Volume Ranking and Sequential Selection in Programmatic Display Advertising

Yuxuan Song, Kan Ren, Han Cai, Weinan Zhang, Yong Yu*
Shanghai Jiao Tong University
{songyuxuan, kren, hcai, wnzhang, yyu}@apex.sjtu.edu.cn

ABSTRACT

Programmatic display advertising, which enables advertisers to make real-time decisions on individual ad display opportunities so as to achieve a precise audience marketing, has become a key technique for online advertising. However, the constrained budget setting still restricts unlimited ad impressions. As a result, a smart strategy for ad impression selection is necessary for the advertisers to maximize positive user responses such as clicks or conversions, under the constraints of both ad volume and campaign budget. In this paper, we borrow in the idea of top-N ranking and filtering techniques from information retrieval and propose an effective ad impression volume ranking method for each ad campaign, followed by a sequential selection strategy considering the remaining ad volume and budget, to smoothly deliver the volume filtering while maximizing campaign efficiency. The extensive experiments on two benchmarking datasets and a commercial ad platform demonstrate large performance superiority of our proposed solution over traditional methods, especially under tight budgets.

1 INTRODUCTION

Nowadays, programmatic display advertising has become the mainstream advertising paradigm [29]. The programmatic buying mechanisms, including programmatic direct (PD) and real-time bidding (RTB), enable the advertisers to leverage fine-grained audience modeling and making real-time ad impression picking or bidding decisions to achieve precise marketing, which leads to high advertising efficiency [26].

A typical process for the programmatic buying is illustrated in Figure 1. When a user visits a web page or an app, an ad request is triggered and sent with the contextual information (such as user data or web page information) to the advertisers. Typically, there is a cascade of ad asking, where the first stage is to ask the PD insertion orders (with a predefined priority) for their decisions of whether to pick the ad impression; if there is no acceptance from the insertion orders, the ad request will be forwarded to private marketplace (PMP) and finally the open RTB auction for bidding via ad exchanges. With computer algorithms on the demand-side platforms (DSPs), each programmatic buying advertiser can make a real-time estimation of the utility, e.g., probability of the positive user response, and the cost of showing each specific ad impression to the corresponding user, and further dynamically determine whether to show the ad (for insertion orders) or how much to bid (for PMP or RTB) so as to maximize the campaign profit in real time [32, 48]. In this work, we focus our scope on the strategy optimization for ad impression in PD insertion orders, which is to make binary decisions of whether to accept each received ad display opportunity at a *fixed pre-negotiated price*. According to Google's report [36], the number of video ad impressions served monthly through PD insertion orders on its DoubleClick Ad Exchange has doubled from January 2016 to September 2016. Moreover, the PD insertion order advertising is expected to achieve a year-over-year increasing of 150% to 200% in 2016. The high quality of the impression and the growing market sharing have already made programmatic direct an essential part in programmatic digital advertising.

When participating in the real-world online advertising, the advertisers would quickly confront a natural contradiction between the nearly unlimited ad volume flow and, however, the limited budget. On one hand, the daily volume of ad requests is significantly huge for each advertiser [16, 44]. While on the other hand, the budget, or the allowed impression volume of the ad campaign is very limited, which requires carefully volume ranking and selection strategy to resolve the contradiction.

To solve this problem, researchers have proposed several scientific solutions. Firstly, as discussed above, one major task for the advertisers is to estimate the utility of the ad impression, which is modeled as a prediction problem for estimating the probability of positive user responses, such as clicks or conversions. We take click-through rate (CTR) estimation[1] as our running example. The estimated CTR value provides a quantitative utility modeling for the ad impression, which has been widely adopted for user ranking and volume selection [9, 11, 27, 38] in online advertising. Secondly a smart volume filtering strategy dealing with sequential ad requests is needed to deliver budget pacing [1, 8, 10]. The pacing methodology has been considered as an adaptive mechanism for controlling budget spending efficiency and smoothness.

