A Novel Benchmark for Decision-Making in Uncertain and Competitive Games

Kefan Su[1,2*], Yusen Huo[2], Zhilin Zhang[2], Shuai Dou[2], Chuan Yu[2], Jian Xu[2], Zongqing Lu[1†], Bo Zheng[2]

[1]School of Computer Science, Peking University

[2]Alibaba Group

[1]{sukefan,zongqing.lu}@pku.edu.cn

[2] {huoyusen.huoyusen,zhangzhilin.pt,doushuai.ds, yuchuan.yc,xiyu.xj,bozheng}@alibaba-inc.com

Abstract

The study of decision-making in large-scale game environments is a crucial domain within artificial intelligence, possessing substantial implications for practical applications. Nevertheless, the lack of comprehensive, realistic game environments and associated datasets has limited progress in this field. To address this and promote research on this important problem, we introduce the Large-Scale Auction (LSA) Benchmark derived from online advertising, a rapidly expanding industry worth $626.8 billion in 2023. The LSA Benchmark consists of an environment and the corresponding dataset. The LSA Environment is augmented with the deep generative model to reduce the gap between the simulation environment and reality while avoiding the risks of sensitive data exposure. The LSA Dataset comprises over **500 million** records, totaling **40 GB** in size, which contains trajectories with **50** diverse agents competing with each other, for effective offline training. We evaluate different types of existing algorithms in the LSA Environment. We hope the LSA benchmark can promote the development of decision-making in large-scale games.

1 Introduction

In the domain of artificial intelligence, decision-making is a fundamental area of research with broad real-world implications, particularly in large-scale games such as online advertising [32], e-commerce [34], financial markets [9], and energy trading [31]. In large-scale games, agents must frequently make strategic decisions to fulfill their objectives under resource constraints in an uncertain and competitive environment, which is marked by substantial randomness and massive quickly shifting competitor strategies. A key challenge faced by academic researchers in this domain is the lack of **large-scale**, **realistic** game environments and datasets, due primarily to restricted data access within the industry. This challenge limits researchers' abilities to investigate this problem and verify their strategies, restricting the development of technologies in this field.

Auto-bidding is a typical large-scale sequential decision-making problem in online advertising, which reached a market size of $626.8 billion in 2023[3]. Auto-bidding aims to provide bidding services for advertisers and has become a hot research topic in the field of online advertising [8, 14, 21]. As shown in Figure 1, the auto-bidding agent sequentially bids for each impression across a large number of continuously arriving opportunities to maximize performance, adhering to certain constraints on behalf of the advertiser.

[*]This work is done during internship at Alibaba Group.

[†]Corresponding author.

[3]https://www.statista.com/statistics/237974/online-advertising-spending-worldwide

38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks.

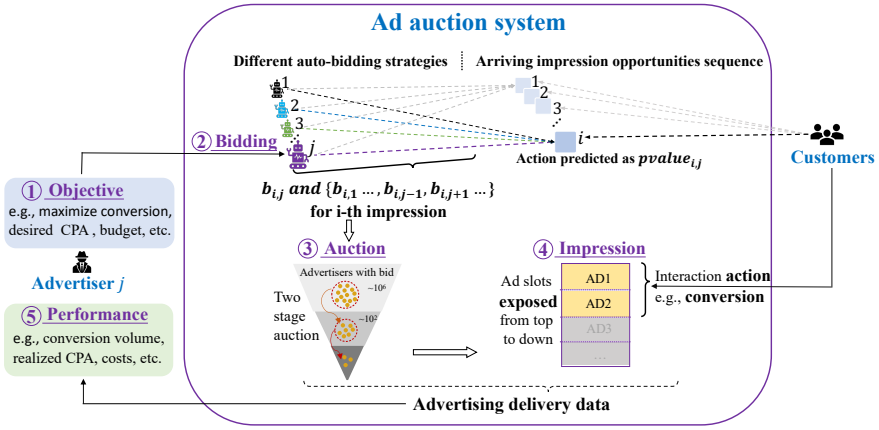

Figure 1: Overview of typical large-scale online advertising platform. Numbers 1 through 5 illustrate how an auto-bidding agent helps advertiser j optimize performance. For each advertiser's unique objective (①), auto-bidding agents make bidding decisions (②) for continuously arriving opportunities, and compete against each other in the auction (③). Then, each agent may win some impressions (④), which may be exposed to customers and potentially result in conversions. Finally, the agents' performance (⑤) will be reported to advertisers.

Though there are unique properties or difficulties of auto-bidding, existing simulation platforms do not reflect these properties of the auto-bidding problem. AuctionGym [14] omits the budget changes of advertisers in multiple auction rounds. AdCraft [8] models other bidders by sampling from a parameterized distribution which cannot completely present the multi-agent property essence of this problem.

Furthermore, previous studies lack discussions about reducing the gap between simulation platforms and the real world. Recently, the deep generative model has made great progress and shown a powerful ability in the language and vision fields [22, 13, 23], which inspires us to model the distribution of real-world impressions with a deep generative model. Additionally, there are many works using deep generative model for synthetic data of real-word applications such as [2, 19, 18] covering the fields of health and geographic information systems.

Therefore, this paper introduces the Large-Scale Auction (LSA) Benchmark, a typical large-scale game-based simulation environment derived from auto-bidding in online advertising. The LSA benchmark consists of the LSA Environment and the LSA Dataset. The LSA Environment is a realistic simulation platform for large-scale decision-making. The LSA Dataset contains a large amount of diverse offline data for effective offline training. We implement and evaluate baseline algorithms including PID Controller[33], Online LP[11], IQL[17], Behavior Cloning[27], and Decision Transformer[6] in the LSA Environment.

In summary, our contributions are as follows:

- We provide the LSA Environment, a realistic data-driven multi-agent simulation environment for auto-bidding augmented with a deep generative model. We also verify the similarity between the generated and real-world data by experiments.
- We provide the LSA Dataset, which comprises over **500 million** records, totaling **40 GB** in size. The LSA Dataset contains trajectories with **50** diverse agents competing with each other for effective offline training.
- We evaluate different types of existing algorithms for auto-bidding and our empirical results provide meaningful insights for decision-making in uncertain and competitive games.

2 The LSA Environment

The LSA Environment models the auto-bidding problem with standardized representations. In this section, we first provide the problem formulation in the LSA Environment. Based on the formulation,

we introduce two main tasks of the LSA Environment. Then we briefly introduce the modules of the LSA Environment. Finally, we discuss the implemented baselines and provide an example code for the environment API of the LSA Environment.

2.1 Problem Formulation

We use a Partially Observable Stochastic Game (POSG) $\mathcal{M}$ [10] to model the auto-bidding problem in one day. A POSG $\mathcal{M}$ is a tuple $\mathcal{M} = \{S, A, P, \boldsymbol{r}, \gamma, Y, O, I, T\}$. $I = \{1, 2, \cdots, n\}$ is the set of all agents. T is the horizon, *i.e.*, the number of time steps in one episode. S is the state space and A is the action space. $P : S \times A \times S \to [0, 1]$ is the transition probability. γ is the discounted factor. Y is the observation space and $O(s, i) : S \times I \to Y$ is the mapping from state to observation for each agent i. $\boldsymbol{r} = r_1 \times r_2 \times \cdots \times r_n$ is the joint reward function of all agents and $r_i(s, \boldsymbol{a}) : S \times A \to \mathbb{R}$ is the individual reward function for each agent i. Given this definition, we can further introduce our problem formulation.

Horizon. We focus on modeling the auto-bidding problem in one time cycle and the time cycle is divided into several decision steps. For instance, we can model the auto-bidding problem in one day and divide the 24 hours into 48 or 96 time steps which means $T = 48$ or $T = 96$. One time step corresponds to all the ad impression opportunities in the corresponding time interval.

State Space. The state $s = (\boldsymbol{b}, \boldsymbol{f}, \boldsymbol{v})$ in one time step contains three parts of information: the budgets of all agents $\boldsymbol{b} = (b_1, b_2, \cdots, b_n)$ where b_i is the budget of agent i in state s; the features of all the ad impression opportunities $\boldsymbol{f} = (f^1, f^2, \cdots, f^m)$ where m is the number of the ad impression opportunities in the time interval and f^j is the feature of the corresponding ad impression; the values of different impression opportunities for different agents $\boldsymbol{v} = \{v_i^j\} \in \mathbb{R}^{n \times m}$, where v_i^j is the value of opportunity j for agent i. Agents in this environment represent advertisers who may have different preferences, so each ad impression opportunity has different values for different agents.

Action Space. The action of agent i is a bid rate $a_i = \alpha_i$ which means the bid of agent i for all the ad impression opportunities is $bid_i = (\alpha_i * v_i^1, \alpha_i * v_i^2, \cdots, \alpha_i * v_i^m)$. Though agents can choose any bid rate as their actions, the bids that exceed the budget will be invalid.

