Trajectory-wise Iterative Reinforcement Learning Framework for Auto-bidding

Haoming Li*
Shanghai Jiao Tong University
Shanghai, China
wakkkka@sjtu.edu.cn

Yusen Huo
Alibaba Group
Beijing, China
huoyusen.huoyusen@alibaba-inc.com

Shuai Dou
Alibaba Group
Beijing, China
doushuai.ds@alibaba-inc.com

Zhenzhe Zheng†
Shanghai Jiao Tong University
Shanghai, China
zhengzhenzhe@sjtu.edu.cn

Zhilin Zhang
Alibaba Group
Beijing, China
zhangzhilin.pt@alibaba-inc.com

Chuan Yu
Alibaba Group
Beijing, China
yuchuan.yc@alibaba-inc.com

Jian Xu
Alibaba Group
Beijing, China
xiyu.xj@alibaba-inc.com

Fan Wu
Shanghai Jiao Tong University
Shanghai, China
fwu@cs.sjtu.edu.cn

ABSTRACT

In online advertising, advertisers participate in ad auctions to acquire ad opportunities, often by utilizing auto-bidding tools provided by demand-side platforms (DSPs). The current auto-bidding algorithms typically employ reinforcement learning (RL). However, due to safety concerns, most RL-based auto-bidding policies are trained in simulation, leading to a performance degradation when deployed in online environments. To narrow this gap, we can deploy multiple auto-bidding agents in parallel to collect a large interaction dataset. Offline RL algorithms can then be utilized to train a new policy. The trained policy can subsequently be deployed for further data collection, resulting in an iterative training framework, which we refer to as iterative offline RL. In this work, we identify the performance bottleneck of this iterative offline RL framework, which originates from the ineffective exploration and exploitation caused by the inherent conservatism of offline RL algorithms. To overcome this bottleneck, we propose Trajectory-wise Exploration and Exploitation (TEE), which introduces a novel data collecting and data utilization method for iterative offline RL from a trajectory perspective. Furthermore, to ensure the safety of online exploration while preserving the dataset quality for TEE, we propose Safe Exploration by Adaptive Action Selection (SEAS). Both offline experiments and real-world experiments on Alibaba display advertising platform demonstrate the effectiveness of our proposed method.

*Work done during internship at Alibaba Group.
†Zhenzhe Zheng is the corresponding author.

CCS CONCEPTS

• **Information systems** → **Computational advertising**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Real-time bidding, Reinforcement Learning, Offline Reinforcement Learning

1 INTRODUCTION

Online advertising is becoming one of the major sources of profit for Internet companies [7]. Due to the complex online advertising environments, auto-bidding tools provided by demand-side platforms (DSPs) are commonly used to bid on behalf of advertisers to optimize their advertising performance. Bidding for arriving ad impressions can be viewed as a sequential decision-making problem, and thus state-of-the-art auto-bidding algorithms leverage reinforcement learning (RL) to optimize bidding policies [4, 12, 31].

However, due to safety concerns, current RL-based auto-bidding policies are trained in simulated environments. Policies trained with simulation are shown suboptimal when deployed in the real-world system [21]. Therefore, it is desirable to optimize the bidding policy by directly interacting with the online environments. Classical (online) RL algorithms iteratively switch between data collection phase and policy update phase, and typically require enormous samples (i.e., transition tuples) to achieve convergence. However, collecting data samples can be extremely time-consuming, *e.g.*, in most RL formulations for auto-bidding [12, 31], an RL episode corresponds to 24 hours, and thus training a policy may take a long

time. Additionally, frequent updates of online policies may cause unstable performance and potentially risky bidding behaviours.

To address these issues, DSPs usually leverage a large number of auto-bidding agents as parallel workers to allow for the efficient collection of large amount of interaction data, and train an auto-bidding policy on the dataset with offline RL [10, 18] — a data-driven RL paradigm that aim to extract policies from large-scale pre-collected datasets. The updated policy could again be deployed for further data collection, which results in an iterative framework for data collection and policy update. We refer to the above training paradigm as iterative offline RL, as depicted in Figure 1. Iterative offline RL presents a promising solution for online policy training in industrial scenarios, and similar ideas have also been mentioned in several academic works [20, 21, 35].

To facilitate effective policy improvement for each iteration in iterative offline RL, it is crucial for the collected datasets to encompass sufficient information regarding various states and actions. This requires the exploration policy to incorporate a degree of randomness, typically achieved by introducing noise perturbation to actions [19–21]. Nonetheless, employing a random exploration policy can adversely affect the performance of the trained policy. This is because the introduced randomness undermines the exploration policy's performance, and offline RL algorithms heavily relies on the exploration policy due to the principle of conservatism (or pessimism) [9, 10, 15, 17, 27], which compels the newly updated policy to be close to the exploration policy. We demonstrate this phenomenon in Section 4, highlighting the challenges of effective exploration and exploitation for iterative offline RL.

In this work, we tackle the aforementioned challenges by adopting a trajectory perspective for both the exploration process (data collection) and the exploitation process (offline RL training). For efficient exploration, we construct exploration policies by introducing noise into the policy's parameter space instead of the traditional action space. This choice is motivated by our key observation that the injection of parameter space noise (PSN) yields an exploration dataset with a more dispersed trajectory return distribution (please refer to Section 5.1 for details). This observation indicates that the dataset contains a considerable number of high-return trajectories, which are valuable for offline training. For effective exploitation, we propose robust trajectory weighting to fully exploit high-return trajectories in the collected dataset. Specifically, instead of uniformly sampling the dataset during training, we assign high probability weights to high-return trajectories, thereby enhancing the impact of high-quality behaviours on the training process and overcoming the conservatism problem. However, the instability of online environments leads to highly random trajectory returns that often fail to reliably reflect the trajectory qualities. To address this issue, we design a new trajectory quality indicator by approximating the expectation of the stochastic rewards, enhancing the robustness of the trajectory weighting method. We leverage PSN for online exploration and utilize robust trajectory weighting to compute sampling probabilities before offline RL training, boosting the effectiveness of exploration and exploitation in iterative offline RL.

Apart from effectiveness, safety constraints must also be taken into consideration during online exploration in real-world advertising systems. Random exploration can lead to risky bidding behaviours, negatively impacting the performance of advertisers. The
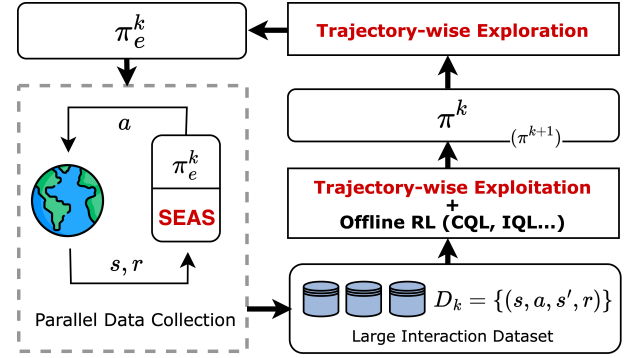

Figure 1: Iterative offline RL with Trajectory-wise Exploration and Exploitation (TEE) and Safe Exploration by Adaptive Action Selection (SEAS). Components proposed in this work are highlighted in red.

safety of an exploration policy is captured by a performance lower bound, which ensures that the performance drop brought by exploration is acceptable. Ensuring safe exploration often requires imposing constraints on the original exploration policy. However, existing safety-guaranteeing methods [21] often lack awareness of action qualities. While they prevent dangerous behaviours, they also restrict some high-quality actions. This may hinder the emergence of high-return trajectories during exploration, and subsequently affect the performance of the training process. To preserve data quality while ensuring safety, we propose SEAS, which dynamically determines the safe exploration action at each time step based on the cumulative rewards up to that step, as well as the predicted future return. By making adaptive decisions, SEAS preserves the quality of the collected dataset to the fullest extent, and achieves theoretically guaranteed safety at the same time.

