaptation to different constraint thresholds while maintaining safety and near Pareto optimal task performance, As introduced in Sec. 3.2, the DT policy predicts target-return-conditioned actions, which provides the flexibility to adjust the agent behaviors after training. However, we observe that simply adding a target cost return token to the input sequence of DT can hardly ensure safety in practice. Therefore, we propose two simple yet effective improvements over DT to train a safe and adaptive policy, which yields the Constrained Decision Transformer (CDT) algorithm.

4.3. Constrained Decision Transformer

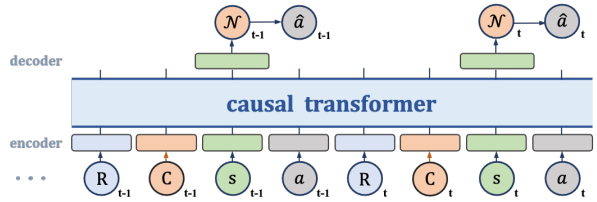


Figure 2. Constrained decision transformer architecture.

CDT is built upon the DT architecture with two main different components: 1) stochastic policy with entropy regularization, and 2) Pareto frontier-oriented data augmentation by target return relabeling. We found that the two techniques play crucial roles in improving safety and robustness against conflicting target returns. We detail them as follows.

Stochastic CDT policy with entropy regularization. The model architecture of CDT is shown in Fig. 2, where the differences between CDT and DT are highlighted in orange. Recall that the input tokens for DT are $\mathbf{R}_{-K:t} = \{R_K, ..., R_t\}$, $\mathbf{s}_{-K:t}$ and $\mathbf{a}_{-K:t-1}$, where $K \in \{1, ..., t-1\}$ is the context length and $R_t = \sum_{t'=t}^{T} r_{t'}$ is the reward return of step t. CDT augments the input sequences with an additional element that represents the target cost threshold $\mathbf{C}_{-K:t} = \{C_K, ..., C_t\}$, where $C_t = \sum_{t'=t}^{T} c_{t'}$ is the cost return starting from timestep t. The intuition is to generate actions conditioned on both the reward return and the cost return. For example, at timestep t, setting $C_t = 10$ and $R_t = 80$ means that we expected the agent to obtain 80 rewards with a maximum allowed 10 costs. Different from DT, which predicts deterministic action sequences, CDT adopts the stochastic Gaussian policy representation, drawing inspiration from the Online Decision Transformer architecture (Zheng et al., 2022). Denote $\mathbf{o}_t := \{\mathbf{R}_{-K:t}, \mathbf{C}_{-K:t}, \mathbf{s}_{-K:t}, \mathbf{a}_{-K:t-1}\}$ as the input tokens and θ as the CDT policy parameters, we have:

$$\pi_\theta(\cdot|\mathbf{o}_t) = \mathcal{N}\left(\mu_\theta(\mathbf{o}_t), \Sigma_\theta(\mathbf{o}_t)\right).$$

The use of a stochastic representation confers multiple benefits. Firstly, a deterministic policy may be more prone to producing out-of-distribution actions due to systematic bias, which can lead to large compounding errors in an offline environment and potentially result in constraint violations (Xu et al., 2022a). Further details on this property can be found in Appendix D.1. Secondly, the stochastic policy representation allows the policy to explore a more diverse range of actions and enhance performance through interaction with the environment. This aligns with the pretraining and finetuning learning paradigm in the literature and shows great promise for real-world applications (Zheng et al., 2022). Finally, we can easily apply a regularizer to prevent the policy from overfitting and improve the robustness against approximation errors (Ziebart, 2010; Eysenbach & Levine, 2021). We adopt the Shannon entropy regularizer $H[\pi_\theta(\cdot|\mathbf{o})]$ for CDT, which is widely used in RL (Haarnoja et al., 2018). The optimization objective is to minimize the negative log-likelihood loss while maximizing the entropy with weight $\lambda \in [0, \infty)$:

$$\min_\theta \quad \mathbb{E}_{\mathbf{o}\sim\mathcal{T}}\left[-\log \pi_\theta(\mathbf{a}|\mathbf{o}) - \lambda H[\pi_\theta(\cdot|\mathbf{o})]\right] \tag{2}$$

Since CDT adopts the target returns-conditioned policy structure, the agent behavior is sensitive to the choices of target reward and cost. In offline RL, one can set a large enough target reward for the agent to maximize the reward. However, as shown in Fig. 1, the feasible choices of valid target cost and reward return pairs are restricted under the RF points, which brings a major challenge for CDT: how can we resolve the potential conflict between desired returns and ensure the target cost is of higher priority than the target reward? For instance, if the initial reward return is set to be slightly higher than the RF value of the initial target cost threshold, then it is hard to determine whether the policy will achieve the desired reward but violate the constraint or satisfy the cost threshold but with a lower reward.

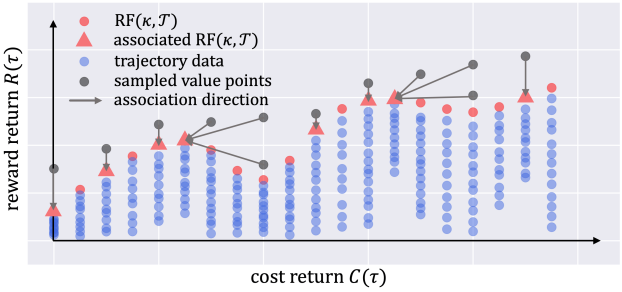


Figure 3. Data augmentation.

Data augmentation by return relabeling. We propose an effective augmentation technique to address the above issue by utilizing the Pareto frontier and reward frontier properties of the trajectory-level dataset $\mathcal{T}$. Suppose (ρ, κ) is an infeasible target return pair, i.e., $\rho > \text{RF}(\kappa, \mathcal{T})$. We associate the conflict target with the safe trajectory that of the maximum

Algorithm 1 Data Augmentation via Relabeling

Input: dataset $\mathcal{T}$, samples N, reward sample max r_{max}
Output: augmented trajectory dataset $\mathcal{T}$

$c_{min} \leftarrow \min_{\tau\sim\mathcal{T}} C(\tau)$, $c_{max} \leftarrow \max_{\tau\sim\mathcal{T}} C(\tau)$
for $i = 1, ..., N$ **do**
  ▷ *sample a cost return*
  $\kappa_i \sim \text{Uniform}(c_{min}, c_{max})$
  ▷ *sample a reward return above the RF value*
  $\rho_i \sim \text{Uniform}(\text{RF}(\kappa_i, \mathcal{T}), r_{max})$
  ▷ *find the closest and safe Pareto trajectory*
  $\tau_i^* \leftarrow \arg\max_{\tau\sim\mathcal{T}} R(\tau), s.t. \quad C(\tau) \leq \kappa_i$
  ▷ *relabel the reward and cost return*
  $\hat{\tau}_i \leftarrow \{\mathbf{R}_i^* + \rho_i - R(\tau_i^*), \mathbf{C}_i^* + \kappa_i - C(\tau_i^*), \mathbf{s}_i^*, \mathbf{a}_i^*\}$
  ▷ *append the trajectory to the dataset*
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\tau}_i\}$
end for

reward return: $\tau^* = \arg\max_{\tau\sim\mathcal{T}} R(\tau), s.t. \quad C(\tau) \leq \kappa$. We can observe that $\tau^* = \{\mathbf{R}^*, \mathbf{C}^*, \mathbf{s}^*, \mathbf{a}^*\}$ is the maximum-reward Pareto optimal trajectory with cost less than κ. Then we append the new trajectory data $\hat{\tau} = \{\mathbf{R}^* + \rho - R(\tau^*), \mathbf{C}^* + \kappa - C(\tau^*), \mathbf{s}^*, \mathbf{a}^*\}$ to the dataset: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\tau}\}$. Note that the operators over $\mathbf{R}^*$ and $\mathbf{C}^*$ are element-wise. The intuition is to relabel the associated Pareto trajectory's reward and cost returns, such that the agent can learn to imitate the behavior of the most rewarding and safe trajectory τ^* when the desired return (ρ, κ) is infeasible. Fig. 3 shows an example of the procedure, where the arrows associate Pareto-optimal trajectories with corresponding augmented return pairs. The detailed augmentation procedures are presented in Algorithm 1.