Reward Function. Given the bids of all agents $\{bid_i\}_{i=1}^n$, determined by the auction mechanism which will be introduced later, agent i will receive the auction result $x_i = (x_i^1, x_i^2, \cdots, x_i^m)$ where $x_i^j = 1$ if and only if agent i wins opportunity j otherwise $x_i^j = 0$. Agent i will only receive a reward from the opportunities won by itself, *i.e.*, $r_i(s, \boldsymbol{a}) = \sum_{j=1}^m x_i^j v_i^j$.

Transition Function. The ad impression opportunity j's cost c^j is determined by the winning price in the auction and the auction mechanism. The budget of agent i will updated by $\hat{b}_i = b_i - \sum_{j=1}^m x_i^j c^j$. Finally, in the next time steps, new ad impression opportunities and corresponding values will be generated $\hat{\boldsymbol{f}} = (\hat{f}^1, \hat{f}^2, \cdots, \hat{f}^{\hat{m}}), \hat{\boldsymbol{v}} = \{\hat{v}_i^j\}$, so the new state will be $s' = (\hat{\boldsymbol{b}}, \hat{\boldsymbol{f}}, \hat{\boldsymbol{v}})$. However, the observation of agent i only contains partial information such as its budget and the auction result. Specifically, the observation of agent i $o_i = (b_i, \boldsymbol{f}, \boldsymbol{v})$. As for actions, Agents can only observe its own action α_i and the bidding results for each impression x_i^j. Agent i can use history information to reduce the effect of the partial observability. As for the issue of randomness, the transition function of the LSA environment is not degenerate since the volume and values of the impression opportunities have randomness in each step. Additionally, the bidding strategies of other agents also have a degree of randomness which will also influence the state transition.

2.2 Tasks

Though LSA provides a general framework for auto-bidding problem studies, we choose two typical scenarios in auto-bidding as basic tasks in LSA for easier understanding.

2.2.1 Budget Constrained Bidding

Budget Constrained Bidding (BCB) [30] is a basic scenario in auto-bidding, where agents maximize their obtained values within the constraint on the budget. The optimization formulation of BCB from

agent i's perspective is as follows:

$$\begin{aligned} &\max \sum_{t=1}^{T} \left\langle \boldsymbol{x}_i^t, \boldsymbol{v}_i^t \right\rangle \\ &\text{s. t.} \sum_{t=1}^{T} \left\langle \boldsymbol{x}_i^t, \boldsymbol{c}^t \right\rangle \leq b_i, \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i^t = (x_i^{t,1}, x_i^{t,2}, \cdots, x_i^{t,m})$ is the auction result of all impression opportunities for agent i in time step t, $\boldsymbol{v}_i^t = (v_i^{t,1}, v_i^{t,2}, \cdots, v_i^{t,m})$ is the value for agent i, $\boldsymbol{c}^t = (c^{t,1}, c^{t,2}, \cdots, c^{t,m})$ is the cost in time step t, b_i is the budget for agent i, and $\langle \cdot \rangle$ is the inner product.

As for the implementation, we know from our problem formulation that $r_i(s_t, \boldsymbol{a}_t) = \langle \boldsymbol{x}_i^t, \boldsymbol{v}_i^t \rangle$, so the objective in the optimization formulation is the same as the objective $\sum_{t=1}^{T} r_i(s_t, \boldsymbol{a}_t)$ in the RL formulation. The budget constraint is guaranteed by ignoring the bids exceeding agents' budgets in the environment. Therefore, BCB corresponds to the default setting of LSA.

2.2.2 Constrained Sparse Bidding

We propose Constrained Sparse Bidding (CSB) based on the real-world scenario Target CPA (Cost Per Action)[4] with some simplifications for understanding. The CPA of agent i is defined as $\mathrm{cpa}_i = \frac{\sum_{t=1}^{T} \langle \boldsymbol{x}_i^t, \boldsymbol{c}^t \rangle}{\sum_{t=1}^{T} \langle \boldsymbol{x}_i^t, \boldsymbol{v}_i^t \rangle}$, which can be seen as the cost taken by agent i for unit value. A low CPA means the budgets are consumed to obtain values effectively. Based on BCB, CSB adds one more constraint on CPA that cpa_i should be lower than the desired CPA d_i. The formulation is as follows:

$$\begin{aligned} &\max \sum_{t=1}^{T} \left\langle \boldsymbol{x}_i^t, \boldsymbol{v}_i^t \right\rangle \\ &\text{s. t.} \sum_{t=1}^{T} \left\langle \boldsymbol{x}_i^t, \boldsymbol{c}^t \right\rangle \leq b_i \\ &\quad \mathrm{cpa}_i \leq d_i. \end{aligned} \tag{2}$$

Given that CPA can only be calculated at the end of one episode, the environment will only provide a sparse reward in CSB, which is different from BCB. Unlike the budget constraint which cannot be violated in the environment, we allow agents to violate the CPA constraint, but we will penalize those agents for violations on their obtained values based on their CPA cpa_i. The sparse reward formulation in CSB is as follows:

$$r_i^{\mathrm{CSB}} = p(\mathrm{cpa}_i; d_i) \sum_{t=1}^{T} \left\langle \boldsymbol{x}_i^t, \boldsymbol{v}_i^t \right\rangle, \tag{3}$$

where $p(\mathrm{cpa}_i; d_i) = \min \left\{ \left(\frac{d_i}{\mathrm{cpa}_i} \right)^{\beta}, 1 \right\}$ is the penalty function for exceeding the CPA constraint. The formulation of $p(\mathrm{cpa}_i; d_i)$ implies that the penalty is incurred only when $\mathrm{cpa}_i > d_i$. The parameter $\beta > 0$ is typically set to 3. Therefore, CSB can be implemented with modifications to the reward function based on BCB.

2.3 Modules

The LSA Environment consists of three main modules: the impression generation module, the auction module, and the bidding module. The objective of the impression generation module is to generate impression features similar to real-world data with a generative model and value prediction model. The task of the auction module is to determine the winner and the winning price given all bids of agents for the ad impression opportunities. The bidding module is responsible for processing the multi-agent interactions between advertisers. The overview of the standard process in the LSA

[4] https://support.google.com/google-ads/answer/6268632

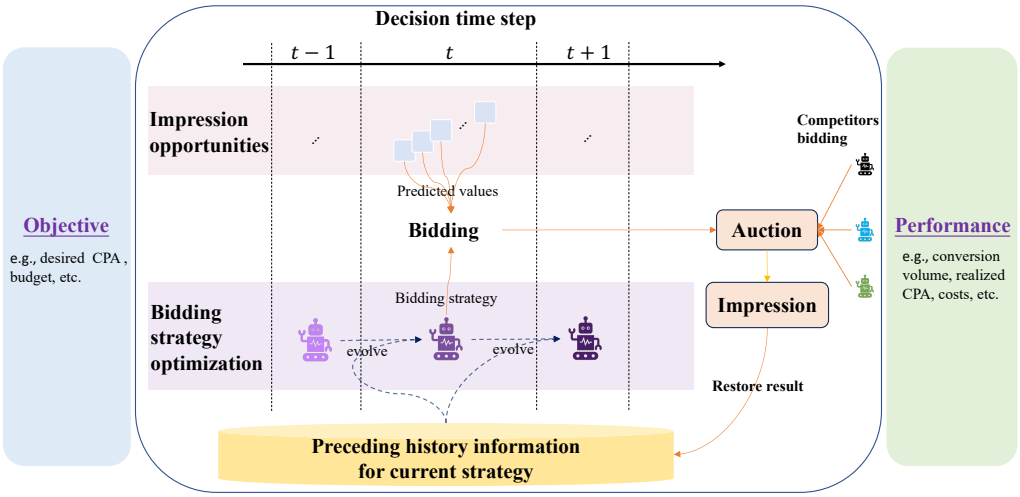

Figure 2: Overview of the standard process in the LSA Environment. The auto-bidding agent sequentially bids for impression opportunities at each step, evolving based on the preceding information.

Environment is included in Figure 2. Due to the space limit, please refer to Appendix E for more details about implementations of these modules.

2.4 Baseline Algorithms

We have implemented multiple baseline algorithms in LSA to facilitate a quick start-up and comprehensive understanding of users. The baseline algorithms include PID Controller[33], Online LP[11], IQL[17], Behavior Cloning[27], and Decision Transformer[6].

PID Controller. PID Controller is a traditional algorithm in the control field with a long history[3]. It is simple but effective in many scenarios. Recently, PID Controller has also been adopted in online advertising[33]. The idea of PID Controller is straightforward: PID Controller takes three parameters λ_P, λ_I, and λ_D for Proportional Control, Integral Control, and Derivative Control, respectively. We use the PID Controller to control the cost or bids of agents in this baseline.