The main contributions of this work are summarized as follows:

- We identify and demonstrate that the performance bottleneck of the current iterative offline RL paradigm for auto-bidding algorithms mainly lies in ineffective exploration and exploitation caused by the conservatism principle of offline RL algorithms.
- We propose TEE, a solution for effective exploration and exploitation in iterative offline RL for auto-bidding. TEE comprises two components: PSN for trajectory-wise exploration, and a novel Robust Trajectory Weighting algorithm for trajectory-wise exploitation.
- For safe exploration in auto-bidding, we design SEAS, which adaptively decides the safe exploration action for each time step based on the cumulative rewards till that step. SEAS can guarantee provable safety and sacrifice minimal performance in policy learning when working together with TEE.
- Extensive experiments in both simulated environments and Alibaba display advertising platform demonstrate the effectiveness of our solution in terms of trained policy's performance, as well as the safety of the training process.

2 RELATED WORK

Reinforcement learning for auto-bidding. Bid optimization in online advertising is a sequential decision procedure, and can be solved via reinforcement learning techniques. Cai et. al. [4] first formulated the auto-bidding problem as an MDP. Wu et. al. [31] and He et. al. [12] leveraged reinforcement learning to optimize bidding policies under various constraints. All of the above works train their RL bidding agent in a simulated environment. Mou et. al. [21] recognized the sim2real problem in auto-bidding, and designed an iterative offline RL framework to train the bidding policy online.

Offline RL. Offline RL [10, 18, 26] (or batch RL) refers to the problem of policy optimization utilizing only previously collected data, without additional online interaction. Due to the distribution shift [10] problem that arises in offline RL, most algorithms conduct conservative policy learning, which compels the learning policy to stay close to the dataset. Various algorithms achieve this by directly regularizing the actor [9, 10, 16, 24], learning conservative value functions [17], or constraining the number of policy improvement steps [3, 15]. However, the conservatism principle causes the performance of trained policy to highly depend on the dataset quality.

Dataset coverage in offline RL. Dataset coverage is a core factor that limits the performance of offline RL algorithms. Schweighofer et. al. [28] conducted extensive experiments to demonstrate that dataset coverage is essential for offline RL algorithms to learn a good policy. Prior theoretical works on offline RL [5, 27, 34] also relied on datasets with sufficient state-action space coverage, which is often characterized by concentrability coefficients, to produce a strong performance guarantee.

Exploration in RL. Exploration is a crucial aspect of RL as it allows the agent to gather information about the environment. Traditional exploration strategies induce novel behaviours by random perturbations of actions, such as ϵ-greedy [29] and entropy regularization [30]. To generate meaningful behavioural patterns for hard exploration tasks, several other approaches such as intrinsic reward-based exploration [6, 23], count-based exploration [2, 22] and PSN [8, 25] have been proposed. However, most of the existing solutions have been proposed for the online RL paradigm, which is substantially different from iterative offline RL where data is collected for training offline RL algorithms.

3 PRELIMINARIES

In this work, we consider auto-bidding with budget constraint, a sequential decision problem, where an advertiser submits bids for incoming ad impressions, aiming to maximize the total value within a fixed budget. For this problem, previous works [36, 37] showed that under the second price auction [?], the optimal bid b^* on an impression is given by $b^* = v/\lambda$, where v represents the impression value (*e.g.* click-through rate) and λ is a scaling factor. However, determining the optimal value of λ in real time is intractable due to its dependence on values and costs of all impressions in the stream. Thus, we formulate the problem of adjusting bidding parameter λ as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$. In our formulation, an episode corresponds to a one-day ad campaign duration, which is divided into T time steps. At each step t, the advertiser observes state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$.

The state s_t is a feature vector describing the advertising status of a campaign, which may contain time, remaining budget, budget consumption speed, and etc. The action a_t is the bidding parameter λ in time step t. The reward $r(s_t, a_t)$ is the total value of impressions won between time step t and $t + 1$, and $p(s_{t+1}|s_t, a_t)$ denotes the transition probability of states. Both r and p are determined by the advertising environment. The discount factor $\gamma \in [0, 1]$ accounts for the future rewards' diminishing impact. A (deterministic) policy $\pi \in \Pi : \mathcal{S} \rightarrow \mathcal{A}$ is a function defining the agent's bidding behaviour. When a policy interacts with the advertising environment over an episode, a trajectory $\tau = \{(s_t, a_t, s_{t+1}, r_t)\}_{t=0}^{T}$ is generated[1], where the initial state s_0 is drawn from a probability distribution ρ.

The (discounted) return of trajectory τ is $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$. The objective of RL is to maximize the expected return:

$$\underset{\pi \in \Pi}{\arg\max} \, J(\pi) := \mathbb{E}_\tau[R(\tau)].$$

In RL, the state value function of a policy π is defined as

$$V^\pi(s) := \mathbb{E}_\pi[\sum_{u=t}^{T} \gamma^u r_u | s_t = s], \quad \forall s \in \mathcal{S}.$$

Similarly, the action value function is

$$Q^\pi(s, a) := \mathbb{E}_\pi[\sum_{u=t}^{T} \gamma^u r_u | s_t = s, a_t = a], \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Iterative offline RL follows a cyclical pattern of data collection and offline policy update, repeatedly for a total of K iterations. Within each iteration $k \in [K]$, an exploration policy π_e^k is constructed based on π^k. Then π_e^k is deployed in the advertising environment to collect interaction dataset $D^k = \{\tau_i\}_{i=1}^N$ containing N trajectories. An offline RL algorithm is then used to learn a policy from D^k, producing the updated policy π^{k+1} for the subsequent iteration.

Safety of Bidding Policies. The performance of auto-bidding policies deployed in the advertising system must be guaranteed in either the stages of policy deployment or policy training. Therefore, safety of the RL training process is formally defined by a performance lower bound:

$$J(\pi_e^k) \geq (1 - \epsilon)J_s, \forall k = 1, \cdots, K,$$

where J_s is the performance of a known safe policy. [2]

4 PERFORMANCE BOTTLENECK OF ITERATIVE OFFLINE RL

In this section, we present empirical observations regarding the performance bottleneck of iterative offline RL, and also introduce the idea of our proposed method. In each iteration of iterative offline RL, the data-collection policy plays a crucial role in determining the input dataset for the subsequent offline training process, which in turn influences the trained policy. The data-collection process

---

[1] For simplicity, we assume all trajectories' lengths are equal to episode length T, though a campaign may exhaust its budget at certain time $t_0 < T$ and cannot afford any impression thereafter. In such cases, we let $r(s_t, a_t) = 0, \forall t_0 \leq t \leq T$.