It is worth noting that real-world datasets can be noisy, occasionally including anomalous "lucky" trajectories that record high reward and low-cost returns despite originating from subpar behavioral policies. These outliers, while rare, can disrupt the data augmentation procedure, thereby negatively affecting CDT's performance. To address this issue, our implementation utilizes two specific techniques. The first involves associating each augmented return pair (r, c) with a trajectory sampled in proximity to the nearest and safe Pareto frontier data point, based on a specified distance metric. The second technique employs a density filter to remove such outliers exhibiting abnormal reward and cost returns during the creation of the training dataset, thus mitigating the outlier concern. More details regarding these two techniques and empirical validations are available in Appendix D.2.

Training and evaluation. CDT generally follows the training and evaluation schemes of return-conditioned sequential modeling methods (Chen et al., 2021b; Zheng et al., 2022). The training procedure is similar to training a Transformer in supervised learning: sample a batch of sequences $\mathbf{o}, \mathbf{a}$ from the augmented dataset $\mathcal{T}$, compute the loss in Eq. (2) to optimize the Transformer policy model π_θ via gradient descent. The evaluation procedure for a trained CDT model is presented in Algorithm 2. Note that it differs from standard RL, where the policy directly predicts the action based on the state. As shown in Fig. 2, the input for the return-conditioned policy is a tuple of four sequences: target reward and cost returns for each step, past states, and actions. Therefore, the output is also a sequence of actions, but we only execute the last one in the environment. The target returns will be updated correspondingly upon receiving new reward and cost signals from the environment.

Algorithm 2 Returns Conditioned Evaluation for CDT

Input: trained Transformer policy π_θ, episode length T, context length K, target reward and cost R_1, C_1, env

Get the initial state: $s_1 \leftarrow$ env.reset()
Initialize input sequence $\mathbf{o} = [\{R_1, C_1, s_1\}]$
for $t = 1, ..., T$ **do**
  Get predicted action $a_t \sim \pi(\cdot|\mathbf{o}[-K:])[-1]$
  Execute the action: $s_{t+1}, r_t, c_t \leftarrow$ env.step(a_t)
  ▷ *compute target returns for the next step*
  $R_{t+1} = R_t - r_t, C_{t+1} = C_t - c_t,$
  Append the new token $o_t = \{R_{t+1}, C_{t+1}, s_{t+1}, a_t\}$ to the sequence $\mathbf{o}$
end for

5. Experiment

In this section, we aim to evaluate the proposed approach and empirically answer the following questions: 1) can CDT learn a safe policy from a small ϵ reducible offline dataset? 2) what is the importance of each component in CDT? 3) can CDT achieve zero-shot adaption to different constraint thresholds? 4) is CDT robust to conflict reward returns? To address these questions, we adopt the following tasks to evaluate CDT and baseline approaches.

Tasks. We use several robot locomotion continuous control tasks that are commonly used in previous works (Achiam et al., 2017; Chow et al., 2019; Zhang et al., 2020b). The simulation environments are from a public benchmark (Gronauer, 2022). We consider two environments (`Run` and `Circle`) and train multiple different robots (`Car`, `Drone`, and `Ant`). In the Run environment, the agents are rewarded for running fast between two boundaries and are given constraint violation cost if they run across the boundaries or exceed an agent-specific velocity threshold. In the Circle environment, the agents are rewarded for running in a circle but are constrained within a safe region that is smaller than the radius of the target circle. We name the task as `robot-environment` such as `Ant-Run`.

Offline datasets. The dataset format follows the D4RL benchmark (Fu et al., 2020), where we add another cost

Methods	Ant-Run		Car-Circle		Car-Run		Drone-Circle		Drone-Run		Average	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
CDT(ours)	**89.76**	**0.83**	**89.53**	**0.85**	**99.0**	**0.45**	**73.01**	**0.88**	**63.64**	**0.58**	**82.99**	**0.72**
BC-Safe	**80.56**	**0.64**	**78.21**	**0.74**	**97.21**	**0.01**	**66.49**	**0.56**	**32.73**	**0.0**	**71.04**	**0.39**
DT-Cost	91.69	1.32	89.08	2.14	100.67	11.83	78.09	2.38	72.3	4.43	86.37	4.42
BCQ-Lag	92.7	1.04	89.76	3.91	96.14	3.21	71.14	3.37	47.61	1.81	79.47	2.67
BEAR-Lag	91.19	1.66	15.48	2.24	**99.09**	**0.09**	72.36	1.99	**19.07**	**0.0**	59.44	1.2
CPQ	**78.52**	**0.14**	**75.99**	**0.0**	**97.72**	**0.11**	55.14	9.67	72.24	4.28	75.92	2.84
COptiDICE	**45.55**	**0.6**	52.17	6.38	**92.86**	**0.89**	36.44	5.54	26.56	1.38	50.72	2.96
CDT(w/o augment)	93.62	1.53	89.8	1.38	99.58	1.89	74.9	1.35	66.93	1.53	84.97	1.54
CDT(w/o entropy)	**87.47**	**0.64**	89.94	1.07	**98.92**	**0.44**	**73.76**	**0.97**	**62.29**	**0.6**	**82.48**	**0.74**
CDT(deterministic)	94.21	1.42	89.53	1.43	101.52	17.53	**76.4**	**1.0**	68.44	1.36	86.02	4.55

Table 1. Evaluation results of the normalized reward and cost. The cost threshold is 1. ↑: the higher reward, the better. ↓: the lower cost (up to the threshold 1), the better. Each value is averaged over 20 episodes and 3 seeds. **Bold**: Safe agents whose normalized cost is smaller than 1. Gray: Unsafe agents. **Blue**: Safe agent with the highest reward.

entry to record binary constraint violation signals. We collect offline datasets using the CPPO safe RL approach with well-tuned hyperparameters (Stooke et al., 2020). We gradually increase its cost threshold such that the trajectories can cover a diverse range of cost returns and reward returns. All the training data for CPPO is stored as the raw dataset, which may contain many repeated trajectories. We further down-sample the data by applying a grid filter over the cost-reward return space (Fig. 1) and trim redundant trajectories to avoid the impact of unevenly distributed data (Gulcehre et al., 2020; Gong et al., 2022; Singh et al.). Namely, we divide the cost-reward space into multiple 2D grids, randomly select a fixed number of trajectories within each grid and discard the remaining ones. The cost-return plots of different datasets used in this work are presented in Appendix A.

Metrics. We adopt the normalized reward return and the normalized cost return as the comparison metrics, which are consistent with the offline RL literature (Fu et al., 2020). Denote $r_{\max}(\mathcal{T})$ and $r_{\min}(\mathcal{T})$ as the maximum reward return and the minimum reward return in dataset $\mathcal{T}$. The normalized reward is computed by:

$$R_{\text{normalized}} = \frac{R_\pi - r_{\min}(\mathcal{T})}{r_{\max}(\mathcal{T}) - r_{\min}(\mathcal{T})} \times 100,$$

where R_π denotes the evaluated reward return of policy π. The normalized cost is defined a bit differently from the reward, which is computed by the ratio between the evaluated cost return C_π and the target threshold κ:

$$C_{\text{normalized}} = \frac{C_\pi}{\kappa + \epsilon},$$

where ϵ is a small positive number to ensure numerical stability if the threshold $\kappa = 0$. Note that the cost return is always non-negative in our setting, and we use $\kappa = 10$ by default. Without otherwise statements, we will abbreviate "normalized cost return" as "cost" and "normalized reward return" as "reward" for simplicity.

We can observe that a policy is unsafe if the cost is greater than 1. We deliberately scale the reward around the range $[0, 100]$ and the cost around 1 to distinguish them in the result table better. The comparison criteria follow the safe RL setting (Ray et al., 2019): a safe policy is better than an unsafe one. For two unsafe policies, the one with a lower cost is better. For two safe policies, the one with a higher reward is better.