However, these algorithms have some issues in our PD insertion orders scenario. For CTR estimation, these models pay more attention on the overall performance for CTR estimation, e.g., log loss and area under ROC curve (AUC) [30]. The obtained model may perform poor in the high predicted probability cases, which we call as the top-N case. Moreover, for budget pacing, many literatures cut the whole campaign running period as several time slices [1, 8] and adjust the impression strategy based on threshold settings, which may not be optimal because of the variance of different request arrivals.

In this paper, we propose a two-phase method to solve this problem. In the first phase, we formulate the programmatic display advertising problem as a top-N ad volume ranking task. Borrowing

*The corresponding author: Weinan Zhang.

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132853

[1]Without loss of generality, we focus on CTR estimation in this paper, while the CVR estimation can be done by following the same token.

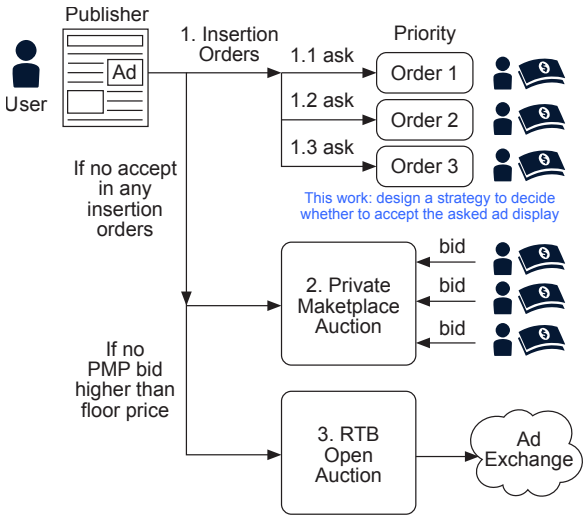

Figure 1: An illustration of the logic of a publisher to ask programmatic display ads, from the prioritized insertion orders to the private marketplace and the RTB open auctions.

the idea from top-N document/item ranking techniques in information retrieval, we derive an ad volume ranking method with dynamic negative sampling (DNS) strategy [45], which leads to a more precise prediction focused on the top ranked ad volume. To the best of our knowledge, this is the first work to explicitly address such a problem with a ranking methodology and adopt strategic negative down sampling to train the user response prediction model. In the second phase, based on the learned ranking estimator, the advertiser agent is able to estimate the relative rank (by the potential user response rate) of each ad request in the whole ad volume. Since the ad requests come to the agent in sequence and the agent needs to decide whether to pick (as an impression) or discard the current ad request in real time, we adopt a reinforcement learning model to make such sequential decisions with the remaining ad volume and budget constraints into consideration. Note that, in our scenario, since the price of each impression is pre-negotiated, so the budget constraints here means the total impression number that the advertiser can deliver (against almost unlimited ad request volume).

Extensive experiments are conducted based on two benchmarking datasets and a commercial advertising platform. The experimental results verify the performance superiority and practicality of our method over the compared baselines. We also perform theoretic derivation and empirical ablation study to analyze the relationship of DNS hyperparameters and the volume ranking performance.

The rest of the paper is organized as follows. We discuss the related work in Sec. 2. Then we formulate the problem and present our methodology in Sec. 3. The experiment settings and the corresponding results are illustrated in Sec. 4. Finally we conclude our work and discuss some future work in Sec. 5.

2 RELATED WORK

User Response Prediction. User response prediction, particularly click-through rate (CTR) or conversion rate (CVR) estimation, plays as a key role in various online advertising scenarios [19]. For auction based advertising mechanisms, such as sponsored search [17] and real-time bidding (RTB) based display advertising [40], the estimated CTR (or CVR) will be utilized as a utility estimation. Thus the existing works commonly train a CTR/CVR estimation model to improve the performance over the entire ad volume, such as log-loss [15], area under ROC curve (AUC) [30] and relative information gain (RIG) [11].