[2] Note that the exploration policy π_e^k in every iteration k should be ensured to be safe. To satisfy the safety constraint in the first iteration, the training process should be initialized by a known safe (but could be suboptimal) policy instead of a random policy. In practice, policies trained in a simulation [12] could serve as an initial policy.
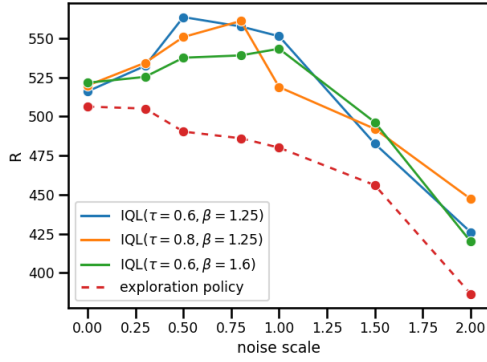
Figure 2: Performance of exploration policies with different noise scale, as well as the performance of policies trained with IQL on datasets collected by those exploration policies.

Figure 3: Comparison of trajectory return distributions of datasets collected by ASN and PSN.

should gather sufficient information of the underlying MDP through effective exploration.

A conventional approach for exploration in iterative offline RL [20] is adding noise perturbations to actions. Concretely, an exploration policy with action space noise (ASN) is constructed as $\pi_e^{ASN}(s_t) = \pi(s_t) + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2 I)$ is sampled from a Gaussian distribution with standard deviation σ. However, a highly random policy often leads to suboptimal performance and results in a dataset consisting primarily of low-quality actions. Consequently, such datasets pose challenges for offline RL algorithms to produce high-quality policies in the subsequent training process. This is because the conservatism principle of these algorithms drives the learned policy close to the low-performing exploration policy.

We conduct experiments in a simulated environment (details provided in Appendix D) to illustrate the aforementioned issue. Based on one deterministic policy, we construct exploration policies by adding different levels of noise perturbation on the actions. These exploration policies are deployed in the environment to collect interaction datasets, which are then utilized by IQL [15], an offline RL algorithm, to train new policies. Figure 2 presents the performance of exploration policies with varying noise scale, as well as performance of the trained policies. We can observe that adding a certain amount of noise boosts the trained policy by gaining more environment information, while an excessively noisy exploration policy undermines the training result. Furthermore, the optimal noise scale depends on the training algorithm, making online tuning of the noise scale impractical.

One potential method for overcoming the difficulty brought by conservatism is to manually identify high-quality behaviours from the noisy dataset and allow the learning algorithm to focus solely on these behaviours. However, evaluating the quality of individual actions within a dataset can be challenging. For one transition tuple (s, a, r, s'), a large reward r does not necessarily indicate a to be a good action, due to the influence of s' on future rewards. Hopefully, if a full trajectory τ attains a high return, then it is reasonable to infer that this trajectory contains high-quality behaviours. In following sections, we present empirical findings to reveal that
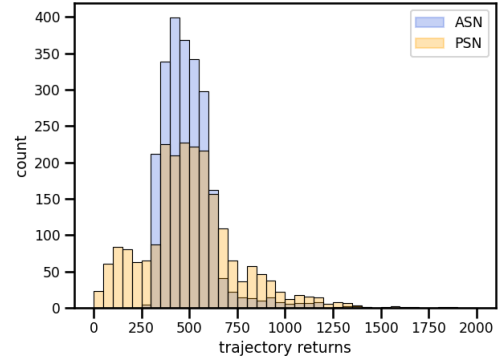
employing PSN instead of ASN for exploration leads to a dataset containing more high-return trajectories.

5 PROPOSED FRAMEWORK

In this section, we present our novel design on iterative offline RL framework for auto-bidding, which comprises two key components: TEE and SEAS.

5.1 Trajectory-wise Exploration

We introduce Parameter Space Noise (PSN) [8, 25] for exploration in the iterative offline RL framework, and explain its effectiveness from a trajectory view. PSN refers to injecting noise in an RL policy's parameter space in order to induce exploratory behaviours. It has brought performance gain on a wide range of control tasks when applied to online deep RL algorithms [1, 13]. For a parameterized policy (e.g. a neural network) $\pi(s; \theta)$, where θ is the parameter vector, applying additive Gaussian noise to θ gives $\hat{\theta} = \theta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Then the exploration policy based on PSN is $\pi_e^{PSN}(s_t) = \pi(s_t; \hat{\theta})$. Importantly, the perturbed parameter vector $\hat{\theta}$ is only sampled at the beginning of each episode and remains fixed afterwards. This is substantially different from ASN where independent noise is added at every time step.

We now present our key observation on datasets collected by PSN by experiments in a simulated bidding environment (details are provided in Appendix D). We first construct two exploration policies with ASN and PSN based on one policy π, and denote them by π_e^{ASN} and π_e^{PSN} respectively. Subsequently, two datasets D^{ASN} and D^{PSN} of equal size are collected with π_e^{ASN} and π_e^{PSN}. [3] As shown in Figure 3, the return distribution of trajectories in D^{PSN} is more dispersed than that of D^{ASN}. Specifically, D^{PSN} contains high-return trajectories (e.g. trajectories with return higher than 750) which are almost absent in D^{ASN}.

The high return variance observed in PSN can be attributed to two main factors: decoupling between the exploration policy and the base policy (i.e. the original noiseless policy π), and the consistency of exploration behaviours. The following example provides

---

[3]To ensure a fair comparison, we control the noise strength of both exploration policies to guarantee that the average return of D^{ASN} and D^{PSN} are equal.

further explanation in the context of auto-bidding, by highlighting the incapability of ASN on generating high-return trajectories. Assume that the current policy π is suboptimal in that it produces relatively low bids across most states. In this case, positive perturbations ϵ_t are desirable when applying ASN. However, when $\pi_e^{ASN}(s_t)$ is lifted, $\pi(s_{t+1})$ would become low since the base policy π aims to maintain the smoothness of budget consumption, thereby offsetting the effect of high bid in step t. The counterbalancing actions of the base policy substantially prevent ASN from effectively exploring unknown areas. Another issue with ASN is that, since the perturbations ϵ_t of different time steps are drawn from independent probability distributions, consistently realizing positive ϵ_t throughout an entire episode is barely possible.

PSN enables us to sample a variety of policy parameters, which could be distributed to a great number of advertising campaigns in a DSP to explore different bidding behaviours. These campaigns run in parallel and collect a large dataset covering a wide range of behaviours, thus achieving large-scale trajectory-wise exploration.

5.2 Trajectory-wise Exploitation

While PSN-based exploration policies could effectively generate valuable high-return trajectories, their collected datasets still consist primarily of inferior trajectories, as depicted in Figure 3. Therefore, the trained policy's performance is still limited by the inherent conservatism of offline RL. To alleviate this problem and fully exploit the dataset, we propose Robust Trajectory Weighting for trajectory-wise exploitation.