Baselines with a fixed cost threshold. We use two recent offline safe RL approaches: **CPQ** (Xu et al., 2022a) and **COptiDICE** (Lee et al., 2022) as two strong baselines. We adopt two Lagrangian-based baselines: BCQ-Lagrangian (**BCQ-Lag**) and BEAR-Lagrangian (**BEAR-Lag**), which is built upon BCQ (Fujimoto et al., 2019) and BEAR (Kumar et al., 2019), respectively. The Lagrangian approach follows the expert policy CPPO implementation, which uses adaptive PID-based Lagrangian multipliers to penalize constraint violations (Stooke et al., 2020). We use the vanilla Decision Transformer (Chen et al., 2021b) with an additional cost return token as another baseline **DT-Cost**, aiming to compare the effectiveness of the proposed CDT training techniques. We also include a Behavior Cloning baseline (**BC-Safe**) that only uses safe trajectories to train the policy. This serves to measure whether each method actually performs effective RL, or simply copies the data.

We also conducted comprehensive studies on the Behavior Cloning method with different datasets, including **BC-all, BC-risky, BC-frontier,** and **BC-boundary**. Due to space constraints, we defer the visualization of datasets and experiment results on these BC-variants to Appendix D.3.

Hyperparameters. We use a fixed set of hyperparameters for CDT across all tasks. Most common parameters, such as the gradient steps, are also the same for CDT and baselines. The detailed hyperparameters are in Appendix C.2.

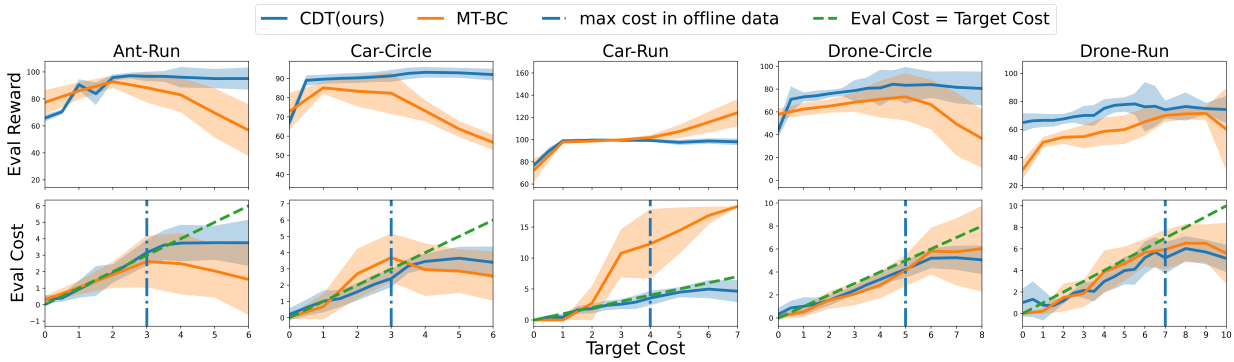


Figure 4. Results of zero-shot adaption to different cost returns. Each column is a task. The x-axis is the target cost return. The first row shows the evaluated reward, and the second row shows the evaluated cost under different target costs. All plots are averaged among 3 random seeds and 20 trajectories for each seed. The solid line is the mean value, and the light shade represents the area within one standard deviation. The vertical line is the maximum normalized cost return in the offline dataset.

5.1. Can CDT learn safe policies from offline datasets?

The evaluation results for different trained policies are presented in Table 1. We can find that only our method (CDT) and BC-Safe successfully learn safe policies for all tasks, while CDT consistently achieves higher reward returns than BC-safe. It makes sense since BC-safe only uses safe data to train and thus fails to explore high-rewarding trajectories. The comparison between BC-safe indicates that CDT performs effective RL rather than copying data.

The results of DT-cost show that simply adding a cost return token to the original DT structure can not train a constraint satisfaction policy, though it successfully learns to maximize the reward return. Note that in the Car-Run task, DT-cost even outperforms the best trajectory's reward in the dataset; however, the cost is also extremely high. The comparison indicates that the proposed training techniques in CDT are crucial in learning a safe policy.

The Lagrangian-based baselines BCQ-Lag and BEAR-Lag fail to behave safely on most tasks, which suggests that directly applying widely-used safe RL techniques to the offline setting can hardly work well. Surprisingly, the CPQ and COptiDICE methods that are designed for offline safe RL also fail to satisfy the constraints in difficult Drone-related tasks. Particularly, the CPQ algorithm either performs over-conservatively with near zero cost or too aggressively with large costs. As we discussed in Sec. 4.2, one reason for the poor performance is the difficult-to-estimate cost value of the optimized policy. The trajectories in the dataset are collected from various behavior policies and may cause biased cost value estimation, which then causes too large or too small dual variables. Accurately fitting a cost critic is still a challenging problem for off-policy safe RL (Liu et al., 2022a), let alone the offline setting. The poor performance of baselines shows the difficulties of the experimental tasks; however, the proposed CDT can learn safe and high-rewarding policies in those tasks very well.

Ablation study. To study the influence of data augmentation, stochastic policy, and entropy regularization, we conduct experiments by removing each component from CDT. The results are shown in the lower part of Table 1. It is clear to see both augmentation and stochastic representation are necessary and important components since we can observe significant safety performance degradation if removing either one of them. Besides, entropy regularization can result in a slight improvement regarding the overall performance.

5.2. Can CDT achieve zero-shot adaption to different constraint thresholds?

As introduced in Sec. 4.2, one significant advantage of CDT over baselines is its capability of zero-shot adaptation to different cost thresholds. It is obvious that the baselines introduced previously lack this capability because they need a fixed pre-defined threshold to solve a constrained optimization problem. Adapting them to new constraint conditions requires re-training. To this end, we add another baseline Multi-task Behavior Cloning (**MT-BC**) to compare the zero-shot adaptation performance with CDT (Xu et al., 2022c). We view each cost return threshold as a task and concatenate the task information (episodic cost return) to the corresponding task's states and train the agent via BC. Namely, the BC policy predicts an action that is conditioned on both state and cost threshold: $a_t = \pi_{\text{MT-BC}}(s_t, \kappa)$.

We fix the target reward and vary the target cost for evaluation rollouts to obtain the results in Fig. 4. We can see that MT-BC has certain adaptation capabilities for the in-distribution target costs. However, when the cost limit exceeds the maximum cost in the datasets, the actual cost increases greatly (`Car-Run`), or the reward decreases significantly in other tasks. On the contrary, the actual cost of CDT is strongly correlated with the target cost return and under the dashed threshold line, which shows great **interpolation** capability. The curves saturate at certain target

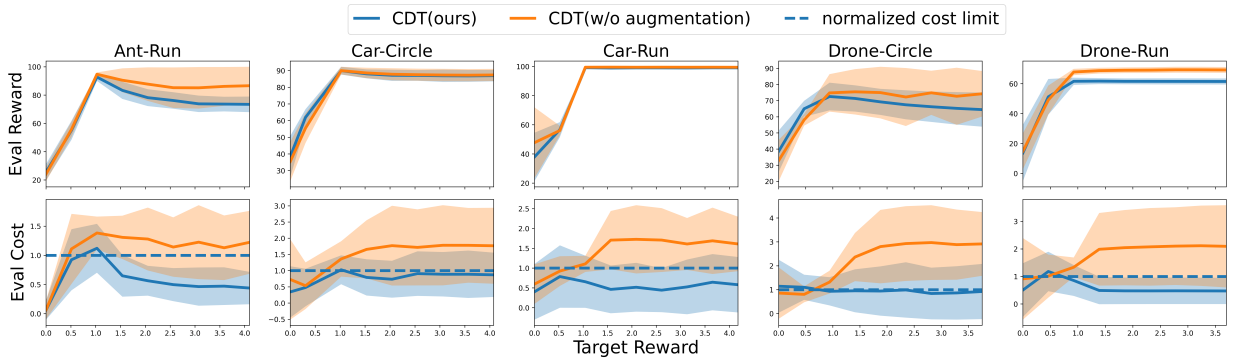


Figure 5. Ablation study of the data augmentation technique. The x-axis is the target reward return. The first row shows the evaluated reward, and the second row shows the evaluated cost under different target costs. The dashed line is the target cost threshold.

costs that are beyond the maximum one in the training data, which shows that the agent can maintain safety even when performing **extrapolation** over unseen target cost returns. Furthermore, the actual reward return of CDT does not drop compared to MT-BC.