Online LP. The optimization formulation (1) is a typical Linear Programming (LP) problem. Moreover, the variable $x_i^{t,j} \in \{0,1\}$ is binary, so the problem in each time step can be converted to a dynamic knapsack problem. Online LP solves this dynamic knapsack problem using a greedy algorithm.

IQL. Implicit Q-learning (IQL) is an offline RL algorithm. The idea of IQL is evaluating offline Q-function only on the actions that appeared in the offline data, to avoid the overestimation in the out-of-distribution data. In practice, IQL utilizes expectile regression to realize the offline Q-learning on in-distribution data.

Behavior Cloning. Behavior Cloning (BC) is a supervised learning algorithm given expert trajectories. The agent's policy learns by predicting the expert's action in the state of given trajectories. BC is a baseline for verifying the effectiveness of RL algorithms.

Decision Transformer. Decision Transformer (DT) utilizes the ability of Transformer[28] for sequential decision-making. DT views the trajectories in MDP as a sequence and predicts actions given previous transitions.

2.5 Environment API

The code of the LSA benchmark is implemented in Python. The environment API of LSA is similar to OpenAI Gym[4], so the construction and interactions of the LSA Environment may be familiar to related researchers. We included an example code as follows:

```python
from LSA import ModelPreRoundController
import numpy as np
# init environment
env = ModelPreRoundController(env_arg)
# init agents
agents = []
for i in range(num_agents):
    agents.append(Agent(agent_arg))
# start an episode
rewards =  np.zeros(shape=(num_agents))
costs =  np.zeros(shape=(num_agents))
obs = env.reset()
for tick_index in range(num_tick):
    bids = []
    for agent in agents:
        bid = agent.action(obs)
        bids.append(bid)
    _, pv_values = obs
    obs, reward, cost, info = envs.step(pv_values, bids)
    rewards += reward
    costs += cost
```

3 The LSA Dataset

The LSA Dataset is derived from advertising data generated via the auction system where massive agents compete against each other. This data can be used to model the auction environment and train the auto-bidding agent. The LSA Dataset includes 7 advertising episodes, each containing more than 1,400,000 impression opportunities and divided into 128 steps. Each opportunity includes the top **50** agents[5] with the highest bids. The dataset comprises over **500 million** records, totaling **40 GB** in size. Each record includes information such as the predicted conversion value, bid, auction, and impression results, among other details. The specific data format and data samples of the LSA Dataset are included in Appendix C.

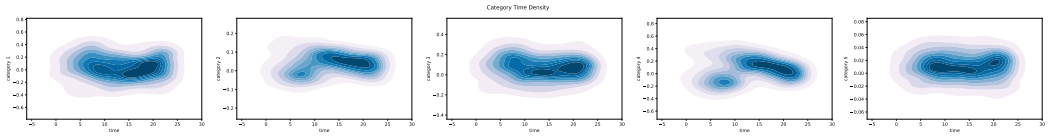

Figure 3: The joint value distribution between different categories and time in the LSA Dataset.

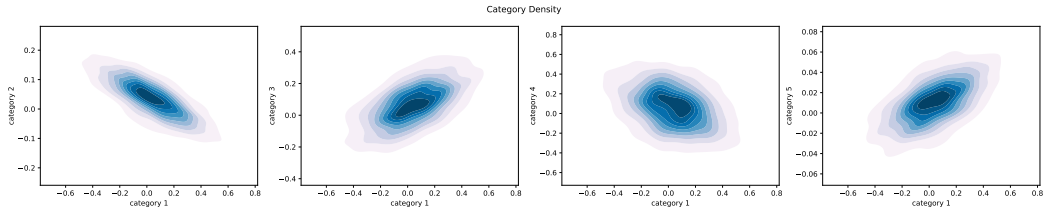

Figure 4: The joint value distribution between Category 1 and other categories in the LSA Dataset.

We have done some analysis of the LSA Dataset to provide some insights. We first investigate the variation of the impression values over time in one day. We selected five categories that appeared in the LSA Dataset. We denote them as Category 1, Category 2, and so on. We can find in Figure 3 that the impression values of different categories exhibit distinct patterns of variation. With the budget constraint, agents should consider the impression value variation over time to bid for appropriate impressions at the appropriate time. Furthermore, we study the relations between the

[5]Real-world data show that 50 agents can ensure competitive pressure for auto-bidding agent training.

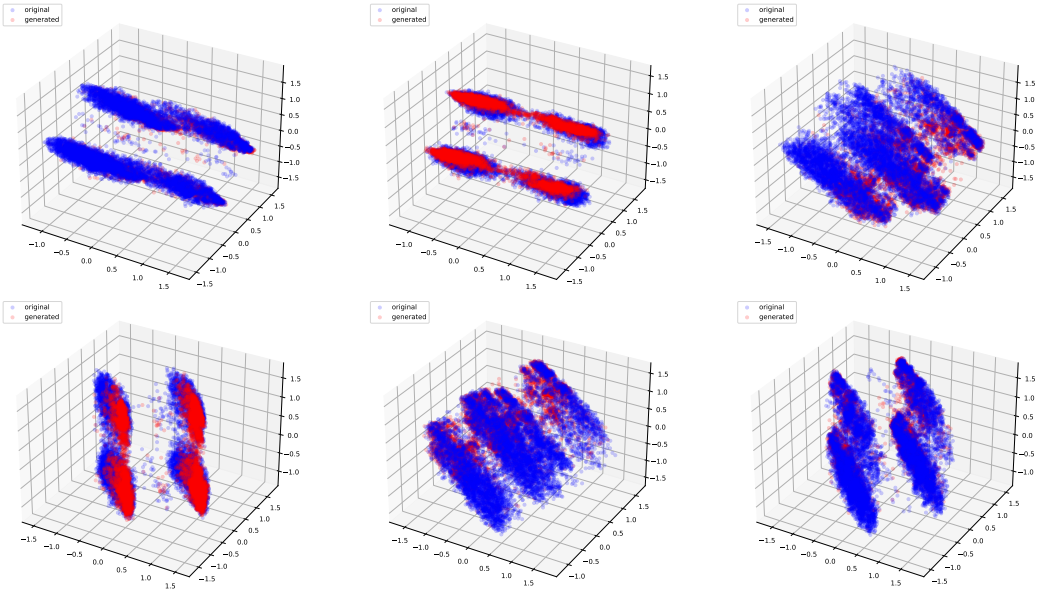

Figure 5: The 3D PCA results of 100K generated data and 100K real-world data.

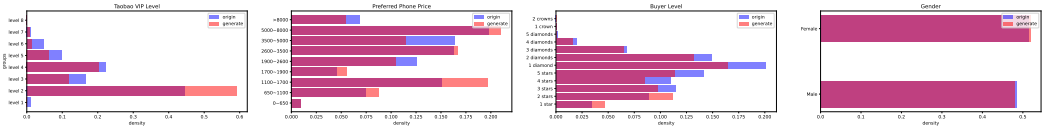

Figure 6: The distribution of identity information including the Taobao VIP level, the preferred phone price, the buyer level, and the gender in 100K generated data and 100K real-world data.

values of different categories. The relations between Category 1 and other categories are illustrated in Figure 4. The impression values of Category 1 and Category 3 are positively correlated, which means the corresponding advertisers are competitors for similar impression opportunities. Therefore, considering the preferences of other agents may be helpful for better bidding strategies.

The full datasheet of the LSA Dataset is included in Appendix B.

4 Experiments

4.1 Verification of The Impression Generation Module

The impression generation module generates data for environment interactions and we need a realistic simulation environment. Therefore, an important question is whether the generated data can reflect the properties of the real-world data. The impression generation module comprises two components: a feature generation model and a value prediction model. We have conducted experiments to verify the effectiveness of the two models.

4.1.1 Feature Generation

We randomly sample 100K real-world online advertising data to compare with the 100K generated data. The details of the generated data can be found in Appendix D. First, we perform PCA[15] to visualize the similarity between real-world and generated data. The 3D PCA results are illustrated in Figure 5. For better presentations, we use six different views in the 3D space. We can find that the generated data overlaps with the original data in the 3D space. Moreover, generated data points have four main separate parts in the 3D space, similar to real-world data points. These visualization results show that generated data resembles real-world data in general.

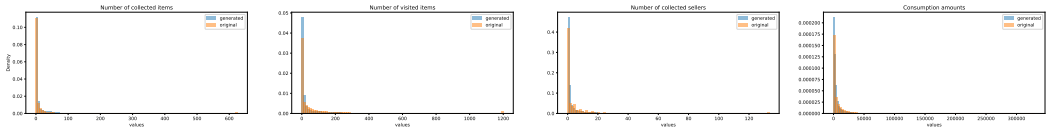

Figure 7: The distribution of consumption behavior information including the number of collected items, the number of visited items, the number of collected sellers, and the consumption amounts in 100K generated data and 100K real-world data.