From modeling perspective, linear models such as logistic regression [19] trained with stochastic gradient descent (SGD), and non-linear models such as boosted trees [13] and (field-aware) factorization machines [15, 28] are widely used in the industry. And there are also some deep learning solutions dedicating to tackle this problem by building deep neural nets to explore high-order feature interactions [30]. Online learning methods immediately perform updating when observing each data instance with ground truth, such as Bayesian probit regression [11] and logistic regression trained with follow-the-regularized-leader (FTRL) learning [25]. However, the performance over small portion of the samples with high predicted probability has gained little attention.

To our knowledge, all above work on user response prediction tries to optimize the overall performance on the entire ad volume, whereas in our work, we only care about the user responses on the selected (i.e., top) ad volume, which has not been explicitly modeled in programmatic display advertising.

Note that learning to decide when (not) to show ad impressions has been studied in web textual advertising [7], where the authors focused on the relevance between the web page and the textual ad and adopted a binary classifier trained based on human judgments. The studied advertising paradigm, involved data and the delivered decision makers are different from our scenario.

Top Optimization. Bipartite ranking [2] aims at learning a ranking function that places positive instances above negative instances according to the predicted ranking scores. For the wide application in several areas such as information retrieval, recommender systems and computational advertising [24, 33], this topic has attracted much attention. Since in these fields, only the top ranked documents/items/ads will be exposed to users, there has been a growing interest in learning ranking functions that perform especially well at the top of the ranking list [35, 42]. We refer to this problem as *top optimization*.

There are several methods designed to solve top optimization. Some literatures aim at optimizing task-specific top ranking metrics [14, 18, 21] while some focus on pairwise ranking with listwise information [3, 31, 39]. Recently, in [6] the authors proposed an algorithm called Accuracy At The Top (AATP) to optimize the accuracy at top τ fraction. And in [5] the authors presented a large-scale convex optimization solution of AATP. Then an effective TopPush algorithm is introduced to place the positive instances in prior to the first negative instance in the ranking list in [20]. Moreover, in [22] the authors proposed a method trying to find the optimal decision boundary by modeling the top-N precision as a Mixed-Integer-Programming problem.

In our work, we start to derive our model from TopPush formulation [20] to solve the large-scale top $\mathcal{R}$ optimization, where $\mathcal{R}$ is a certain percentage of impressions against the total ad volume in our target campaign to select. There is an example for better understanding. In the real-world application, assume that we can only select $\mathcal{R} = 1\%$ in the whole request stream of this campaign, then the goal of our selection phase can be understood as picking the top 1% request within the ad requests under the guide of our ranking

function. Therefore, we only need to focus on the performance of ranking function on top 1% volume.

In TopPush algorithm, they focus on the optimization of Precision@Top, which measures the precision of the samples prior to the first negative sample. It is a suitable metric in some applications such as recommendation system that only care the very top items which are recommended to the user. However, the derivation in [20] is problematic in our situation since the optimization goal is the samples ranked before the first negative one, rather than the precision in top $\mathcal{R}$ proportion of samples.

To solve this problem, instead of optimizing the Precision@Top, we propose an effective framework to directly optimize the top $\mathcal{R}$ percentage volume precision and utilize the predicted ranking score of each ad request for sequential impression selection in the next phase.

Reinforcement Learning. Reinforcement learning (RL) has been widely used in many scenarios that involve sequential decision making or budget smoothing, including ad selection for publisher [43], music recommendation [46] and information retrieval [41], for learning an automatic and adaptive decision maker from the interaction with users.

RL provides techniques for optimizing the decision maker in a sequential interaction process with the environment to achieve a specific goal [37]. Consider the interaction process as a Markov Decision Process (MDP), which can be defined by a tuple $(\boldsymbol{S}, \{\boldsymbol{A}_s\}, \{P(s, s', a)\}, \{R(s, a)\})$, where the set of states is presented by $\boldsymbol{S}$; the set of available actions in a state $s \in \boldsymbol{S}$ is denoted as $\boldsymbol{A}_s$; $P(s, s', a)$ represents the state transition probability from state $s \in \boldsymbol{S}$ to another state $s' \in \boldsymbol{S}$ when taking action $a \in \boldsymbol{A}_s$; the reward function $R(s, a)$ represents the reward received when taking action a in state s. Under the MDP framework, the goal of the agent is expressed via the maximization of the received cumulative reward starting from an initial state.