5.3. Is CDT robust to conflict reward returns?

As mentioned in section 4.3, there exists infeasible target cost and reward pairs that could influence safety performance, which motivates us to propose the data augmentation technique. To test its effectiveness, we fix the target cost return and vary the target reward to evaluate CDT and CDT without data augmentation. The results are shown in Fig. 5. We can observe that the actual reward increases and then saturates as the target reward increases. However, the actual cost keeps increasing and finally exceeds the cost limit if removing data augmentation, while CDT can maintain safety even if the target return is large. The results show that data augmentation is a necessary component in CDT to handle conflicting target returns.

5.4. Can ϵ-reducible characterize the task difficulty?

To corroborate our hypothesis on the reducibility attribute of offline datasets in Sec. 4.1, we conduct experiments with both full (small ϵ) and reduced (large ϵ) datasets. The reduced dataset was constructed by removing trajectories whose costs exceeded the threshold and with high rewards. This process ensures that the most rewarding trajectories are safe, i.e., $\text{PF}(\kappa, \mathcal{T}) > \text{IPF}(\kappa, \mathcal{T})$. Then we train standard offline RL algorithms, such as DT (Chen et al., 2021b), BCQ (Fujimoto et al., 2019), and BEAR (Kumar et al., 2019) on these datasets. Due to the page limit, we present the results in Appendix A.

The results show that these algorithms perform poorly in safety performance on the full dataset, which is as expected since maximizing reward is the sole objective. However, when training them on reduced datasets with large ϵ values, we can observe a significant improvement in terms of safety performance. This observation aligns with our conjecture: larger ϵ-reducible problems are relatively easier to solve for the same task, as using standard offline RL algorithms can achieve good performance.

6. Conclusion

We study the offline safe RL problem from the multi-objective optimization perspective and propose an empirically verified ϵ-reducible concept to characterize the task difficulty. We further propose the CDT method that is capable of learning a safe and high-reward policy in challenging offline safe RL tasks. More importantly, CDT can achieve zero-shot adaptation to different constraint thresholds without re-training and is robust to conflicting target returns, while prior works fail. These advantages make CDT preferable for real-world applications with safety constraints.

There are also several limitations of CDT: 1) it more computing resources due to the Transformer architecture; 2) it lacks rigorous theoretical guarantees for safety; 3) it requires instant reward and cost feedback during the policy deployment and rollout; 4) improper target reward return and cost return can still deteriorate the performance, and 5) achieving zero-constraint violations is still challenging. Therefore, studying a more lightweight method to address the above issues could be promising for future work. Nevertheless, we hope our findings can inspire more research in this direction to study the safety and generalization capability in offline learning.

Acknowledgements

We express our profound gratitude to the National Science Foundation for their generous support under the CAREER CNS-2047454 grant. Funding to attend this conference was also provided by the CMU GSA/Provost Conference Funding. Equally, we extend our sincerest appreciation to the reviewers for their invaluable insights and constructive suggestions, which significantly contributed to the enhancement of our manuscript.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Altman, E. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical methods of operations research*, 48(3):387–417, 1998.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A. P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5, 2021.

Chen, B., Liu, Z., Zhu, J., Xu, M., Ding, W., and Zhao, D. Context-aware safe reinforcement learning for non-stationary environments. *arXiv preprint arXiv:2101.00531*, 2021a.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021b.

Chen, Y., Dong, J., and Wang, Z. A primal-dual approach to constrained markov decision processes. *arXiv preprint arXiv:2101.10895*, 2021c.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., and Ghavamzadeh, M. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

Eysenbach, B. and Levine, S. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

Flet-Berliac, Y. and Basu, D. Saac: Safe reinforcement learning as an adversarial game of actor-critics. *arXiv preprint arXiv:2204.09424*, 2022.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=CAjxVodl_v.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Gong, C., Yang, Z., Bai, Y., He, J., Shi, J., Sinha, A., Xu, B., Hou, X., Fan, G., and Lo, D. Mind your data! hiding backdoors in offline reinforcement learning datasets. *arXiv preprint arXiv:2210.04688*, 2022.

Gronauer, S. Bullet-safety-gym: Aframework for constrained reinforcement learning. 2022.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Yang, Y., and Knoll, A. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T., Gómez, S., Zolna, K., Agarwal, R., Merel, J. S., Mankowitz, D. J., Paduraru, C., et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. 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.

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

Ji, J., Zhou, J., Zhang, B., Dai, J., Pan, X., Sun, R., Huang, W., Geng, Y., Liu, M., and Yang, Y. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *arXiv preprint arXiv:2305.09304*, 2023.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

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

Lee, J., Paduraru, C., Mankowitz, D. J., Heess, N., Precup, D., Kim, K.-E., and Guez, A. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. *arXiv preprint arXiv:2204.08957*, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liu, Z., Zhou, H., Chen, B., Zhong, S., Hebert, M., and Zhao, D. Constrained model-based reinforcement learning with robust cross-entropy method. *arXiv preprint arXiv:2010.07968*, 2020.

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, S., Li, B., and Zhao, D. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pp. 13644–13668. PMLR, 2022a.

Liu, Z., Guo, Z., Cen, Z., Zhang, H., Tan, J., Li, B., and Zhao, D. On the robustness of safe reinforcement learning under observational perturbations. *arXiv preprint arXiv:2205.14691*, 2022b.

Lu, Y., Fu, J., Tucker, G., Pan, X., Bronstein, E., Roelofs, B., Sapp, B., White, B., Faust, A., Whiteson, S., et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. *arXiv preprint arXiv:2212.11419*, 2022.

Luo, Y. and Ma, T. Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations. *Advances in Neural Information Processing Systems*, 34, 2021.

Nachum, O., Chow, Y., Dai, B., and Li, L. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in Neural Information Processing Systems*, 32, 2019a.

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

Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Polosky, N., Da Silva, B. C., Fiterau, M., and Jagannath, J. Constrained offline policy optimization. In *International Conference on Machine Learning*, pp. 17801–17810. PMLR, 2022.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*, 2022.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019.

Saunders, W., Sastry, G., Stuhlmueller, A., and Evans, O. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv:1707.05173*, 2017.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Singh, A., Kumar, A., Chebotar, Y., Levine, S., et al. Offline reinforcement learning from heteroskedastic data via support constraints. In *Deep Reinforcement Learning Workshop NeurIPS 2022*.

Sinha, S., Mandlekar, A., and Garg, A. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pp. 907–917. PMLR, 2022.

Sootla, A., Cowen-Rivers, A. I., Jafferjee, T., Wang, Z., Mguni, D. H., Wang, J., and Ammar, H. Sauté rl: Almost surely safe reinforcement learning using state augmentation. In *International Conference on Machine Learning*, pp. 20423–20443. PMLR, 2022.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, 2020.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Wagener, N., Boots, B., and Cheng, C.-A. Safe reinforcement learning using advantage-based intervention. *arXiv preprint arXiv:2106.09110*, 2021.

Wang, Z., Novikov, A., Zolna, K., Merel, J. S., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N., Gulcehre, C., Heess, N., et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33: 7768–7778, 2020.

Xie, T., Jiang, N., Wang, H., Xiong, C., and Bai, Y. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

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

Xu, M., Liu, Z., Huang, P., Ding, W., Cen, Z., Li, B., and Zhao, D. Trustworthy reinforcement learning against intrinsic vulnerabilities: Robustness, safety, and generalizability. *arXiv preprint arXiv:2209.08025*, 2022b.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pp. 24631–24645. PMLR, 2022c.

Yang, Q., Simão, T. D., Tindemans, S. H., and Spaan, M. T. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. In *AAAI*, pp. 10639–10646, 2021.

Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. Gendice: Generalized offline estimation of stationary values. *arXiv preprint arXiv:2002.09072*, 2020a.

Zhang, Y., Vuong, Q., and Ross, K. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 2020b.