We consider a dataset containing multiple trajectories $D = \{\tau_i\}_{i=1}^{N}$, where $\tau_i = \{(s_{i,t}, a_{i,t}, s_{i,t+1}, r_{i,t})\}_{t=0}^{T}$. Inspired by [14, 32, 33], instead of uniform sampling, we assign large sample probabilities to well-performing trajectories during training. A straightforward way to realize this is to assign weight w_i for trajectory τ_i based on its return $R_i := R(\tau_i)$, as high return typically indicates good behaviours. Nonetheless, two main issues arise when applying return-based trajectory weighting in the auto-bidding task. Firstly, due to the instability of auction environments and user feedbacks, the reward function $r(s_t, a_t)$ is highly stochastic, thus a high return might not necessarily be achieved by a good policy, but rather a "lucky" trial that happens to obtain high rewards in most time steps. Secondly, trajectories within the dataset may come from different advertising campaigns, and the intrinsic characteristics of a campaign, such as its budget level and item category, also affect the trajectory return. Thus, directly comparing trajectories from different campaigns by their returns is unfair.

We address the first issue by learning a reward model $\bar{r}$ on the dataset for predicting the expected reward of a state-action pair:

$$\bar{r} = \arg\min_r \sum_{i=1}^{N} \sum_{t=0}^{T} (r(s_{i,t}, a_{i,t}) - r_{i,t})^2.$$

The reward model could be implemented by a neural network with the above loss function. Then the original stochastic rewards $r_{i,t}$ are replaced with their expectations $\bar{r}_{i,t} = \bar{r}(s_{i,t}, a_{i,t})$ to calculate $\bar{R}_i = \sum_{t=0}^{T} \gamma^t \bar{r}_{i,t}$, producing more robust quality indicators.

To deal with the second issue mentioned above, we regularize the trajectory returns by subtracting the value of the initial state of the trajectory, estimated as $\hat{V} = \arg\min_V \sum_{i=1}^{N} (V(s_{i,0}) - \bar{R}_i)^2$. The

---

Algorithm 1 Safe Exploration by Adaptive Action Selection (SEAS)

1: **Input:** Exploration policy π_e, n distinct safe policies $\{\pi_s^i\}_{i=1}^{n}$ and their state-action value function $\{Q^{\pi_s^i}\}_{i=1}^{n}$, safe performance J_s and safety coefficient $\epsilon \in (0, 1)$
2: **Initialize:** Sample initial state $s_0 \sim \rho(s)$, $temp \leftarrow 1$
3: **for** $t = 0, 1, \cdots, T$ **do**
4: $\quad a_e \leftarrow \pi_e(s_t), a_s \leftarrow \pi_s^{temp}(s_t)$
5: $\quad temp \leftarrow \arg\max_i Q^{\pi_s^i}(s_t, a_e), Q_{max} \leftarrow Q^{\pi_s^{temp}}(s_t, a_e)$
6: $\quad$ **if** $\sum_{u=0}^{t-1} r_u + Q_{max} \geq (1 - \epsilon)J_s$ **then**
7: $\quad\quad a_t \leftarrow a_e$
8: $\quad$ **else**
9: $\quad\quad a_t \leftarrow a_s$
10: $\quad$ **end if**
11: $\quad$ Take action a_t, observe r_t, s_{t+1}
12: **end for**

---

initial state typically contains information (*e.g.* the total budget) of the advertising campaign, therefore $\hat{V}(s_{i,0})$ provides estimation of the expected return of the campaign behind trajectory i. Our final indicator of trajectory quality is $\hat{A}_i$:

$$\hat{A}_i = (\bar{R}_i - \hat{V}(s_{i,0}))/\hat{V}(s_{i,0}), \quad \forall 1 \leq i \leq N.$$

The sample probability $w_{i,t}$ of transition tuple $(s_{i,t}, a_{i,t}, s_{i,t+1}, r_{i,t})$ is computed according to $\hat{A}_i$ as follows:

$$w_{i,t} \propto \exp(\hat{A}_i/\alpha), \quad \forall 1 \leq i \leq N, 0 \leq t \leq T,$$

where $\alpha \in \mathbb{R}^+$ is a temperature parameter, and the weights should be normalized to ensure $\sum_{i=1}^{N} \sum_{t=0}^{T} w_{i,t} = 1$.

After computing weights for dataset D, we could run any model-free offline RL algorithm (*e.g.* CQL, IQL) using the reweighted data sampling strategy. We provide a theoretical justification of Robust Trajectory Weighting in Appendix A, showing how it alleviates the problem brought by conservative algorithms.

5.3 Safe Exploration

Although TEE boosts the effectiveness of iterative offline RL, the safety of data-collecting policies, which is of great importance when training in real-world advertising systems, have not been considered. In this section, we propose a novel algorithm named SEAS to guarantee the safety of online exploration.

On the problem of safe exploration in auto-bidding, Mou et. al. [21] designed a method based on safety zone, to restrict the exploratory actions around a safe policy. Specifically, $\pi_e(s_t) \leftarrow clip(\pi_e(s_t), \pi_s(s_t) - \xi, \pi_s(s_t) + \xi)$, where π_s is a known safe policy. Though this method is provably safe under some assumptions on the MDP, its safety heavily relies on the radius ξ which is intractable to determine. Besides, this approach lacks awareness of the quality of original exploration actions, and poses constraint on both bad and good actions, thus hurting the quality of collected datasets. SEAS mitigates these problems through an adaptive design. The aim of SEAS is to prevent the low-performing trajectories caused by the original exploration policy (*e.g.* π_e^{PSN}) to emerge, while preserving the high-quality ones to the fullest extent.

The procedure of SEAS interacting with the environment for one episode is shown in Algorithm 1. In each step, SEAS selects

between an exploratory action a_e and a safe action a_s according to the condition in line 6. Note that multiple safe policies are provided to the algorithm, and in each step the safe policy with maximum Q value is chosen for constructing the condition. Utilizing multiple safe policies instead of one loosens the restriction in line 6, thus better preserving exploratory actions of π_e. The algorithm is "adaptive" in the sense that each action a_t depends on the rewards accumulated up to time t.

The following theorem shows that SEAS theoretically guarantees safety for any exploration policy π_e. Proof of Theorem 1 is provided in Appendix B.

THEOREM 1. *For any policy π_e and any $\epsilon \in (0, 1)$, given safe policies $\{\pi_s^i\}_{i=1}^n$ that satisfy $J(\pi_s^i) \geq J_s, \forall 1 \leq i \leq n$, the expected return $\mathbb{E}_\tau[R(\tau)]$ of trajectories generated by SEAS satisfies $\mathbb{E}_\tau[R(\tau)] \geq (1 - \epsilon)J_s$.*

The advantages of SEAS are summarized as follows: (i) SEAS needs only one hyperparameter ϵ, which is in the definition of safety and is straightforward to set. (ii) The safety of SEAS is provable without additional assumptions on the underlying MDP. (iii) Experiments demonstrate that when functioning together with TEE, SEAS exhibits minimal performance sacrifice compared to baseline methods.

TEE and SEAS together form an iterative RL framework for auto-bidding. Appendix C provides a detailed description of the overall procedure.

6 EXPERIMENTS

We provide empirical evidence for the effectiveness of our approach by both simulated experiments and real-world experiments on Alibaba display advertising platform.

6.1 Overall Performance in a Simulated Environment

A simulated advertising system is constructed for all the offline experiments in this work. Details of the setup and hyperparameters are provided in Appendix D.