When considering the dynamics of the environment, i.e., the modeled state transition probability and the reward function, maximizing the cumulative rewards can be achieved through dynamic programming [4], typically value iteration and policy iteration in RL. [8] presented an MDP method for RTB scenario and we follow the idea about the framework and improve the method with top optimization for sequential selection of ad impressions.

3 PROBLEM AND FORMULATION

In our scenario, we consider an agent, representing the advertiser, confronts the sequential arriving huge ad volume flow and determines whether to pick or discard each individual ad impression to achieve the advertiser's specific target (typically user responses on the delivered ad impressions). Without loss of generality, we consider clicks as the target objective, and other KPIs such as conversions can be adopted similarly. To obtain as many clicks as possible under the constrained budget, i.e., affordable impression number, a volume ranking method is required to filter low quality ad requests while preserving those with high probability of positive user responses.

In this section, we introduce a two-stage solution for this problem as illustrated in Figure 2. First, we obtain a ranking function which can provide better performance at the top of the ranking list, since we care more about the top ranked ad requests under the constrained budget of the impression numbers. Second, with the obtained ranking function, we derive the optimal sequential selection policy in a reinforcement learning fashion to smoothly spend the impression budget within each episode.

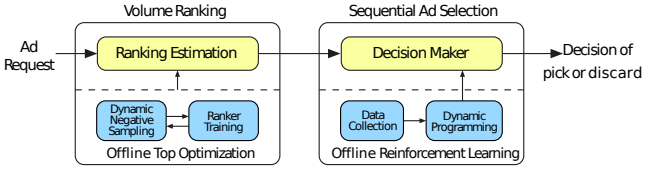

Figure 2: A diagram of our solution of volume ranking and sequential ad selection.

3.1 Preliminaries: Top Optimization

In the bipartite ranking (Sec. 2), let $\mathcal{S} = \{\boldsymbol{x}\}$ be the instance set, which can be further divided into $\mathcal{S}^+ = \{\boldsymbol{x}_i^+\}_{i=1}^m$, the set of all m positive instances, and $\mathcal{S}^- = \{\boldsymbol{x}_i^-\}_{i=1}^n$, the set of all n negative instances. To optimize the overall performance upon the whole instance set $\mathcal{S}$, e.g. in terms of AUC [12], the ranking models are typically trained to minimize the loss:

$$\mathcal{L}(f; \mathcal{S}) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}(f(\boldsymbol{x}_i^+) \le f(\boldsymbol{x}_j^-)), \tag{1}$$

where f is the bipartite ranking function and $\mathbf{1}(\cdot)$ is the indicator function.

When the optimization target is the performance over top instances in the ranking list, e.g., in information retrieval, rather than the overall performance, then the loss function in Eq. (1) is not suitable. Instead, the authors in [20] proposed an alternative loss function, which aims to place all the positive samples prior to the first negative one by learning the optimal ranking function f:

$$\mathcal{L}(f; \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(f(\boldsymbol{x}_i^+) \le \max_{1 \le j \le n} f(\boldsymbol{x}_j^-)). \tag{2}$$

While in our scenario, we hope to maximize the proportion of positive instances within top $\mathcal{R}$ (e.g., top 2%) instances in the ranking list (Precision@$\mathcal{R}$), which corresponds to the proportion of ad impressions that the advertiser is affordable to buy with limited budget w.r.t. the whole ad volume. Since we hold a different optimization target, directly applying the loss function of Eq. (1) or Eq. (2) can be problematic. In the following Sec. 3.2, we introduce an effective solution for directly optimizing Precision@$\mathcal{R}$.