Zhao, W., He, T., and Liu, C. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

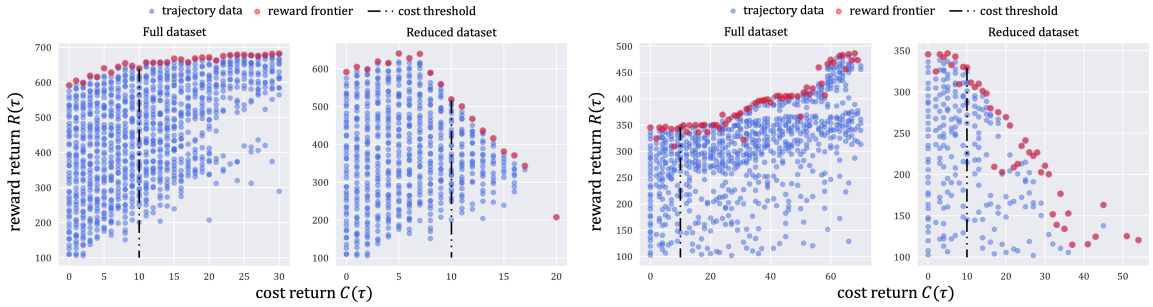
Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. *arXiv preprint arXiv:2202.05607*, 2022.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A. Results and Discussions on the ϵ-reducible Datasets

To validate our hypothesis of the reducible property of the offline safe RL problem, we first construct reduced dataset (large ϵ) from full dataset (small ϵ) as shown in Figure 6. We remove the trajectories if their costs exceed the cost threshold and their rewards are higher than the trajectories whose costs are equal to the cost threshold to ensure that the trajectories with the highest reward satisfy the safety constraints, namely, $\text{PF}(\kappa, \mathcal{T}) > \text{IPF}(\kappa, \mathcal{T})$. Then we train the standard offline RL methods such as DT (Chen et al., 2021b), BCQ (Fujimoto et al., 2019) and BEAR (Kumar et al., 2019) on both the full and reduced dataset. The evaluation results are shown in Figure 2.

As expected, on the small-ϵ-reducible full data, these standard offline RL methods have poor safety performance since maximizing reward is the only optimization objective. They learn to mimic the trajectories with higher rewards but violate the safety constraint in the dataset. However, when they are trained on the large-ϵ-reducible dataset, we can observe a significant improvement in terms of safety performance. The safety constraint will be naturally satisfied because the high-reward trajectories in the reduced dataset have smaller cost values than the threshold. In summary, we found that standard offline RL algorithms can solve the large-ϵ-reducible problems well in most cases, which serves as strong evidence for our conjecture: the larger ϵ, the easier to solve the problem.



(a) Ant-Run task datasets, $\hat{\epsilon}|_{\kappa=10} = -0.040$ within full dataset, $\hat{\epsilon}|_{\kappa=10} = 0.189$ within reduced dataset.

(b) Drone-Run task datasets, $\hat{\epsilon}|_{\kappa=10} = -0.281$ within full dataset, $\hat{\epsilon}|_{\kappa=10} = 0.102$ within reduced dataset.

Figure 6. The cost-reward return plot of reduced datasets. The normalized ϵ-reducible value for each dataset is normalized by the maximum return value in the dataset.

Methods	Full Dataset (small ϵ)						Reduced Dataset (large ϵ)					
	Ant-Run ($\hat{\epsilon} = -0.040$)		Drone-Run ($\hat{\epsilon} = -0.281$)		Average		Ant-Run ($\hat{\epsilon} = 0.189$)		Drone-Run ($\hat{\epsilon} = 0.102$)		Average	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓
DT	96.13	2.47	49.09	4.69	72.61	3.58	**81.26**	**0.91**	63.88	1.03	**72.57**	**0.97**
BCQ	97.99	5.39	60.1	3.01	79.04	4.2	82.59	1.19	**42.34**	**0.51**	**62.46**	**0.85**
BEAR	91.05	1.69	**42.13**	**0.48**	66.59	1.08	**84.2**	**0.55**	**33.29**	**0.0**	**58.74**	**0.28**

Table 2. Evaluation results of the normalized reward and cost. The cost threshold is 1. ↑ / ↓: the higher/lower, the better. Each value is averaged over 20 episodes and 3 seeds. **Bold**: Safe agents whose normalized cost is smaller than 1. Gray: Unsafe agents. The normalized ϵ-reducible value for each dataset, which is normalized by the maximum return value in the dataset, is also labeled in the table.

One implicit assumption for ϵ-reducible property is that the learning capability of an offline RL learner is limited, and the datasets are of good quality that can cover high-reward spaces, i.e., the agent can hardly achieve any trajectory τ with a higher reward $r > \text{PF}(\mathcal{T}, \kappa)$ under safety constraint κ given the collected dataset $\mathcal{T}$. With this limited learning ability assumption, given a positive-reducible dataset, the agent will not achieve a reward that is higher than $\text{PF}(\mathcal{T}, \kappa)$ even if we remove safety constraints during offline training.

B. Implementation Details

B.1. CDT Training Procedure Pseudo Code

Algorithm 3 CDT Training Procedure

Input: Transformer model π_θ, dataset $\mathcal{T}$, learning rate α, context length K, batch size B, entropy weight λ, gradient steps M, maximum episode length T, augment samples N, reward sample max r_{max}
Output: trained Transformer model π_θ

Augment dataset: $\mathcal{T} \leftarrow Augment(\mathcal{T}, N, r_{max})$
for update step $= 1, ..., M$ **do**
  ▷ *sample a batch of sequences of length K*
  $\mathcal{B} = \{\mathbf{a}_{i,t}, \mathbf{o}_{i,t}\}_{i=1}^{B} \sim \mathcal{T}, t \sim \text{SampleInt}(1, T)$
  ▷ *compute the NLL loss and entropy loss*
  $\ell_{\text{nll}} = -\frac{1}{|\mathcal{B}|} \sum_{\mathbf{a},\mathbf{o} \in \mathcal{B}} \log \pi_\theta(\mathbf{a}|\mathbf{o})$
  $\ell_{\text{ent}} = -\frac{1}{|\mathcal{B}|} \sum_{\mathbf{o} \in \mathcal{B}} H[\pi_\theta(\cdot|\mathbf{o})]$
  ▷ *update the policy parameter*
  $\ell_{\text{cdt}} = \ell_{\text{nll}} + \lambda \ell_{\text{ent}}$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \ell_{\text{cdt}}$
end for

B.2. Dataset Collection

B.2.1. Algorithm

We collect offline datasets using the CPPO safe RL approach (Stooke et al., 2020), which is an improved version of the PPO-Lagrangian method by using a PID controller to update the dual variable (Ray et al., 2019). Suppose the reward and cost value functions V_r, V_c are parameterized by θ_r and θ_c networks respectively. We use GAE (Schulman et al., 2015) to update the value functions:

$$\begin{aligned} \theta_r &\leftarrow \arg\min_{\theta_r} \mathbb{E}_{s_\tau \sim \mathcal{D}} \left[\left(V_r(s_\tau) - \sum_{t=0}^{T-\tau} (\lambda^{gae}\gamma)^t r(s_t, a_t) \right)^2 \right] \\ \theta_c &\leftarrow \arg\min_{\theta_c} \mathbb{E}_{s_\tau \sim \mathcal{D}} \left[\left(V_c(s_\tau) - \sum_{t=0}^{T-\tau} (\lambda^{gae}\gamma)^t c(s_t, a_t) \right)^2 \right] \end{aligned} \tag{3}$$

where γ is the discounting factor, and λ^{gae} is the GAE constant. The objective of clipped PPO has the form (Schulman et al., 2017):

$$\ell_{ppo} = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_r^{\pi_{\theta_k}}(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\right) A_r^{\pi_{\theta_k}}(s,a)\right) \tag{4}$$

We use PID Lagrangian (Stooke et al., 2020) that addresses the oscillation and overshoot problem in Lagrangian methods. The loss of the PPO-Lagrangian has the form:

$$\ell_{ppol} = \frac{1}{1+\lambda}(\ell_{ppo} - \lambda A_c^{\pi_{\theta_k}}(s,a)) \tag{5}$$

The Lagrangian multiplier λ is computed by applying feedback control to V_c^π and is determined by positive constants k_P, k_I, and k_D. Instead of using a fixed cost threshold ϵ, we apply a time-varying cost threshold so that we can collect data within a wide range of reward and cost values. The procedure of `CPPO` is summarized in Alg. 4.

B.2.2. Hyperparameters of the expert CPPO policy

To collect datasets with a large range of cost and reward return values, we fine-tune the hyperparameters in `CPPO`. The key hyperparameters for our dataset collection are listed in Tab. 3.