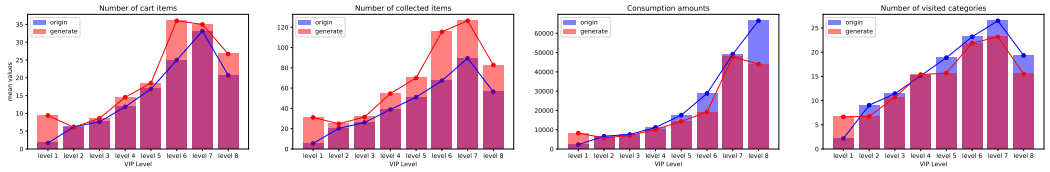

Figure 8: The mean values of consumption behavior information including the number of cart items, the number of collected items, the consumption amounts, and the number of visited categories in different VIP levels in 100K generated data and 100K real-world data.

To further compare these two datasets, we study the value distribution of identity information and consumption behavior information in two datasets, and the empirical results are included in Figure 6 and Figure 7. The feature vector contains 20+ fields as we described in Appendix D, so we only select a part of these fields for our experiments. As for identity information, the generated value distributions are similar to the real-world value distributions on the whole, while biases exist on some terms such as the 'level 7' for the Taobao VIP Level. The distributions with more choices are more difficult to match, and the gender distributions are almost the same in the two datasets. As for the consumption behavior information, we can find that the distributions of the two datasets in the selected fields share a great resemblance and are all long-tail distributions. The long-tail distribution means most customers don't consume often and customers with a high volume of consumption behavior are rare. This phenomenon matches up with our experiences in online advertising.

We investigate whether the generated data can capture the connections between different fields. We select the connection between the Taobao VIP level and the consumption behavior from the observation that customers with higher VIP levels usually have a higher volume of consumption behavior. We select four consumption behavior fields. The mean values of these fields in different VIP levels are shown in Figure 8. We find the overall monotonically increasing trend is captured by the generated data while biases exist in the specific values. Moreover, the drop of values from 'level 7' to 'level 8' is also captured by the generated data in three out of four fields except for the consumption amount. The rarity of 'level 8' data points may be the reason that the generative model is unable to distinct different trends for different fields.

4.1.2 Value Prediction

In real-world online advertising, the metrics for bidding strategy evaluation are Click-Through Rate (CTR) and Conversion Rate (CVR) and the bidding strategies make decisions based on the

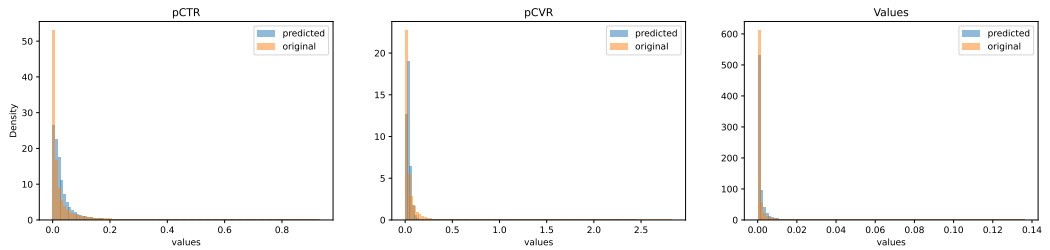

Figure 9: The distributions of the predicted pCTR, pCVR, and value compared with the ground truth.

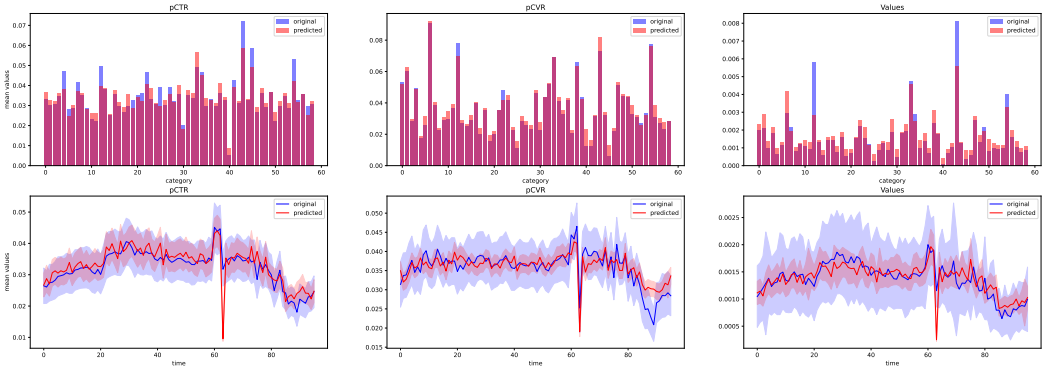

Figure 10: The means of the predicted pCTR, pCVR, and value in different categories and time steps compared with the ground truth. The shaded areas are related to the standard deviation.

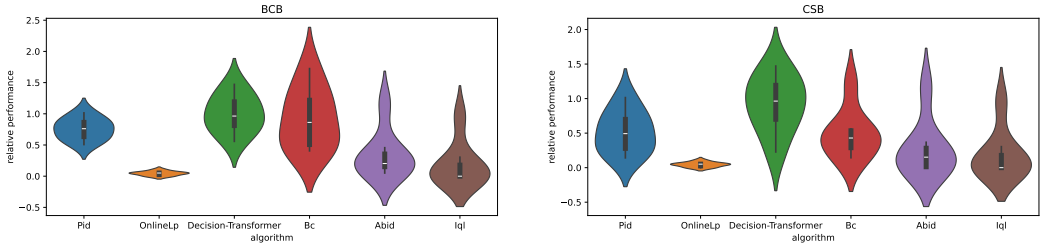

Figure 11: The empirical results of baseline algorithms on the two tasks including BCB and CSB in the LSA Environment.

pCTR and pCVR which are the estimated values of CTR and CVR respectively. For simplicity, in the LSA Environment, we assume that the estimation is accurate and define the value as $\text{value} = \text{pCTR} * \text{pCVR}$. Our value prediction model learns to predict pCTR and pCVR and calculates the value. We predict the pCTR, pCVR, and value of the 100K real-world data and compare the prediction with the real-world ground truth.

We first compare the overall distributions of predicted pCTR, pCVR, and value with the ground truth. The empirical results are illustrated in Figure 9. We find the two distributions are similar and the long-tail distributions arise again since the value corresponds to the consumption ability.

We hope the value prediction model can capture the value variation over the changes of category and time. The means of predicted pCTR, pCVR, and values in different categories and time steps compared with the ground truth are illustrated in Figure 10. The empirical results show that the variation trends over the changes in category and time of predictions are similar to the ground truth in general.

To present the results more intuitively, we provide some extra quantitative results. We compare the MSE between the generated and original distribution with the standard deviation of the original distribution. The quantitative results are shown in Table 1. It can be found that the MSEs are all smaller than the original_stds which means our prediction model can capture the patterns of the value variation and is accurate.

Table 1: The comparison of the MSE between the generated and original distribution with the standard deviation of the original distribution.

	original_std	MSE
pCVR_category	0.0685	**0.0341**
pCTR_category	0.0517	**0.0280**
value_category	0.00573	**0.00496**
pCVR_time	0.0637	**0.0313**
pCTR_time	0.0590	**0.0259**
value_time	0.00625	**0.00176**

4.2 Evaluations of Baseline Algorithms

In this section, we evaluate the baseline algorithms including PID[33], Online LP[11], IQL[17], BC[27], and DT[6] in the LSA Environment. We have introduced these baselines in Section 2.4. To

better illustrate the performances, we add a heuristic method, Abid, to the experiments. Abid means the agent will give a fixed bid rate for all impressions. Its performance can be seen as a reference in comparison. More details of the evaluation can be found in Appendix A.

The empirical results are included in Figure 11. For better illustration, we normalize the performances of all the baselines by the value of best mean episode rewards among the baselines. Therefore, the mean relative performance of Decision-Transformer is 1.0 in the BCB task. In both tasks, we can see that OnlineLp performs poorly with a low cumulative reward as it did not successfully bid for some impression opportunities in the auction process. The performance of IQL has been troubled by variance issues, and in some experiments, IQL struggles to secure high-quality impression opportunities. The excellent performance of the Decision Transformer in experiments highlights the potential of learning-based approaches compared to traditional methods. Furthermore, the reward drops of all baselines, especially BC and Pid, in the CSB tasks are caused by the CPA penalty in (3) for exceeding the constraint.