We test the effectiveness of TEE and SEAS by combining them with three different offline RL algorithms. We also implement three baselines: iterative versions of IQL with and without exploration, as well as SORL [21]. An expert policy is trained with TD3 [11], an online RL algorithm, to serve as a performance upper bound. Methods for comparison are listed below.

- **TEE+SEAS+IQL [15] / CQL [17] / TD3BC [9].** Iterative offline RL with the proposed TEE and SEAS, using IQL / CQL / TD3BC as the offline RL algorithm.
- **IterIQL+ASN.** Iterative offline RL using IQL as the offline RL algorithm. IQL is selected as a representative because it is the state-of-the-art model-free offline RL algorithm. Exploration policy π_e^k is constructed by adding ASN on π^k.
- **IterIQL.** Iterative offline RL using IQL as the offline RL algorithm. No exploration noise is added, therefore $\pi_e^k = \pi^k$.
- **SORL [21].** SORL follows the iterative offline RL framework. The authors proposed V-CQL for offline policy training and designed an SER policy for safe and efficient exploration.
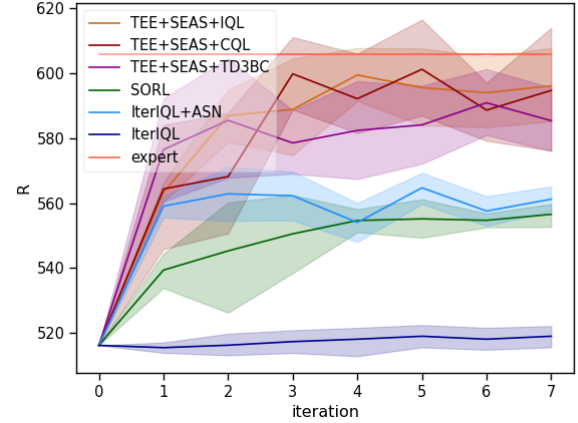

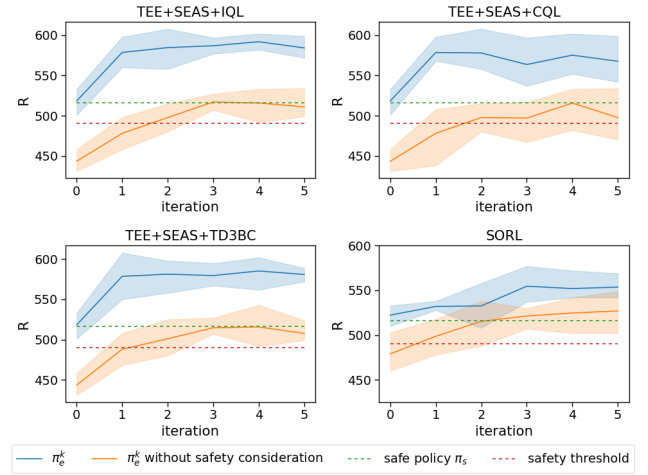**Figure 4: Overall performance in a simulated environment.**

Figure 5: Safety constraint satisfaction.

Evaluation Metrics. We evaluate the performance of a policy in terms of expected return. Besides, we check the safety of exploration policy by comparing average return in the collected dataset with the safety threshold $(1 - \epsilon)J_s$.

Figure 4 presents the overall performance through iterations. Our proposed framework, combined with any of the three offline RL algorithms, substantially outperforms the baseline methods and achieves near-expert performance in approximately 5 iterations. Both Iterative IQL+ASN and SORL get stuck in suboptimal policies after limited performance improvements, suffering from the performance bottleneck we discussed in section 4. Additionally, Iterative IQL without exploration achieves barely any performance improvement, which demonstrates the necessity of exploration in iterative offline RL.

Table 1: Performance of our proposed method in real-world experiments.

iteration	BuyCnt	ROI	CPA	ConBdg	GMV
1	+1.82%	+2.64%	-1.81%	-0.03%	+2.61%
2	+2.21%	+2.94%	-2.14%	+0.02%	+2.95%
3	+2.94%	+3.33%	-1.78%	+1.12%	+4.49%
4	+3.59%	+2.44%	-2.60%	+0.90%	+3.36%

We present the performance of exploration policies during the training process in Figure 5. Combined with any offline RL algorithm, our framework consistently ensures the performance of exploration policies to be above the safety threshold $(1 − \epsilon)J_s$, which validates the safety guarantee ability of SEAS. We also show the results when SEAS is omitted, in which the safety constraint is violated in early stages.

6.2 Online Experiments

We conduct real-world experiments on Alibaba display advertising platform. In each iteration, we utilize 20000 advertising campaigns to collect data for an epsiode, and employ IQL for offline training. Each episode contains 48 time steps, which means that the bidding parameter is adjusted every 30 minutes. After each policy update, we conduct a 10-day online A/B test using 1500 campaigns.

Evaluation Metrics. The return of the trained policy acts as our main metric of performance, and is referred to as BuyCnt in the online experiments. Additionally, we introduce several other metrics that are commonly used in the auto-bidding field.

- **BuyCut.** The total value of ad impressions won by the advertiser. Our objective J in the RL formulation.
- **ROI.** The ratio between the total revenue and the consumed budget of the advertiser.
- **CPA.** Cost per aquisition, defined as the average cost for each successfully converted impression. A smaller CPA indicates a better performance of an auto-bidding policy.
- **ConBdg.** The total consumed budget of the advertiser.
- **GMV.** Gross Merchandise Volume, the total amount of sales over the campaign duration.

Table 1 shows the performance of our method in each iteration, compared with a static baseline policy trained by CQL[17] on a pre-collected dataset. We can see that the BuyCnt of our policy improves steadily through iterations, and our method consistently outperforms the baseline across all metrics.

6.3 Ablation Studies

We conduct ablation studies for a deeper analysis of the how different components work with each other in our method. Specifically, we aim to answer the following questions: (1) Do trajectory-wise exploration and trajectory-wise exploitation operate in close conjunction instead of being two independent components? (2) Does the reward model effectively reduce the influence of stochastic rewards in Robust Trajectory Weighting? (3) Does SEAS achieve the theoretical safety bound in practice? (4) While achieving the same

safety bound, does SEAS sacrifices less performance than other baseline safety-guaranteeing methods?

To answer these questions, we conduct extensive experiments in the simulated environment described in Section 6.1.

To answer Question 1. We develop variants of TEE to delve deeper into how trajectory-wise exploration and trajectory-wise exploitation work together. We focus on one data-collection process followed by one offline training process. We omit SEAS in this experiment, to focus solely on TEE. An IQL policy is used as base policy π. Details of the variants are presented as follows.