Algorithm 4 CPPO

Input: Cost interval $[C_{\text{min}}, C_{\text{max}}]$, starting epoch n_1, ending epoch n_2, epoch n, PID parameter $\{k_P, k_I, k_D\}$, learning rate η_ϕ.
Output: Policy parameter ϕ, dataset Γ.

Initialization: target cost error $e_0 \leftarrow 0$, replay buffer $\mathcal{D} \leftarrow \{\}$, dataset $\Gamma \leftarrow \{\}$.
for $i = 1, ..., n$ **do**
  Compute the threshold $\epsilon \leftarrow \min\{C_{\text{max}}, \max\{C_{\text{min}}, \frac{C_{\text{max}} - C_{\text{min}}}{n_2 - n_1}(i - n_1) + \epsilon_1\}\}$
  Sample N trajectories $\{s_0, a_0, \ldots, s_T\}_{n=1,\ldots,N}$ with policy π_i and store transition data to replay buffer $\mathcal{D}$.
  Compute the expectation of cost return J_c.
  Calculate the error between the real cost and the cost threshold: $e_i \leftarrow J_c - \epsilon$.
  Update dual variable $\lambda \leftarrow k_P\, e_i + k_I \max\{0, \sum_{j=1}^{i} e_j\} + k_D \max\{0, e_i - e_{i-1}\}$.
  Update value functions based on Eq. (3).
  Update policy: $\phi \leftarrow \phi + \eta_\phi \nabla_\phi (\mathcal{L}_r - \lambda \mathcal{L}_c)$.
  Save the dataset $\Gamma \leftarrow \Gamma \cup \mathcal{D}$ and empty the buffer $\mathcal{D} \leftarrow \{\}$.
end for

parameters	Car-Run	Ant-Run	Car-Circle	Drone-Circle	Drone-Run
ϵ_{clip}	0.2	0.2	0.2	0.15	0.15
λ^{gae}	0.97	0.97	0.97	0.95	0.95
γ	0.99	0.99	0.99	0.98	0.98
$[\epsilon_1, \epsilon_2]$	$[5, 80]$	$[5, 80]$	$[5, 80]$	$[10, 80]$	$[5, 80]$
$[n_1, n_2]$	$[50, 400]$	$[45, 200]$	$[50, 200]$	$[20, 550]$	$[10, 150]$
n	400	210	210	570	160

Table 3. The hyperparameters for `CPPO` algorithm for data collection

B.2.3. DATASET VISUALIZATION

The dataset cost-reward return plots for the training tasks `Ant-Run`, `Car-Circle`, `Car-Run`, `Drone-Circle`, and `Drone-Run` are shown in Fig. 7. Due to the limited sampling number, the reward frontier value is not monotonically increasing with respect to the cost. However, we can observe the trend that high-cost values are with high maximum reward values in most cases. This is consistent with our intuition and our motivation for loosening safety constraints: large reward values are traded off by the high risk of violating safety constraints. However, in some cases, such as the `Car-Run` task, this trend is not conspicuous. It is because `Car-Run` is an easy task – with easy robot dynamics and a simple environment, that the safety constraint can hardly block the `CPPO` agent from reaching a higher reward.

C. Experiment Setting and Hyperparameters

C.1. Experiment Description

We use the Bullet safety gym (Gronauer, 2022) environments for this set of experiments. In the Run tasks, agents are rewarded for running fast between two safety boundaries and are given costs for violation constraints if they run across the boundaries or exceed an agent-specific velocity threshold. The reward and cost functions are defined as:

$$r(\boldsymbol{s_t}) = ||\boldsymbol{x_{t-1}} - \boldsymbol{g}||_2 - ||\boldsymbol{x}_t - \boldsymbol{g}||_2 + r_{robot}(s_t)$$
$$c(\boldsymbol{s_t}) = \mathbf{1}(|y| > y_{lim}) + \mathbf{1}(||\boldsymbol{v_t}||_2 > v_{lim})$$

where v_{lim} is the speed limit, y_{lim} specifies the safety region, $\boldsymbol{v_t} = [v_x, v_y]$ is the velocity of the agent at timestamp t, $\boldsymbol{g} = [g_x, g_y]$ is the position of a fictitious target, $\boldsymbol{x_t} = [x_t, y_t]$ is the position of the agent at timestamp t, and $r_{robot}(\boldsymbol{s_t})$ is the specific reward for different robot. For example, an ant robot will gain reward if its feet do not collide with each other. In the Circle tasks, the agents are rewarded for running in a circle in a clockwise direction but are constrained to stay within a

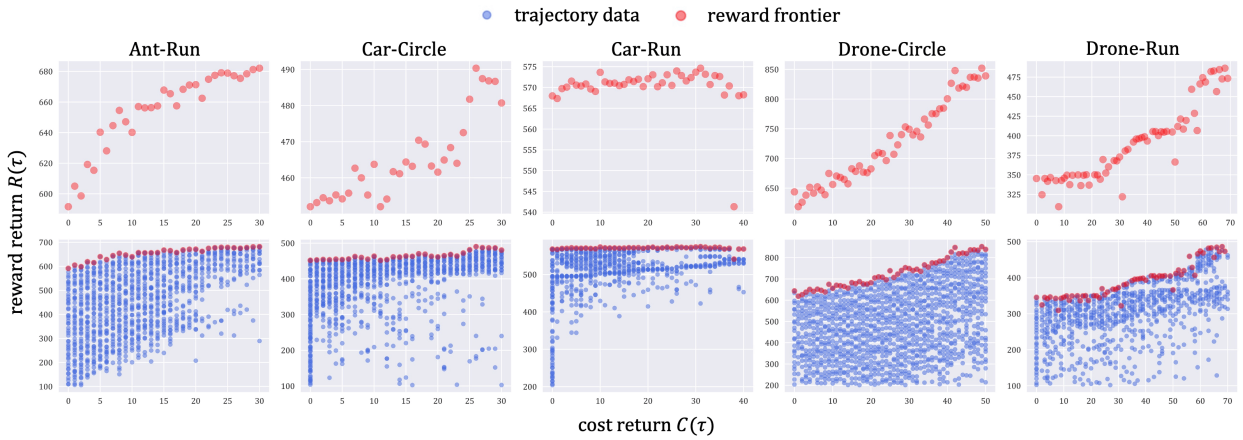


Figure 7. Cost-reward return plot. The reward frontiers of each sampling cost are marked in red. Each column represents a task. The first row shows the reward frontier points in the dataset, and the second row shows the whole dataset. Each point represents collected trajectories (not necessarily to be unique) with corresponding episodic cost and reward value. The cost values are discrete because all these tasks adopt the 0-1 cost. The normalized ϵ-reducible values at threshold $\kappa = 10$ for these datasets are listed as Ant-Run: -0.040, Car-Circle: -0.056, Car-Run: -0.005, Drone-Circle: -0.208, Drone-Run: -0.281. The normalized ϵ-reducible value for each dataset is normalized by the maximum return value in the dataset.

safe region that is smaller than the radius of the target circle. The reward and cost functions are defined as:

$$r(\boldsymbol{s_t}) = \frac{-y_t v_x + x_t v_y}{1 + |||\boldsymbol{x_t}||_2 - r|} + r_{robot}(\boldsymbol{s_t})$$
$$c(\boldsymbol{s_t}) = \mathbf{1}(|x| > x_{lim})$$

where r is the radius of the circle, and x_{lim} specifies the range of the safety region.

C.2. Hyperparameters

For baselines, we use Gaussian policies with mean vectors given as the outputs of neural networks, and with variances that are separate learnable parameters. The policy networks and Q networks for all experiments have two hidden layers with ReLU activation functions. The K_P, K_I and K_D are the PID parameters (Stooke et al., 2020) that control the Lagrangian multiplier for the Lagrangian-based algorithms. We use the same 10^5 gradient steps and rollout length which is the maximum episode length for CDT and baselines for fair comparison. The cost threshold for baselines is 10 across all the tasks. The hyperparameters that are not mentioned are in their default value for baselines. The complete hyperparameters used in the experiments are shown in Table 4.