5 Related Work

Simulation Environment has been widely applied in RL research and successfully promoted the developments of related studies [5, 20, 29, 25, 26]. However, simulation environments for real online advertising systems are relatively rare in the auto-bidding field. AuctionGym [14] models the bidding problem as a contextual bandit problem [1], where the advertiser decides the bidding value given the information of the impression opportunity as a context. The context bandit has only one time step for an episode, which means AuctionGym doesn't consider the budget in the automated bidding. Moreover, AuctionGym describes the auto-bidding problem from the perspective of single-agent RL and ignores the influence of other agents. AdCraft [8] is a simulation environment for the bidding problem in Search Engine Marketing (SEM). Though AdCraft explicitly models the influences of other agents, these agents' policies are sampled from some parameterized distributions, which cannot reflect the multi-agent property essence of this problem. Despite the discussion above, these existing simulation environments lack a data-driven method for modeling the real online advertising system.

6 Conclusion and Limitations

We provide LSA, a novel large-scale game-based simulation environment derived from auto-bidding, and the corresponding dataset for environment interactions and offline training. We introduce the deep generative model to reduce the gap between the simulation environment and the online environment while avoiding the cost and risks of online experiments. We evaluate different types of algorithms in LSA and verify the similarity between the generated data and the real-world data. Though the generated data and the real-world data are similar in general, there are biases in some details and the performance of the generative model can be improved. Moreover, the LSA Environment is implemented in Python, which makes it computationally slower than environments that utilize highly optimized game engines in C++.

Acknowledgments

This work was supported in parts by NSFC under grants 62450001 and 62476008 and Alibaba Group through Alibaba Innovative Research Program. The authors would like to thank the anonymous reviewers for their valuable comments and advice.

References

[1] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

[2] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 2019.

[3] S. Bennett. Development of the pid controller. *IEEE Control Systems Magazine*, 13(6):58–62, 1993.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

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

[7] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[8] Maziar Gomrokchi, Owen Levin, Jeffrey Roach, and Jonah White. Adcraft: An advanced reinforcement learning benchmark environment for search engine marketing optimization. *arXiv preprint arXiv:2306.11971*, 2023.

[9] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503, 2023.

[10] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

[11] Xiaotian Hao, Zhaoqing Peng, Yi Ma, Guan Wang, Junqi Jin, Jianye Hao, Shan Chen, Rongquan Bai, Mingzhou Xie, Miao Xu, et al. Dynamic knapsack optimization towards efficient multi-channel sequential advertising. In *International Conference on Machine Learning*, pages 4060–4070. PMLR, 2020.

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

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[14] Olivier Jeunen, Sean Murphy, and Ben Allison. Off-policy learning-to-bid with auctiongym. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4219–4228, 2023.

[15] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *Deep RL Workshop NeurIPS 2021*, 2021.

[18] Vaibhav Kulkarni, Natasa Tagasovska, Thibault Vatter, and Benoit Garbinato. Generative models for simulating mobility trajectories. *arXiv preprint arXiv:1811.12801*, 2018.

[19] Skyler Norgaard, Ramyar Saeedi, Keyvan Sasani, and Assefaw H Gebremedhin. Synthetic sensor data generation for health applications: A supervised deep learning approach. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1164–1167. IEEE, 2018.

[20] C Berner OpenAI, Greg Brockman, Brooke Chan, Vicki Cheung, P Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[21] Weitong Ou, Bo Chen, Xinyi Dai, Weinan Zhang, Weiwen Liu, Ruiming Tang, and Yong Yu. A survey on bid optimization in real-time bidding display advertising. *ACM Transactions on Knowledge Discovery from Data*, 18(3):1–31, 2023.

[22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[25] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[26] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[27] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2018.

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

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

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

[31] Yuxin Wu, Tianyang Zhao, Haoyuan Yan, Min Liu, and Nian Liu. Hierarchical hybrid multi-agent deep reinforcement learning for peer-to-peer energy trading among multiple heterogeneous microgrids. *IEEE Transactions on Smart Grid*, 2023.

[32] Wei Zhang, Yanjun Han, Zhengyuan Zhou, Aaron Flores, and Tsachy Weissman. Leveraging the hints: Adaptive bidding in repeated first-price auctions. *Advances in Neural Information Processing Systems*, 35:21329–21341, 2022.

[33] Weinan Zhang, Yifei Rong, Jun Wang, Tianchi Zhu, and Xiaofan Wang. Feedback control of real-time display advertising. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 407–416, 2016.

[34] Yiheng Zhu, Yang Zhan, Xuankun Huang, Yuwei Chen, Jiangwen Wei, Wei Feng, Yinzhi Zhou, Haoyuan Hu, Jieping Ye, et al. Ofcourse: A multi-agent reinforcement learning environment for order fulfillment. *Advances in Neural Information Processing Systems*, 36, 2024.

A Evaluation Details

There are 48 agents of 7 types in our experiments and each type corresponds to one algorithm. We test 7 rounds where we permute the order of agents in each round. Therefore, agents will represent different advertisers with different budgets in different rounds. We choose the best agent as the representative of an algorithm if there are multiple agents of this algorithm. We use the average performances of the 7 rounds as the final performance of all the algorithms. We provide the model file of these agents and the evaluation code for reproduction.

B Datasheet for LSA

We present a datasheet[7] for the LSA Dataset.

B.1 Motivation

For what purpose was the dataset created?

In general, learning from interactions with the real-world online advertising system is difficult and expensive, so offline RL algorithms are more popular in auto-bidding. Therefore, we build the Auction Dataset to facilitate offline training of users. Moreover, the Auction Dataset will also be provided to the participants of the competition we will hold in the future.

Who created the dataset?

The dataset was created by the authors of this paper. The dataset was not created on the behalf of any entity.

Who funded the creation of the dataset?

Alibaba Group funds the creation of the LSA Dataset.

B.2 Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)?

The LSA Dataset contains trajectories of diverse agents competing with each other. Please refer to Appendix C and Section 3 for more details.

Is there a label or target associated with each instance?

The LSA Dataset contains offline trajectories where the actions or bids of agents can be seen as labels for the time step.

Is any information missing from individual instances?

Not to our knowledge.

Are there recommended data splits (e.g., training, development/validation, testing)?

No.

Are there any errors, sources of noise, or redundancies in the dataset?

The LSA Dataset contains trajectories of diverse agents, some of these agents may not perform well. However, the tasks in the LSA Environment are still difficult for some algorithms and we think keeping agents diverse in the LSA Dataset is beneficial.

Do/did we do any data cleaning on the dataset?

We did not. All data is presented exactly as collected.

B.3 Collection Process

How was the data associated with each instance acquired?

The LSA Dataset is collected from the interactions of baseline agents in the LSA Environment.

Who was involved in the data collection process and how were they compensated?

The data collection process is done by the authors and not involve with any crowdsource.

Over what timeframe was the data collected?

The LSA Dataset was collected between March 2024 and May 2024.

B.4 Uses

Has the dataset been used for any tasks already?

No.

Is there a repository that links to any or all papers or systems that use the dataset?

No.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?

We do not believe so since the LSA Dataset consists of data generated by the interactions of baseline agents.

B.5 Distribution

Will the dataset be distributed to third parties?

Yes, but the LSA Dataset and environment are involved with a large competition we will hold in NeurIPS 2024, so we will not distribute them until the end of the competition considering competition fairness. However, we will open-source the LSA Dataset as soon as possible.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)? Does the dataset have a digital object identifier (DOI)?

The LSA Dataset will be distributed by a Github link after the end of the competition we will hold. The LSA Dataset doesn't have a digital object identifier now.

All data is under the MIT license.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

No.

B.6 Maintenance

Who will be supporting/hosting/maintaining the dataset?

The authors of this paper will provide needed maintenance to the datasets.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

Please email us at huoyusen.huoyusen@alibaba-inc.com.

Is there an erratum?

There is not and we believe generated features, predicted values, and trajectories in our datasets do not involve an erratum.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?

Yes, but as we won't add extra data points, the update will be minimal.

C Data Format of LSA Dataset

The specific data format of the LSA Dataset is as follows:

(c1). deliveryPeriodIndex: The index of the current delivery period.

(c2). advertiserIndex: The unique identifier of the advertiser.

(c3). advertiserCategoryIndex: The index of the advertiser's category.

(c4). budget: The advertiser's budget for a period.

(c5). CPAConstraint: The CPA constraint of the advertiser.

(c6). timeStepIndex: The index of the current decision time step.

(c7). remainingBudget: The advertiser's remaining budget before the current step.

(c8). pvIndex: The index of the impression opportunity.

(c9). pValue: The conversion action probability when the advertisement is exposed to the customer.

(c10). pValueSigma: The variance of predicted probability.

(c11). bid: The agent's bid for the impression opportunity.