- **TEE** is implemented as described.
- **w/o T-explore** removes trajectory-wise exploration (i.e. PSN). Instead, traditional ASN is used for π_e. To ensure a fair comparison, we control the strength of ASN to guarantee that the average return in D are equal to that of PSN.
- **w/o T-exploit** removes trajectory-wise exploitation (i.e. Robust Trajectory Weighting). After collecting D by PSN, we train a new policy with uniform sampling.
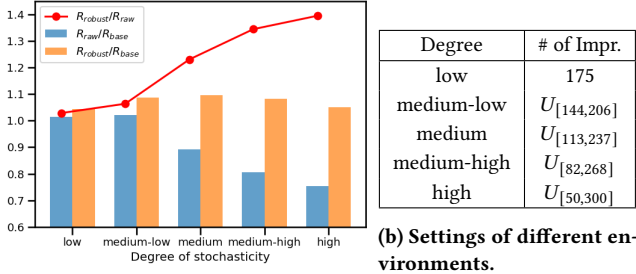- **w/o TEE** removes TEE. ASN is used for exploration, and uniform sampling for training.

Table 2 presents the performance of policies trained under different settings, and their performance gain over the base policy. We observe that TEE achieves significant performance improvement over the base policy under various budgets, while the absence of either component hurts its effectiveness. Interestingly, eliminating Trajectory-wise Exploitation leads to a substantial performance degradation. The reason behind this is that datasets collected by PSN contains a large fraction of undesirable actions, which are imitated by conservative algorithms. The above observation indicates that Trajectory-wise Exploration and Trajectory-wise Exploitation are inherently interconnected components, and their combination contributes substantially to the enhancement of the algorithm's performance.

To answer Question 2. We test the effectiveness of reward model in environments with different degrees of stochasticity. We first construct simulated environments with various stochasticity by controlling the variance of impression numbers per time step. In each environment, we collect an exploratory dataset with the same policy, and use two different data sampling strategies to train a policy on the dataset: a) Our proposed robust trajectory weighting. b) Trajectory weighting with the raw rewards $r_{i,t}$ instead of the reward model's prediction $\bar{r}_{i,t}$. We compare the performance of trained policies, to examine how reward model benefits the trajectory weighting method.

The results of the experiments are shown in Figure 6a, where R_{robust} denotes the return of policy trained with Robust Trajectory Weighting, R_{raw} represents the return of policy trained with raw rewards, and R_{base} the return of the data-collecting policy. Figure 6b shows the probability distributions of the impression number, where $U_{[a,b]}$ denotes a uniform distribution over $[a, b]$. The red line in Figure 6a indicates that the reward model is increasingly useful as the stochasticity of the environment intensifies. Moreover, the blue bars in the figure exhibit values lower than 1 in high stochasticity instances, which reflects that raw stochastic rewards could be misleading signals for trajectory weighting.

Table 2: Ablation study on TEE. The last column presents the performance improvement over base policy. Best performance of each column is marked as bolded.

Ablation Settings	Budget					#Improve
	1500	2000	2500	3000	avg	
Base policy	448.39	514.89	580.18	657.96	550.36	-
TEE	**490.19±13.38**	**569.35±5.89**	**634.88±31.88**	671.68 ± 59.69	**591.53±24.79**	**+7.48%**
w/o T-explore	431.09 ± 11.4	520.12 ± 12.05	594.14 ± 13.46	663.18 ± 14.97	552.13 ± 7.68	+0.32%
w/o T-exploit	365.53 ± 10.55	414.21 ± 29.05	495.89 ± 39.82	571.24 ± 51.27	461.72 ± 29.96	-16.10%
w/o TEE	414.38 ± 30.74	515.66 ± 10.35	607.47 ± 8.91	**678.67±17.53**	554.04 ± 7.13	+0.66%

(a) Improvements of policy performance.

Degree	# of Impr.
low	175
medium-low	$U_{[144,206]}$
medium	$U_{[113,237]}$
medium-high	$U_{[82,268]}$
high	$U_{[50,300]}$

(b) Settings of different environments.

Figure 6: The effectiveness of reward model in environments with different degrees of stochasticity.

Table 3: Safety-ensuring ability of SEAS.

ϵ	0.4	0.3	0.2	0.1	0.05	0.01
$1 - J(\pi_e^k)/J_s$	0.202	0.137	0.039	-0.002	-0.004	-0.005

To answer Question 3. We fully evaluate the safety-ensuring ability of SEAS by setting different values of ϵ and observe the rate of performance decrease $1 - J(\pi_e^k)/J_s$, where π_e^k is the policy produced by SEAS. We expect the safety constraint $1 - J(\pi_e^k)/J_s \leq \epsilon$ to be satisfied. In this experiment, we take a USCB [12] policy as the safe policy, and obtain its Q function through fitting a dataset collected by itself.

From Table 3, we observe that SEAS consistently satisfies the safety constraint over a wide range of input ϵ. Interestingly, for small ϵ values, the exploration policy generated by SEAS even outperforms the base policy.

To answer Question 4. We compare SEAS with two baseline safety-ensuring methods in terms of performance sacrificing, while ensuring safety to the same degree ($\epsilon = 0.05$). One USCB [12] policy is leveraged as the safe policy. In this experiment, we start from an IQL policy, preserve the design of TEE, and substitute SEAS with baselines presented below. Strength of PSN is $\sigma = 0.05$.

- **Small noise.** We limit the strength of PSN at a low level, by setting $\sigma = 0.01$.

Table 4: Different safety ensuring methods' impact on performance of trained policy. The third column presents the performance improvement of trained policy over base policy.

Safe Exploration Methods	Return	#Improve
No constraint	594.25 ± 18.28	+15.16%
SEAS	**572.35± 20.42**	**+10.92%**
Small noise	528.07 ± 19.45	+2.34%
Fixed range	532.91 ± 6.74	+3.28%

- **Fixed range.** As proposed in [21], the safe exploratory action a_e is given by $a_e \leftarrow clip(\pi_e(s_t), \pi_s(s_t) - \xi, \pi_s(s_t) + \xi)$. The safe policy π_s is the same as that in SEAS, and ξ is set to 0.1.

The results are presented in Table 4. We also show the result of not imposing any safety constraint, which is a performance upper bound. SEAS significantly outperforms the two baselines in terms of the average return of the trained policy. This observation suggests that the adaptive design of SEAS allows for minimal performance sacrifice.

7 CONCLUSION

In this work, we have presented a new iterative RL framework for auto-bidding from a trajectory perspective. We also pay particular attention on the safety of exploration in online advertisement systems, and propose SEAS for this issue. Through comprehensive experiments, our method has been shown to be effective, achieving superior results compared to other baselines. In future work, we plan to test the effectiveness of TEE in other fields such as recommender systems and healthcare, where online policy training is challenging but urgently needed.

ACKNOWLEDGMENTS

REFERENCES

[1] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. 2020. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*. PMLR, 507–517.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[3] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. 2021. Offline rl without off-policy evaluation. *Advances in neural information processing systems* 34 (2021), 4933–4946.

[4] Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo. 2017. Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the tenth ACM international conference on web search and data mining*. 661–670.

[5] Jinglin Chen and Nan Jiang. 2019. Information-Theoretic Considerations in Batch Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 1042–1051. https://proceedings.mlr.press/v97/chen19e.html

[6] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. 2004. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems* 17 (2004).

[7] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2007. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review* 97, 1 (2007), 242–259.

[8] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. 2018. Noisy Networks For Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[9] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.

[10] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.

[11] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1587–1596. https://proceedings.mlr.press/v80/fujimoto18a.html

[12] Yue He, Xiujun Chen, Di Wu, Junwei Pan, Qing Tan, Chuan Yu, Jian Xu, and Xiaoqiang Zhu. 2021. A unified solution to constrained bidding in online display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2993–3001.