Parameter	All tasks
Number of layers	3
Number of attention heads	8
Embedding dimension	128
Batch size	2048
Context length K	10
Learning rate	0.0001
Droupout	0.1
Adam betas	(0.9, 0.999)
Grad norm clip	0.25

Parameter	Ant-Run	Car-Circle	Car-Run	Drone-Circle	Drone-Run
Actor hidden size	[256, 256] BCQ-Lag, BEAR-Lag [300, 300] CPQ				
VAE hidden size	[750, 750] BEAR-Lag [400, 400] CPQ				
Rollout length	200	300	200	300	100
$[K_P, K_I, K_D]$	[0.1, 0.003, 0.001] BCQ-Lag, BEAR-Lag				
Batch size	512				
Actor learning rate	0.0001	0.001	0.0001	0.0001	0.001
Critic learning rate	0.001	0.001	0.001	0.001	0.001

Table 4. Hyperparameters for CDT (left) and baselines (right).

D. More Results and Discussions

D.1. Why do stochastic policies have better empirical safety performance?

From the experiments, we can observe that using a stochastic policy head is much better than using a deterministic policy. We conjecture that this is because stochastic policies can help alleviate potential constraint violations caused by out-of-distribution (OOD) actions and extrapolation errors in offline safe RL. In other words, if the policy learns to incorrectly estimate an unseen action as the most rewarding and safe, but it is actually unsafe, a deterministic policy will always take that action with probability 1. However, a stochastic policy can choose other safe actions by sampling, which improves safety.

We provide a numerical example as follows. Consider a problem with discrete action spaces: $\mathcal{A} = \{a_1, a_2, a_3\}$. We focus on the decision at state s. The rewards and costs are

$$r(s, a_1) = 1, r(s, a_2) = 2, r(s, a_3) = 3,$$

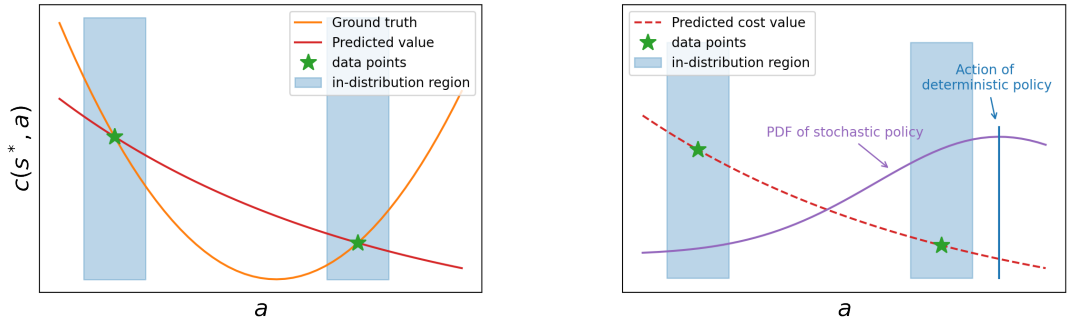
$$c(s, a_1) = 0, c(s, a_2) = 0, c(s, a_3) = 10,$$

and the state-action pairs $(s, a_1), (s, a_2)$ are in the dataset while (s, a_3) is not. Due to the increasing trend of reward and the safe actions based on the observed data at state s, the policy may wrongly estimates that the OOD action a_3 is most rewarding and also safe.

Therefore, a deterministic policy will execute a_3 with probability 1, and the expected cost is 10.

In contrast, a stochastic policy may output a policy distribution that is proportional to the reward: $\pi(a_1|s) = 1/6, \pi(a_2|s) = 2/6, \pi(a_3|s) = 3/6$. Thus, the expected cost should be $\pi(a_3|s)c(s, a_3) = 3/6 \times 10 = 5$, which is significantly smaller than the deterministic policy's cost.

We also provide a figure illustration for continuous action space. As shown in fig. 8, if we consider the cost function (which is also applicable to other safety-related functions depending on different methods) at a fixed state $s = s^*$, it may not be well-estimated with limited offline data. In this case, to reduce the cost to keep safety, the deterministic policy will output an out-of-distribution action (blue line), while a stochastic policy can still maintain a part of probability weight on in-distribution and safe regions (orange line).



(a) When offline data is limited, we may not estimate the cost function correctly for out-of-distribution regions.

(b) The action from corresponding deterministic policy is always out-of-distribution while the stochastic policy can still sample in-distribution actions.

Figure 8. The predicted cost function and corresponding stochastic and deterministic policies. Deterministic policy usually suffers from out-of-distribution issue more severely, which leads to worse safety performance especially when the extrapolation error of cost estimation is large.

Therefore, employing a stochastic policy can improve safety performance when facing out-of-distribution data and extrapolation errors, making it a crucial technique for CDT in offline safe RL. More in-depth theoretical analysis will be a promising direction for future research.

D.2. Ablation studies for noisy datasets

The real-world dataset could be noisy and contains trajectories that accidentally record high reward returns and small cost returns despite following a poor behavioral policy. Such lucky trajectories would be rare (outliers) and should be omitted in learning. However, when such a lucky trajectory exists in the dataset, it may affect the data augmentation procedure in CDT and thus negatively affect its performance. To investigate this issue, our implementation adopts two techniques. They are summarized as follows.

1. The first technique is to associate each augmented return pair (r, c) with a trajectory sampled in the neighborhood of the nearest and safe Pareto frontier data point, and the sampling distribution is based on a distance metric $W((r_p, c_p), (r', c')) := ||(r_p, c_p) - (r', c')||_2 + \beta$, where $\beta \in R_+$ is a constant, (r_p, c_p) is the return of the Pareto frontier, and $(r', c') \in U((r_p, c_p))$ is the return of the data point in the neighbor of (r_p, c_p). That is to say, we may associate the return pair (r, c) to a range of data points around the Pareto frontier (r_p, c_p). The probability of data selection is inversely proportional to the distance, i.e., $p((r, c) \leftarrow (r', c')) \propto 1/W((r_p, c_p), (r', c'))$. This sampling-based association can not only increase the diversity of augmented trajectories, but also mitigate the outlier issue.

2. The second technique uses a density filter to remove such outliers with abnormal reward and cost returns when creating the training dataset. In particular, we implemented a grid filter that first segments the reward and cost return space into evenly distributed grids and then counts the number of trajectories within each grid. As such, we can easily filter the outlier trajectories of small densities.

To show the effectiveness of the above techniques, we create such a dataset that contains different portions ($\alpha\% = 0.1\%, 0.4\%, 1\%, 1.5\%, 2\%$) of outlier trajectories based on the Drone-Run dataset. We consider the task with stochastic reward and cost function, i.e., high-cost trajectories have the probability of $\alpha\%$ to be labeled as a "lucky" trajectory with high reward and low cost. Specifically, we select $\alpha\%$ high-cost trajectories and modify their cost return to be less than the cost threshold, as shown in the red dots in Fig. 9.

The middle figure visualizes the sampling-based association technique. Note that the outliers do not only affect the association, but also influence the sampled return and cost pairs. We can see that some of the augmented returns are wrongly associated with the outliers, but the remaining ones are paired correctly.

The right figure demonstrates the result of applying a density filter. We can see that the outliers are removed since their densities on the reward and cost return space are small. Therefore, the augmented returns are associated correctly. However, we can observe that some normal trajectories on the lower right regions are also filtered.

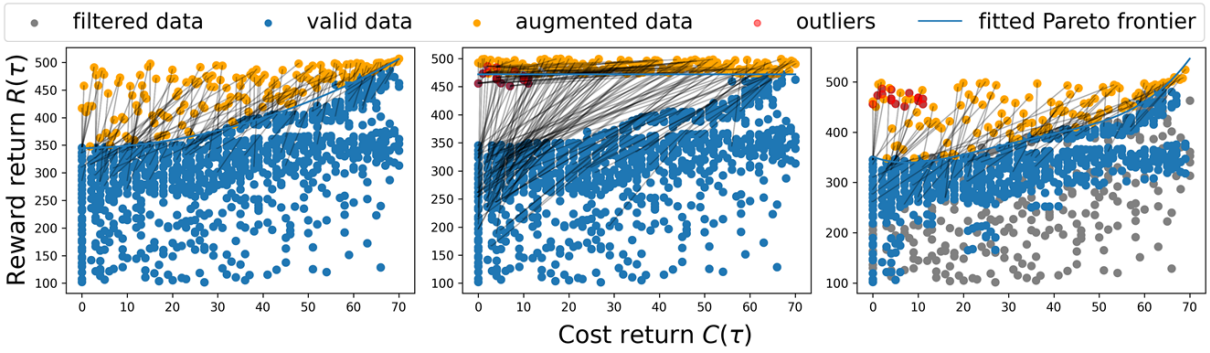


Figure 9. Cost-reward return illustrations of the Drone-Run dataset. Each point denotes the trajectories with corresponding cost-return values. Left: the original dataset w/o outlier trajectories. Middle: sampling-based association with 2% outlier trajectories. Right: density filter with 2% outlier trajectories. The black lines show the association.