(c12). xi: The winning status of the agent for the impression opportunity.

(c13). adSlot: The won ad slot.

(c14). cost: The cost needs to be paid if the ad is exposed to the customer.

(c15). isExposed: The indicator signifying whether the ad in the slot was displayed to the customer.

(c16). conversionAction: The indicator signifying whether the conversion action has occurred.

(c17). leastWinningCost: The minimum cost to win the impression opportunity.

(c18). isEnd: The completion status of the advertising period.

Table 2 presents an impression opportunity involving the top five advertisers. The top three advertisers, numbered 31, 22, and 15, won the impression opportunity with the highest bids and were allocated to ad slots 1, 2, and 3, respectively. During this impression, slots 1 and 2 were exposed to the customer, while slot 3 remained unexposed. Consequently, ads in slots 1 and 2 need to pay 0.2702 and 0.2154, respectively. Additionally, the customer engaged in a conversion action with the ad in slot 2.

Table 2: Bidding, auction, and impression processes for each advertiser during the same opportunity.

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18
1	31	2	6500.00	27.00	5	5962.49	101000	0.0103542	0.0021549	0.2845	1	1	0.2702	1	0	0.1832	0
1	22	6	7000.00	38.00	5	5988.25	101000	0.0070297	0.0005213	0.2702	1	2	0.2154	1	1	0.1832	0
1	15	7	7000.00	42.00	5	6132.52	101000	0.0051392	0.0004312	0.2154	1	3	0.1832	0	0	0.1832	0
1	39	3	6000.00	30.00	5	5443.27	101000	0.0062134	0.0007254	0.1832	0	0	0	0	0	0.1832	0
1	43	9	7500.00	25.00	5	6421.81	101000	0.0045392	0.0006215	0.1099	0	0	0	0	0	0.1832	0

Table 3 presents a data sample illustrating an advertiser's bidding process across time steps within a delivery period. The advertiser has a budget of 7500, a CPA constraint of 40, and belongs to industry category 6. Throughout different time steps, the advertiser engages in bidding for every available impression and obtains the corresponding results. During this period, the advertiser's remaining budget decreases correspondingly. Additionally, the advertiser adjusts their bidding strategy based on prior performance, although this adjustment will not be directly evident in the data.

D The Structure of Generated data

Structure of the feature vector. The feature vector consists of several types of information including one-hot vectors, integers and float numbers. The specific data format of the feature vector is as follows:

(c1). idAgeLevel: Represents the age level of the customer. The meanings of values: 0 for unknown, 1$\sim$8 for ages over 12, 18, 22, 25, 30, 35, 40 and 50 respectively. Data format: one-hot vector, dimension $[0, 9)$.

Table 3: An advertiser's bidding process across time steps.

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18
3	48	6	7500.00	40.00	1	7500.00	1	0.0032157	0.0003567	0.1345	0	0	0	0	0	0.1628	0
3	48	6	7500.00	40.00	1	7500.00	2	0.0146256	0.0021352	0.5852	0	0	0	0	0	0.6421	0
3	48	6	7500.00	40.00	1	7500.00	3	0.0054324	0.0007631	0.1924	1	1	0.1673	1	1	0.1454	0
3	48	6	7500.00	40.00	1	7500.00	4	0.0073145	0.0006529	0.2786	0	0	0	0	0	0.2862	0
...																	
3	48	6	7500.00	40.00	2	7341.25	20901	0.0076453	0.0006579	0.2856	0	0	0	0	0	0.3245	0
3	48	6	7500.00	40.00	2	7341.25	20902	0.0139234	0.0012358	0.5629	1	2	0	0	0	0.6782	0
3	48	6	7500.00	40.00	2	7341.25	20903	0.0077212	0.0006579	0.3045	0	0	0	0	0	0.3122	0
3	48	6	7500.00	40.00	2	7341.25	20904	0.0021341	0.0001873	0.0926	0	0	0	0	0	0.1151	0
...																	
3	48	6	7500.00	40.00	43	0.00	895201	0.0065274	0.0005689	0.0000	0	0	0	0	0	0.1243	1
3	48	6	7500.00	40.00	43	0.00	895202	0.0032125	0.0002986	0.0000	0	0	0	0	0	0.2986	1
3	48	6	7500.00	40.00	43	0.00	895203	0.0112986	0.0013253	0.0000	0	0	0	0	0	0.0932	1
3	48	6	7500.00	40.00	43	0.00	895204	0.0051678	0.0006782	0.0000	0	0	0	0	0	0.1687	1

(c2). idGender: Represents the gender of the customer. The meanings of values: 0,1 and 2 for unknown, Female and Male respectively. Data format: one-hot vector, dimension $[9, 12)$.

(c3). isForeign: Represents whether the customer is foreign. The meanings of values: 0,1 and 2 for unknown, No and Yes respectively. Data format: one-hot vector, dimension $[12, 15)$.

(c4). cityLevel: Represents the level of the city where the customer is living. The meanings of values: 0 for unknown, 1$\sim$6 for different city development levels in descending order. Data format: one-hot vector, dimension $[15, 22)$.

(c5). isCap: Represents whether the city the customer living in is the capital. The meanings of values: 0 for No and 1 for Yes. Data format: one-hot vector, dimension $[22, 24)$.

(c6). buyerStarName: Represents the rating of the customer as a buyer. The meanings of values: 0 for unknown, 1$\sim$5 for 1$\sim$5 stars respectively, 6$\sim$10 for 1$\sim$5 diamonds respectively, 11$\sim$15 for 1$\sim$5 crowns respectively, 16$\sim$20 for 1$\sim$5 golden crowns respectively, 21 for credit score ≤ 3, and 22 for credit score $= 0$. In general, the order of these values' ratings is $22 < 21 < 1 < 2 < \cdots < 20$. Data format: one-hot vector, dimension $[24, 47)$.

(c7). tmLevel: Represents the VIP level of the customer in Tmall. The meanings of values: 0 for unknown or no VIP level, 1$\sim$5 for VIP levels 1$\sim$5 respectively. Data format: one-hot vector, dimension $[47, 53)$.

(c8). vipLevelName: Represents the VIP level of the customer in Taobao. The meanings of values: 0 for unknown or no VIP level, 1$\sim$8 for VIP levels 1$\sim$8 respectively. Data format: one-hot vector, dimension $[53, 62)$.

(c9). phonePriceLevelPrefer: Represents the preferred phone price interval of the customer. The meanings of values: 0 for unknown, 1 for 0$\sim$650 CNY, 2 for 1100$\sim$1700 CNY, 3 for 1700$\sim$1900 CNY, 4 for 1900$\sim$2600 CNY, 5 for 2600$\sim$3500 CNY, 6 for 3500$\sim$5000 CNY, 7 for 5000$\sim$8000 CNY, 8 for 650$\sim$1100 CNY, 9 for higher than 8000 CNY. Data format: one-hot vector, dimension $[62, 72)$.

(c10). zipCode: Represents the zip code of the address where the customer is living. The meanings of values: The zip code contains 6 digits and each digit is a number from 0$\sim$9. We encode each digit with an one-hot vector and concatenate them together. Data format: one-hot vector, dimension $[72, 132)$.

(c11). idBirthyear: Represents the birthyear of the customer. Data format: integer, dimension $[132, 133)$.

(c12). nationId: Represents the nation of the customer. The meanings of values: 1 for China. (The real-world training data contains almost no data points from other countries. So does our generated data.) Data format: integer, dimension $[133, 134)$.

(c13). payOrdAmt: Represents the order amounts of the customer in the last one month, one year, three months and six months. Data format: float numbers, dimension $[134, 138)$.

(c14). payOrdCnt: Represents the customer's number of orders in the last one month, one year, three months and six months. Data format: integers, dimension $[138, 142)$.

(c15). payOrdDays: Represents the number of days when the customer placed orders in the last one month, one year, three months and six months. Data format: integers, dimension $[142, 146)$.

(c16). payOrdItmCnt: Represents the number of item types the customer bought in the last one month, one year, three months and six months. Data format: integers, dimension $[146, 150)$.

(c17). payOrdItmQty: Represents the number of items the customer bought in the last one month, one year, three months and six months. Data format: integers, dimension $[150, 154)$.

(c18). pvAndIpv: Represents the PV and IPV value of the customer bought in the last month. Data format: float numbers, dimension $[154, 156)$.

(c19). vstSlrCnt: Represents the number of sellers the customer visited in the last month. Data format: integers, dimension $[156, 157)$.

(c20). vstCateCnt: Represents the number of categories the customer visited in the last month. Data format: integers, dimension $[157, 158)$.

(c21). vstItmCnt: Represents the number of items the customer visited in the last month. Data format: integers, dimension $[158, 159)$.