[13] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[14] Zhang-Wei Hong, Pulkit Agrawal, Remi Tachet des Combes, and Romain Laroche. 2023. Harnessing Mixed Offline Reinforcement Learning Datasets via Trajectory Weighting. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

[15] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*.

[16] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

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

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

[19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[20] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. 2021. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[21] Zhiyu Mou, Yusen Huo, Rongquan Bai, Mingzhou Xie, Chuan Yu, Jian Xu, and Bo Zheng. 2022. Sustainable Online Reinforcement Learning for Auto-bidding. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[22] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *International conference on machine learning*. PMLR, 2721–2730.

[23] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*. PMLR, 2778–2787.

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

[25] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. 2018. Parameter Space Noise for Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[26] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[27] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. 2021. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems* 34 (2021), 11702–11716.

[28] Kajetan Schweighofer, Marius-constantin Dinu, Andreas Radler, Markus Hofmarcher, Vihang Prakash Patil, Angela Bitto-Nemling, Hamid Eghbal-zadeh, and Sepp Hochreiter. 2022. A dataset perspective on offline reinforcement learning. In *Conference on Lifelong Learning Agents*. PMLR, 470–517.

[29] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[30] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

[31] Di Wu, Xiujun Chen, Xun Yang, Hao Wang, Qing Tan, Xiaoxun Zhang, Jian Xu, and Kun Gai. 2018. Budget constrained bidding by model-free reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1443–1451.

[32] Yang Yue, Bingyi Kang, Xiao Ma, Gao Huang, Shiji Song, and Shuicheng Yan. 2023. Offline Prioritized Experience Replay. *arXiv preprint arXiv:2306.05412* (2023).

[33] Yang Yue, Bingyi Kang, Xiao Ma, Zhongwen Xu, Gao Huang, and Shuicheng Yan. 2022. Boosting offline reinforcement learning via data rebalancing. *arXiv preprint arXiv:2210.09241* (2022).

[34] Wenhao Zhan, Baihe Huang, Audrey Huang, Nan Jiang, and Jason Lee. 2022. Offline Reinforcement Learning with Realizability and Single-policy Concentrability. In *Proceedings of Thirty Fifth Conference on Learning Theory (Proceedings of Machine Learning Research, Vol. 178)*, Po-Ling Loh and Maxim Raginsky (Eds.). PMLR, 2730–2775. https://proceedings.mlr.press/v178/zhan22a.html

[35] Ruiyi Zhang, Tong Yu, Yilin Shen, and Hongxia Jin. 2022. Text-Based Interactive Recommendation via Offline Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11694–11702.

[36] Weinan Zhang, Kan Ren, and Jun Wang. 2016. Optimal real-time bidding frameworks discussion. *arXiv preprint arXiv:1602.01007* (2016).

[37] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1077–1086.

A THEORETICAL JUSTIFICATION OF ROBUST TRAJECTORY WEIGHTING

We formally show that under the assumption of deterministic transition and stochastic reward, applying Robust Trajectory Weighting is equivalent to training offline RL on a dataset collected by a better behaviour policy.

For a dataset $D = \{\tau_i\}_{i=1}^{N}$ where $\tau_i = \{(s_{i,t}, a_{i,t}, s_{i,t+1}, r_{i,t})\}_{t=0}^{T}$. Each trajectory τ_i is collected by a different deterministic policy π_i, as in the case of Trajectory-wise Exploration. The behaviour policy π of D is then defined as sampling a policy from $\{\pi_i\}_{i=1}^{N}$ uniformly at the start of an episode, then acting according to the sampled policy till the end of the episode. Similarly, we define a weighted behaviour policy π' as first sampling a policy from $\{\pi_i\}_{i=1}^{N}$ according to probabilities $\{w_i\}_{i=1}^{N}$, then acting with it. We aim to show that $J(\pi') \geq J(\pi)$.

The performance $J(\pi)$ could be expressed as expectation of value function over all possible initial states:

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho}[V^{\pi}(s_0)], J(\pi') = \mathbb{E}_{s_0 \sim \rho}[V^{\pi'}(s_0)]. \quad (1)$$

For any initial state s_0, let $G_{s_0} = \{i|s_{i,0} = s_0\}$, $N_{s_0} = |G_{s_0}|$, we have

$$V^{\pi}(s_0) = \sum_{i \in G_{s_0}} V^{\pi_i}(s_{i,0})/N_{s_0}. \quad (2)$$

Similarly, for weighted behaviour policy π',

$$V^{\pi'}(s_0) = \sum_{i \in G_{s_0}} w_i V^{\pi_i}(s_{i,0}) / \sum_{i \in G_{s_0}} w_i. \quad (3)$$

Assuming deterministic transition and stochastic rewards of the underlying MDP, we have $V^{\pi_i}(s_{i,0}) = \bar{R}_i$, where $\bar{R}_i = \sum_{t=0}^{T} \gamma^t \bar{r}_{i,t}$ is the relabeled return in Robust Trajectory Weighting. Plugging it in (2) and (3) gives us

$$V^{\pi}(s_0) = \sum_{i \in G_{s_0}} \bar{R}_i/N_{s_0}, \quad (4)$$

$$V^{\pi'}(s_0) = \sum_{i \in G_{s_0}} w_i \bar{R}_i / \sum_{i \in G_{s_0}} w_i. \quad (5)$$

Subtracting (4) by (5),

$$V^{\pi'}(s_0) - V^{\pi}(s_0) \quad (6)$$

$$= \sum_{i \in G_{s_0}} w_i \bar{R}_i / \sum_{i \in G_{s_0}} w_i - \sum_{i \in G_{s_0}} \bar{R}_i/N_{s_0} \quad (7)$$

$$= \frac{1}{N_{s_0} \sum_{i \in G_{s_0}} w_i} (N_{s_0} \sum_{i \in G_{s_0}} w_i \bar{R}_i - (\sum_{i \in G_{s_0}} w_i)(\sum_{i \in G_{s_0}} \bar{R}_i)) \quad (8)$$

$$= \frac{1}{N_{s_0} \sum_{i \in G_{s_0}} w_i} (\sum_{i,j \in G_{s_0}, i<j} (w_i - w_j)(\bar{R}_i - \bar{R}_j)). \quad (9)$$

In (9), each term $(w_i - w_j)(\bar{R}_i - \bar{R}_j)$ inside the summation is non-negative, because

$$w_i = \exp((\bar{R}_i/V(s_{i,0}) - 1)/\alpha)/Z,$$

where $Z = \sum_{i=1}^{N} w_i$ is a normalization term and $\alpha \in \mathbb{R}^+$, is non-decreasing with respect to $\bar{R}_i$. Therefore, $V^{\pi'}(s_0) - V^{\pi}(s_0) \geq 0$ for every initial state s_0. Then applying (1) gives $J(\pi') \geq J(\pi)$.

The above derivation also highlights the significance of our proposed reward model. Relabeling rewards with the reward model's output makes $V^{\pi_i}(s_{i,0}) = \bar{R}_i$, allowing us to deal with stochastic rewards.