We perform CDT to train the agents based on the above two techniques. The evaluation results are listed in Table 5. Note that the natural performance of CDT on the clean dataset is $r = 63.64, c = 0.58$. We can observe that as the outlier percentage increases, CDT's safety performance does be affected: the cost may also increase. However, in most cases, CDT can still learn a safe policy, and the reward doesn't drop too much. The results indicate that the proposed two techniques can mitigate the negative effect induced by outlier trajectories.

Methods	$\alpha = 0.1$		$\alpha = 0.4$		$\alpha = 1.0$		$\alpha = 1.5$		$\alpha = 2.0$		Average	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
CDT(ours) sampling-based association	**63.5**	**0.78**	**63.11**	**0.43**	**68.13**	**0.31**	**62.63**	**0.68**	**58.8**	**0.86**	**63.23**	**0.61**
CDT(ours) density filter	**62.37**	**0.7**	**62.67**	**0.72**	**60.85**	**0.99**	**67.55**	**0.75**	64.31	1.32	**63.39**	**0.76**

Table 5. Evaluation results of the normalized reward and cost w.r.t different portions of outlier trajectories. The cost threshold is 1. ↑: the higher reward, the better. ↓: the lower cost (up to the threshold 1), the better. Each value is averaged over 20 episodes and 3 seeds. **Bold**: Safe agents whose normalized cost is smaller than 1. Gray: Unsafe agents.

Apart from the above existing techniques used in our work, we also provide additional ideas to address this issue in different cases, which are inspired by the out-of-distribution (OOD) detection domain. We detail them as follows.

- For the datasets collected in environments with highly stochastic transition dynamics, we can filter outlier trajectories based on the probability of transition dynamics. We first train an empirical transition dynamics density estimator $\hat{p}(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ by randomly sample transitions (s,a,s') from the dataset, and then compute the transition probability of each trajectory τ in the dataset: $p(\tau) = \prod_{t \geq 0} \hat{p}(s_{t+1}|s_t, a_t)$. We can then discard $\alpha\%$ trajectories with the lowest probabilities, where α is the percent of outliers since they are rare in the datasets.
- For the datasets that might contain lucky trajectories with high reward and low cost, we can reject the paired Pareto trajectory based on the counts of associated augmentation samples. More specifically, after sampling reward and cost return pairs and finishing association, we can count the number of associated return pairs for each Pareto trajectory. If the count is of a significantly high portion among the total samples, we could discard this Pareto trajectory and continue the process. For example, if we have 100 augmented return pairs in total, and one Pareto data has 80 associated pairs, then we could regard the data as an outlier and remove it from the dataset.

Nevertheless, for extremely noisy datasets, the augmentation technique proposed in CDT may still fail. Investigating how to pre-process the datasets and detect these abnormal trajectories would be interesting for future work.

D.3. Comparison with more behavior-cloning variants

To determine whether our approach truly employs efficient RL or merely duplicates the offline data, we adopt multiple variants of BC that utilize different portions of the offline data for training agents. As illustrated in Figure 10, **BC-all** utilizes the entire set of data, **BC-Safe** solely relies on safe trajectories, **BC-risky** exclusively employs high-cost trajectories, **BC-frontier** utilizes the trajectories that are close to the Pareto frontier, while **BC-boundary** focuses on the trajectories that are near the cost threshold.

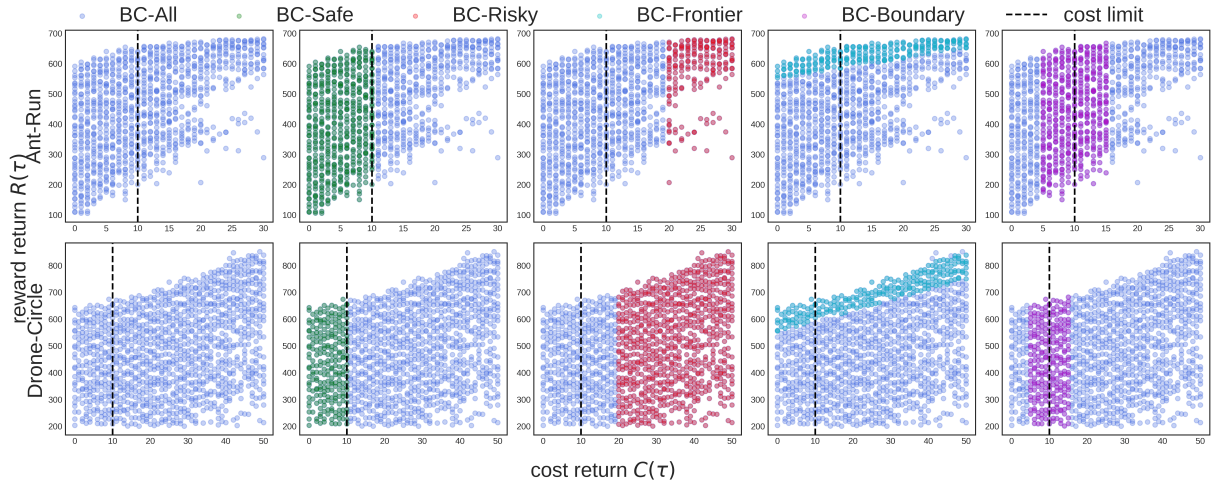


Figure 10. Cost-reward return illustrations of the datasets used to train different BC agents (Ant-Run and Drone-Circle dataset). Each point denotes the trajectories with corresponding cost-return values.

From Table 6, we can find that only our approach (CDT) and BC-Safe can successfully learn safe policies for all tasks, with CDT consistently achieving higher rewards than BC-Safe. The under-performance of BC-Risky is not surprising, given that it exclusively utilizes unsafe data. As expected, BC-risky fails to learn safe policies since it only uses unsafe data. The poor performance of BC-Boundary and BC-Frontier indicates that relying on a more aggressive expert who can generate high-risk, high-reward trajectories is insufficient for exploring safe boundaries. BC-All can be deemed an average of the other BC variants and demonstrates the ability to learn safe policies for some tasks. The comparisons between BC variants and CDT indicates that CDT performs effective RL rather than copying data.

Methods	Ant-Run		Car-Circle		Car-Run		Drone-Circle		Drone-Run		Average	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
CDT(ours)	**89.76**	**0.83**	**89.53**	**0.85**	**99.0**	**0.45**	**73.01**	**0.88**	**63.64**	**0.58**	**82.99**	**0.72**
BC-Safe	**80.56**	**0.64**	**78.21**	**0.74**	**97.21**	**0.01**	**66.49**	**0.56**	**32.73**	**0.0**	**71.04**	**0.39**
BC-all	90.86	1.45	**82.81**	**0.62**	**97.48**	**0.01**	73.29	2.81	**49.58**	**0.28**	78.8	1.03
BC-risky	95.31	3.14	84.1	2.52	96.73	2.71	79.68	3.89	66.74	4.17	84.51	3.29
BC-boundary	86.01	1.04	**83.57**	**0.86**	**97.76**	**0.0**	**67.07**	**0.24**	62.93	3.57	79.47	1.14
BC-frontier	95.08	1.55	89.76	1.51	98.74	1.58	85.62	3.11	75.36	3.44	88.91	2.24

Table 6. Evaluation results of the normalized reward and cost. The cost threshold is 1. ↑: the higher reward, the better. ↓: the lower cost (up to the threshold 1), the better. **Bold**: Safe agents whose normalized cost is smaller than 1. Gray: Unsafe agents. **Blue**: Safe agent with the highest reward.