(c22). vstItmCnt: Represents the number of items the customer visited in the last month. Data format: integers, dimension $[158, 159)$.

(c23). vstDays: Represents the number of days the customer visited items in the last month. Data format: integers, dimension $[159, 160)$.

(c24). stayTimeLen: Represents the number of seconds the customer spent on visiting items in the last month. Data format: integers, dimension $[160, 161)$.

(c25). cartItmCnt: Represents the number of items the customer added to the cart in the last one week, two weeks, one month, three months, and six months. Data format: integers, dimension $[161, 166)$.

(c26). cltSlrCnt: Represents the number of sellers the customer collected in the last one week, two weeks, one month, six months, and one year. Data format: integers, dimension $\{166, 168, 170, 172, 174\}$.

(c27). cltItmCnt: Represents the number of items the customer collected in the last one week, two weeks, one month, six months, and one year. Data format: integers, dimension $\{167, 169, 171, 173, 175\}$.

Structure of the value vector. The value vector has 60 dimensions corresponding to 59 categories involved in our environment and one conserved category for the undefined or unknown category. The corresponding relations between the dimension indexes and categories are listed in Table 4.

E Implementation and Modules

The environment of LSA consists of three main modules: the impression generation module, the auction module, and the bidding module. The general process of one time step in LSA can be concluded as follows:

1) The impression generation module generates features $\boldsymbol{f} = (f^1, f^2, \cdots, f^m)$ and values $\boldsymbol{v} = \{v_i^j\}$ of m ad impression opportunities for n agents, where the number of opportunities m is sampled from an intern distribution within LSA. This intern distribution is obtained from real-world online advertising statistics
2) Agents bid for all the ad impression opportunities considering the predicted values provided by the environment and the historical auction logs.
3) The auction module determines the winner of each auction, rewards, and costs by the auction mechanism.
4) Agents receive rewards, costs, and new auction logs. The budgets of all agents are updated according to auction results. In the next time step, all the processes above will be repeated.

Given this general process, we will introduce the three main modules in order. The impression generation module will generate features $\boldsymbol{f}$ of ad impressions related to the real online data. The auction module supports a hierarchical auction system similar to real-world online advertising and realizes several popular auction mechanisms for different research purposes. The bidding module supports explicitly modeling a multi-agent environment with several implemented baselines.

Table 4: Corresponding relations for categories.

ID	Category	ID	Category
1	Snacks	31	Travel Services
2	Personal Care	32	Tmall Home & Living
3	Electric Vehicles	33	Maternity & Childcare
4	Tmall Underwear	34	Movies, Shows & Sports
5	Smart Toys & Games	35	Education & Teaching
6	Tea	36	Taobao Bags & Accessories
7	Household Cleaning	37	Taobao Underwear
8	Chilled Food	38	Audio & Video Electronics
9	Tmall Women's Clothing	39	Gaming
10	Enterprise Services	40	Pets
11	Dairy Products	41	Vehicles
12	Fragrances and Aromatherapy	42	Major Appliances
13	Life Services	43	Tmall Footwear
14	Household Appliances	44	Food Coupons
15	Taobao Men's Clothing	45	Auto Accessories
16	Tmall Home Decor	46	Mobile Phones
17	Taobao Home & Living	47	Taobao Footwear
18	Taobao Home Decor	48	Grains & Instant Food
19	Instant Drinks	49	Tmall Bags & Accessories
20	Alcohol	50	Mobile & Digital Accessories
21	Taobao Women's Clothing	51	Taobao Watches & Glasses
22	Auto Aftermarket	52	Jewelry & Accessories
23	Fruits and Vegetables	53	Sports
24	Flowers and Gardening	54	Toys & Fun
25	Office & School Supplies	55	Entertainment Recharge
26	Computers	56	Beverages
27	Tmall Watches & Glasses	57	Outdoor
28	Computer Accessories	58	Motorcycles
29	Aquatic Products, Meat, Poultry & Eggs	59	Tmall Men's Clothing
30	Cosmetics	0	Other

E.1 Impression Generation Module

The target of the impression generation module is to generate diverse impression features similar to real online advertising data. The core of this module is the generative model. The objective of the generative model in LSA is to generate data resembling real advertising delivery data. Useful information in the real advertising delivery data can be divided into four parts: features of impression opportunities (customers' information), features of advertisers, time when the impression opportunity arises, and the values of the impression opportunities. In our model, we simplify the feature of advertisers to be the advertisers' industry categories. We focus on the generation of impression opportunity features and take the categories and time as conditions. The generative model consists of two components: the generative model for impression opportunity features and the prediction model for the values.

Feature Generation. The impression feature contains two parts of information: the basic identity information of customers and the consumption records such as the consumption amount. The identity information is discrete and the consumption records are continuous in general, which are processed with different measures. Diffusion [13] model is the most popular generative model recently which obtains SOTA performances in image generation with a simplified training process. We would like to adopt the diffusion model to generate the impression feature but struggle with the denoising operation which can result in unreasonable outputs such as a negative consumption amount. So we follow the idea of the Latent Diffusion Model (LDM)[23] to generate impression features. LDM adds noises and denoises in the latent space with a diffusion model and generates data from the latent space with an encoder and decoder. More details can be found in Appendix F.1.

Value Prediction. The value prediction model needs to handle three types of information: impression features, the industry category information of advertisers, and time information. We simplify the category and time information as discrete values. Therefore, we aim to integrate the category and time information into the impression features for better value prediction. Besides, we hope this integration can partly reflect the variation pattern of the impression values related to advertisers'

features and time. Multi-head attention (MHA), as a popular network architecture and the critical part of Transformer [28], can capture the relations among a sequence, thus we hope to utilize MHA for better integration. We combine cross-attention and self-attention to integrate the three types of information. We also follow the idea of position embedding in the Transformer to process the time information. More details are included in Appendix F.2.

For the consideration of interaction efficiency in LSA, the environment utilizes a dataset consisting of generated features and corresponding predicted values. More details of the dataset will be discussed in Section 4.1. Though the impression generation module is trained with real online advertising data, an important question is whether the generated data can reflect the properties of real data. Therefore, we have done several related experiments and the empirical results will also be discussed in Section4.1.

E.2 Auction Module

The task of the auction module is to determine the winner and the winning price given all bids of agents for the ad impression opportunities. The costs of agents will change given different auction rules. The most commonly discussed auction rule is the Generalized Second-Price (GSP) Auction which means the winner should pay a cost slightly higher than the second-highest bid instead of the highest bid. The auction module internally supports several popular auction rules including GSP for the convenience of researchers. Besides, researchers can also design a specific auction rule related to their purposes with the interface of the auction module.

Additionally, the property of multiple slots has been implemented in our simulation platform. Multiple slots emerge from the application in the industry, which means one impression opportunity has multiple ad slots for ad displays. The ad slots are ranked by their exposure rates. A higher exposure rate slot is more valuable for advertisers. Suppose the number of slots is l, then the auction module will attribute l slots to the top l bidders and these bidders will receive different values according to different exposure rates of slots. In the LSA Environment, l is set to 3. Let k_i^j represent the slot of impression opportunity j wined by agent i and $e_i^j \in [0, 1]$ represent the exposure rate of slot k_i^j, then the optimization formulation of BCB with multiple slots is as follows:

$$\begin{aligned} &\max \sum_{t=1}^{T} \sum_{j=1}^{m} e_i^{t,j} x_i^{t,j} v_i^{t,j} \\ &\text{s. t.} \sum_{t=1}^{T} \sum_{j=1}^{m} e_i^{t,j} x_i^{t,j} c^{t,j} \leq b_i, \end{aligned} \tag{4}$$

In summary, the multiple slots property increases the complexity of the optimal bidding strategy, since the exposure rate is a discount factor for both the cost and values. For instance, a strategy using a lower budget to bid for slots with relatively lower ranks may be better than the strategy that always chases the highest value slot. We believe supporting multiple slots in LSA will be beneficial to reducing the gap between related research and the real-world online advertising system.

E.3 Bidding Module

The bidding module is responsible for processing the multi-agent interactions between advertisers. This module implements the budget constraint and models the auto-bidding problem with sequence decision-making. Therefore, LSA supports the mainstream paradigms including Budget Constrained Bidding (BCB) [30] and Multiple Constraints Bidding (MCB) [12] in the auto-bidding field. This will help researchers validate and gain insights from existing algorithms.

In the bidding module, we explicitly model the multi-agent setting. Researchers can implement multi-agent algorithms to achieve competition or cooperation among different agents. The varying bidding strategies of other agents can better reflect the complex and dynamic auction environment in real-world online advertising systems. Besides, researchers can only control a part of the agents in LSA while others is uncontrollable. This scenario is closer to the real advertising platform. The multi-agent setting of LSA can adapt to different research objectives.