3.2 DNS Solution for Optimizing Precision@$\mathcal{R}$

In real-world applications such as online advertising, the bipartite ranking confronts large bias since there are much fewer positive samples (clicks) than the negative ones, which results in difficulties in learning and prediction [13]. The researchers usually take negative down sampling as the solution [13, 45], which randomly samples negative training data to balance the ratio between the two contrarily labeled datasets. In our solution, we propose a dynamic negative down sampling method to deliver comprehensive volume ranking results in the top proportion. In this section we will at first discuss a two-step iterative improvement method to gradually improve the performance of volume ranking and then we delve more deeply into the strategy and control of the sampling operation.

Iterative Improvement Method. For optimizing Precision@$\mathcal{R}$, similar to Eq. (2), we propose a new loss function:

$$\mathcal{L}(f;\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}(f(\boldsymbol{x}_i^+) \le f(\boldsymbol{x}_{r^*}^-)) \tag{3}$$

where $\boldsymbol{x}_{r^*}^-$ is at the position of top $\mathcal{R}$ in the ranking list. Intuitively, this method wants to maximize the number of positive instances above the position of top $\mathcal{R}$. Since it is difficult to derive the closed-form solution for Eq. (3), instead we use an iterative improvement method through Dynamic Negative Sampling (DNS) to solve this problem.

The algorithm will be split up into 2 steps. Firstly, we call it as "sampling step", which means sampling the negative instances according to the ranking scores predicted by the current model. And the sampling condition is as Eq. (3). However, it is not practical to directly use a single negative instance for the next training step. For stable model training, a balanced ratio between positive and negative is significant. Therefore, during the sampling step, we try to sample a number of negative instances around $\boldsymbol{x}_{r^*}^-$ with the ratio defined as the "sampling ratio" like that in [13] for balancing negative and positive samples. Secondly we update our model parameter w according to the loss function, i.e., cross entropy [34] in our setting, with $\{\boldsymbol{x}_{r^*}^-\}$ sampled from the last sampling operation and all positive instances as the training data, and we refer to this stage as "training step". We iterative these two steps until the model convergence.

Top-1-of-k Sampling and Control Strategy. The efficiency is a great challenge in the sampling operation. Finding the instances at the exact top $\mathcal{R}$ position of the ranking function is really expensive in computation, since we may need to calculate the ranking scores for all the instances. But the instance space in display advertising is extremely large, which makes the computation cost not acceptable. To solve this problem, we propose an effective sampling and control strategy in the following.

To begin with, we define the *relative rank* in a ranked list. An item can be presented by a vector $\boldsymbol{x}$ which is a high-dimensional representation vector in display ads. And we make $r(\boldsymbol{x})$ stand for the relative rank of item $\boldsymbol{x}$ in the ranking function, e.g., if an item is ranked at 20% in the top, and the relative rank score will be 0.2. Therefore, the relative rank of all items in the list actually follows a uniform distribution. And we introduce the notation of sampling probability density function as $\mathcal{P}(r(\boldsymbol{x}))$, which presents the probability of the item $\boldsymbol{x}$ with relative rank $r(\boldsymbol{x})$ to be sampled. Then for different sampling strategies, we can derive the corresponding $\mathcal{P}(r(\boldsymbol{x}))$, the ideal $\mathcal{P}(r(\boldsymbol{x}))$ for Eq. (3) is a step function,

$$\mathcal{P}(r(\boldsymbol{x})) = \begin{cases} 1 & \text{if } \mathcal{R} - \epsilon < r(\boldsymbol{x}) < \mathcal{R} + \epsilon \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

which means we only sample the negative instances whose relative rank is around $\mathcal{R}$.