B PROOF OF THEOREM 1

THEOREM 1. *For any policy π_e and any $\epsilon \in (0,1)$, given safe policies $\{\pi_s^i\}_{i=1}^{n}$ that satisfy $J(\pi_s^i) \geq J_s, \forall 1 \leq i \leq n$, the expected return $\mathbb{E}_\tau[R(\tau)]$ of trajectories generated by SEAS satisfies $\mathbb{E}_\tau[R(\tau)] \geq (1 - \epsilon)J_s$.*

PROOF. In each time step, SEAS takes either action a_e or a_s. For a trajectory τ generated by SEAS, let t_0 be the last step it takes exploratory action a_e. For those trajectories where it never takes a_e, set t_0 to 0. Let $temp_{t_0}$ denote the value of $temp$ at step t_0. Let $\mathbb{1}(\tau, s, a, i)$ be the indicator function of s_{t_0}, a_{t_0} and $temp_{t_0}$ for trajectory τ, defined as follows:

$$\mathbb{1}(\tau, s, a, i) = \begin{cases} \delta(s - s_{t_0}, a - a_{t_0}), & \text{if } i = temp_{t_0} \\ 0, & \text{otherwise}, \end{cases}$$

where $\delta(\cdot)$ is the Dirac delta function. Therefore

$$\forall \tau, \quad \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \mathbb{1}(\tau, s, a, i) dads = 1. \quad (10)$$

Multiply both sides of (10) by $R(\tau)$,

$$\forall \tau, \quad \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} R(\tau)\mathbb{1}(\tau, s, a, i) dads = R(\tau).$$

Take expectation with respect to τ on both sides,

$$\mathbb{E}_\tau[R(\tau)] = \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \mathbb{E}_\tau[R(\tau)\mathbb{1}(\tau, s, a, i)] dads.$$

Split trajectory τ by t_0, and define $R(\tau^-) = \sum_{u=0}^{t_0-1} r_u$, $R(\tau^+) = \sum_{u=t_0}^{T} r_u$, then

$$\mathbb{E}_\tau[R(\tau)] = \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \mathbb{E}_\tau[(R(\tau^-) + R(\tau^+))\mathbb{1}(\tau, s, a, i)] dads$$

$$= \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \{\mathbb{E}_\tau[R(\tau^-)\mathbb{1}(\tau, s, a, i)]$$
$$+ \mathbb{E}_\tau[R(\tau^+)\mathbb{1}(\tau, s, a, i)]\} dads.$$

For trajectory τ, since t_0 is the last time it takes a_e, all actions after t_0 follows safe policy $\pi_s^{temp_0}$. Therefore $E_\tau[R(\tau^+)\mathbb{1}(\tau, s, a, i)] = Q^{\pi_s^i}(s,a)\mathbb{1}(\tau, s, a, i)$, then

$$\mathbb{E}_\tau[R(\tau)] = \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \{\mathbb{E}_\tau[R(\tau^-)\mathbb{1}(\tau, s, a, i)]$$
$$+ Q^{\pi_s^i}(s,a)\mathbb{1}(\tau, s, a, i)\} dads$$

$$= \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \mathbb{E}_\tau[(R(\tau^-) + Q^{\pi_s^i}(s,a))\mathbb{1}(\tau, s, a, i)] dads$$

$$\geq \sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} \mathbb{E}_\tau[(1 - \epsilon)J_s\mathbb{1}(\tau, s, a, i)] dads$$

$$= \mathbb{E}_\tau[\sum_{i=1}^{n} \int_{\mathcal{S}} \int_{\mathcal{A}} (1 - \epsilon)J_s\mathbb{1}(\tau, s, a, i) dads]$$

$$= \mathbb{E}_\tau[(1 - \epsilon)J_s]$$

$$= (1 - \epsilon)J_s,$$

where the inequality step is by line 6 in algorithm 1, and the following steps are from (10) and simple algebra. □

C OVERALL ITERATIVE FRAMEWORK

TEE and SEAS are combined to form an iterative framework for online policy training in auto-bidding. The training process is initialized with the current policy running in the bidding system, which is typically suboptimal but safe. In each iteration k, we employ PSN for exploration based on the current policy π^k. We pick a large number of advertising campaigns, and independently sample parameter vectors for different campaigns' policies. The exploration policies are input to SEAS to guarantee safety. A subset of previous policies $\{\pi^\kappa\}_{\kappa=1}^{k-1}$ can be utilized as safe policies for SEAS, and ϵ is a pre-defined constant parameter through iterations. The Q functions of safe policies for SEAS could be obtained through different approaches. For policies trained by value-based or actor-critic RL algorithms, we can directly query the existing value network. Alternatively, new value networks can be fitted on the collected datasets. Specifically, for approximating the Q function of π^κ, we can perform SARSA-style policy evaluation on dataset D^κ. Although this yields underestimated values due to exploration when collecting D^κ, it does not incur violation of the safety constraint, because Theorem 1 still holds even when the input Q functions have underestimated values. The safe exploration policies are then distributed and deployed to multiple advertising campaigns, running in parallel for a specific duration (*e.g.* one day) to collect an interaction dataset D^k. Subsequently, we employ an offline RL algorithm (*e.g.* IQL) to perform policy training on D^k, where the sampling probabilities are computed using Robust Trajectory Weighting. The resulting trained policy π^{k+1} becomes the input for the subsequent iteration, and the process continues iteratively.

D DETAILS OF OFFLINE EXPERIMENTS

Setup. For offline experiments, we construct a simulated advertising system mainly based on the simulated system in [21]. There are 30 advertisers competing for advertising impressions. An episode corresponds to one day in simulation, which is divided into 96 time steps. The number of impressions in each time step is random, and follows a uniform distribution on [50, 300]. The budget of each advertiser follows a uniform distribution on [1500, 3000]. Before an episode starts, the number of impressions in every time step, as well as the value of each impression for each advertiser is initialized. The state of an advertiser is three-dimensional: $[time, budget\_consumed, budget\_left]$. The reward of an advertiser is the total value she wins in all ad auctions during one time step. Given current states and actions of all advertisers, the simulation returns next states and rewards. This is achieved by simulating ad auctions for all impressions in a single time step. For each impression, the system performs pre-ranking and ranking, and decides the winner of the auction. The winner gets the value of the impression, and pays for it according to the auction mechanism. We train a bidding policy for one advertiser while keeping other 29 advertisers' policy fixed. Therefore, the policies of other bidders could be seen as part of the environment which is stationary. The trained auto-bidding policy could serve for bidders with different budgets since the information of total budget is contained in the state representation. The implementation of the pre-ranking and ranking(auction) part follows from that in [21], where more details could be found.

Parameter Settings. Datasets collected in each iteration consists of 100000 transition tuples. Strength of trajectory weighting α is set to 0.1. Safety threshold ϵ is 0.05. PSN is implemented as factorised Gaussian noise [8] with σ searched from [0.01,0.03,0.05] and kept fixed during training. ASN is Gaussian noise with σ searched from [0.3,0.5,1]. In the IQL algorithm, the expectile parameter is set to 0.6 and β is 1.25. Conservative factor is chosen as $\alpha = 0.8$ in CQL, and $\alpha = 2.5$ in TD3+BC. Implementation of SORL is consistent with the original paper [21] and code.