There are several different metrics for different business goals of advertisers in online advertising systems such as Return-on-Investment (ROI) and Return-On-Ad-Spend (ROAS). LSA has several built-in metrics covering the popular metrics used by the major advertising platform. Researchers can adopt these metrics conveniently to evaluate the performances of their auto-bidding strategies. Besides, researchers can define customized metrics according to their research objectives. Additionally, several popular algorithms in the auto-bidding field have been implemented as baselines in LSA, including PID Controller[33], Online LP[11], IQL[17], Behavior Cloning[27], and Decision Transformer[6]. This can facilitate the interested researchers to quickly start up and evaluate these baselines in a unified environment.

F Details of Generative Model

F.1 Impression Feature Generation

The impression feature contains two parts of information: one is the basic identity information of customers including gender, age, address and so on; another is the consumption records such as the consumption amount and the number of orders in the last three months. The identity information consists of discrete fields and each field has several candidates. The consumption records are continuous in general. Therefore, we process these two types of information with different measures.

Diffusion [13] model is the most popular generative model recently which obtains SOTA performances in image generation with a simplified training process. The principle of the diffusion model is adding Gaussian noises to original data in training and denoising from Gaussian noises in the generation process. We would like to adopt the diffusion model to generate the impression feature but struggle with the denoising operation which can result in unreasonable outputs such as a negative consumption amount. So we follow the idea of the Latent Diffusion Model (LDM)[23]. LDM has a latent space to encode the original data. LDM combines the idea of diffusion model with VAE[16]. LDM adds noises and denoises in the latent space with a diffusion model and generates data from the latent space with an encoder and decoder.

Specifically, let $X \subset \mathbb{R}^d$ be the space of impression feature data $(x_1, x_2, \cdots, x_N)$ where d is the dimension of original data and N is the volume of the impression feature dataset. Let $Y \subset \mathbb{R}^l$ be the latent space. The encoder and decoder are represented as g_ϕ and h_ψ respectively, where ϕ and ψ are the parameters. The function of the encoder g_ϕ is obtaining a latent representation of original data as follows:

$$g_\phi(x_i) = (\mu_i, \sigma_i), \quad y_i \sim \mathcal{N}(\mu_i, \sigma_i^2),$$

where $y_i \in Y$ is the latent representation. In practice, the reparameterize trick [16] is applied to make sure this operation is differentiable in the backpropagation. Given the latent representation y_i, the decoder is responsible for reconstructing the original data from y_i, *i.e.*, $h_\psi(y_i) = \tilde{x}_i \in X$. Besides the reconstruction, the latent distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ is expected to be close to the standard Gaussian distribution $\mathcal{N}(0, 1)$. Therefore, we have the following loss function for the encoder and decoder:

$$\mathcal{L}_{recons} = \frac{1}{N}\sum_{i=1}^{N} \|x_i - h_\psi(y_i)\|_2^2, \quad \mathcal{L}_{reg} = \frac{1}{N}\sum_{i=1}^{N} D_{\mathrm{KL}}\left(\mathcal{N}(\mu_i, \sigma_i^2) \| \mathcal{N}(0, 1)\right),$$

where $\mathcal{L}_{recons}$ is the reconstruction loss and $\mathcal{L}_{reg}$ is the regularization loss for the latent distribution.

Different from the original idea of VAE, where the latent variable $y \in Y$ is sampled from $\mathcal{N}(0, 1)$ in the generation process, LDM uses a diffusion model in the latent space to generate the latent variable. In general, the idea of the diffusion model is adding Gaussian noises to the original data to obtain variables in $\mathcal{N}(0, 1)$ and denoising from $\mathcal{N}(0, 1)$ for generation. Given a latent variable y, we denote the noisy version of y after t iterations as y^t. The diffusion model has a network to predict noise $\epsilon_\theta(y^t, t)$ and the loss function can be represented as

$$\mathcal{L}_{LDM} = \frac{1}{N}\sum_{i=1}^{N} \|\epsilon - \epsilon_\theta(y_i^{t_i}, t_i)\|_2^2,$$

where $\epsilon \sim \mathcal{N}(0, 1)$, y_i is the latent embedding of x_i and t_i is uniformly sampled from the set $\{1, 2, \cdots, t_{\max}\}$. $\epsilon_\theta(y^t, t)$ is the only learnable network in the diffusion model, with which the process of adding noises and denoising can be completed by the basic operations.

As for the generation process, a latent variable $\tilde{y}$ is sampled from $\mathcal{N}(0,1)$ and y is obtained by $t_{\max}$ denoising steps from $\tilde{y}$ given the noise prediction network ϵ_θ. Finally, the decoder generates an impression feature based on y as $x = h_\psi(y)$.

F.2 Value Prediction

The value prediction model needs to handle three types of information: impression features, category information and time information. The category information corresponds to the industry categories of advertisers and the time information corresponds to the time when the impression opportunity arrived. In our model, the category and time information are simplified as discrete values. Therefore, we aim to integrate the category and time information into the impression features for better value prediction. Besides, we hope this integration can partly reflect the variation pattern of the impression values related to advertisers' features and time.

Multi-head attention (MHA) [28] is a popular network architecture and the critical part of Transformer [28]. MHA can capture the relations among a sequence, thus we hope to utilize MHA for better integration. The formulation of the attention network is straightforward as $\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}}) \cdot V$. Multi-head attention can be viewed as applying the attention network in different representation subspaces as follows:

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \mathrm{head}_2, \cdots, \mathrm{head}_h)W^O, \tag{5}$$

$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{6}$$

W_i^Q, W_i^K, W_i^V are the parameters for the projection networks of head i and W^O is the output network parameters of the MHA model.

We combine cross-attention and self-attention to integrate the three types of information. Suppose x_i, x_i^{time} and x_i^{cate} are the impression feature, time information and category information in one record respectively, then we will process the information as follows:

$$\begin{aligned}
&Q^{(1)} = \tau_Q^{(1)}(x_i^{\mathrm{time}}), \quad K^{(1)} = \tau_K^{(1)}(x_i), \quad V^{(1)} = \tau_V^{(1)}(x_i),\\
&z_i^{(1)} = \mathrm{MultiHead}(Q^{(1)}, K^{(1)}, V^{(1)}),\\
&Q^{(2)} = \tau_Q^{(2)}(z_i^{(1)}), \quad K^{(2)} = \tau_K^{(2)}(z_i^{(1)}), \quad V^{(2)} = \tau_V^{(2)}(z_i^{(1)}),\\
&z_i^{(2)} = \mathrm{MultiHead}(Q^{(2)}, K^{(2)}, V^{(2)}),\\
&Q^{(3)} = \tau_Q^{(3)}(x_i^{\mathrm{cate}}), \quad K^{(3)} = \tau_K^{(3)}(z_i^{(2)}), \quad V^{(3)} = \tau_V^{(3)}(z_i^{(2)}),\\
&z_i^{(3)} = \mathrm{MultiHead}(Q^{(3)}, K^{(3)}, V^{(3)}),\\
&Q^{(4)} = \tau_Q^{(4)}(z_i^{(3)}), \quad K^{(4)} = \tau_K^{(4)}(z_i^{(3)}), \quad V^{(4)} = \tau_V^{(4)}(z_i^{(3)}),\\
&z_i = \mathrm{MultiHead}(Q^{(4)}, K^{(4)}, V^{(4)}),
\end{aligned}$$

where $\tau^{(1)}, \tau^{(2)}, \tau^{(3)}, \tau^{(4)}$ are the projection function for the multi-head attention network.

The variation of impression opportunity values has some temporal patterns in the real world. Therefore, we follow the position encoding idea in Transformer [28] and Diffusion Model [13] to process the time information. Let $\mathrm{PE} : \mathbb{N} \to \mathbb{R}^d$ represent the position encoding function, then

$$\mathrm{PE}_{2i}(t) = \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right), \quad \mathrm{PE}_{2i+1}(t) = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right),$$

where t is the discrete time and i corresponds to the dimension in the embedding $\mathrm{PE}(t)$. Let $e_i = \mathrm{PE}(x_i^{time})$, then the value prediction is conducted by $\hat{v}_i = U_\xi(z_i, e_i)$, where $U_\xi(z_i, e_i)$ is the prediction network with a similar architecture to the U-Net [24] used by the Diffusion Model [13]. The loss of the value prediction model is shown below:

$$\mathcal{L}_{\mathrm{pred}} = \frac{1}{N}\sum_{i=1}^{N} \|v_i - \hat{v}_i\|_2^2,$$

where v_i is the true value of the impression opportunity in the record of x_i.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We believe our benchmark does not involve any potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix B.5.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [Yes] See Appendix B.5.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Appendix B.5.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix B.5.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix B.3.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]