Obviously, the complexity varies with different sampling probability functions. For instance, when the probability density function $\mathcal{P}(r(\boldsymbol{x}))$ is the step function as described in Eq. (4). The sampling process will be in the complexity of $O(N \log N)$, N is the size of the whole negative set (we need to sort the predicted scores to get the relative rank). However, when the size of dataset is large, the computation will be too costly. From this perspective, we can design effective approximation strategies and make the corresponding $\mathcal{P}(r)$ be close to the ideal situation. Now considering a particular case of a $k-1$ degree polynomial function, when $\mathcal{P}(r) \propto (1-r)^{k-1}$, and the sampling process is equal to randomly sampling k negative samples, and selecting the top ranked one [45]. If T^- negative samples are needed for the training step, the complexity is $O(T^- k \log k)$. And the k is always a constant value less than 100, so the complexity actually is $O(T^-)$, consider the down sampling situation ($T^- \ll N$), this sampling strategy is highly effective. And we call this strategy as top-1-of-k sampling.

Then the correlation between the sampling probability function $\mathcal{P}(r)$ and $\mathcal{R}$ is crucial for sampling control. The expected sampled relative rank is a major characteristic of a sampling strategy. For a sampling strategy π with probability function $\mathcal{P}(r)$, the expected sampled relative rank can be computed as follows:

$$\mathcal{E} = \mathbb{E}_\pi[r] = \sum_{r\sim U(0,1)} \mathcal{P}(r) r = \int_0^1 \mathcal{P}(r) r \, dr \ . \tag{5}$$

And in top-1-of-k sampling framework, the expected sampled relative rank is:

$$\mathbb{E}_{\mathcal{P}}[r] = \int_0^1 \mathcal{P}(r) r \, dr = \frac{\int_0^1 (1-r)^{k-1} r \, dr}{\int_0^1 (1-r)^{k-1} \, dr} = \frac{1}{k+1} \tag{6}$$

Therefore, towards different $\mathcal{R}$, we can adjust k accordingly. For instance, if we want to optimize performance at the very top, which means the proportion $\mathcal{R}$ is small, then we can set a large k in sampling step. More details about the relationship between k and $\mathcal{R}$ are provided in Sec. 4.4.

3.3 RL Solution for Sequential Selection

In online advertising, after the ranking of the ad volume, the next problem is to sequentially select high quality impressions from the continually received ad requests.

Sequentially selecting the top ranked impressions in the nearly unlimited ad request flow to maximize a specific target (typically number of clicks or profit) can be naturally modeled as an episodic RL problem [8], where the agent makes decisions on whether to pick the impression at each step and is allowed to deliver at maximum B impressions, i.e. the budget of the agent, while every episode comprises T ($\gg B$) ad impressions in total. We define the budget ratio $\mathcal{R} = B/T$. To make the optimal decision to maximize the target in each episode, the agent needs to intelligently adjust its policy according to the remaining ad impressions $t \in \{0, \cdots, T\}$, the unspent budget $b \in \{0, \cdots, B\}$, and the current ad impression which is represented by a high dimensional feature vector $\boldsymbol{x}$ via one-hot encoding when making each decision. In this section, we firstly show such an environment can be easily modeled with MDP and further provide a dynamic programming solution for optimizing the agent's policy in the MDP.

Model-Based RL. An MDP, which can be represented by the tuple $(\boldsymbol{S},\ \{\boldsymbol{A}_s\},\ \{P(s,s',a)\},\ \{R(s,a)\})$, provides a general framework for modeling the sequential agent-environment interaction process (Sec. 2). In our scenario, we consider the tuple $(t, b, \boldsymbol{x}_t)$ as a state s where $\boldsymbol{x}_t$ is drawn i.i.d. from a probability density function $p_x(\boldsymbol{x})$. The full state space $\boldsymbol{S}$ is given as $\boldsymbol{S} = \{0, \cdots, T\} \times \{0, \cdots, B\} \times \boldsymbol{X}$, where $\boldsymbol{X}$ represents the whole feature vector space. And a state with $t = 0$ is a terminal state which corresponds to the end of an episode. At each non-terminal s, the agent determines whether to *pick* or *discard* the impression. As such, the set of available actions in the state s is given as $\boldsymbol{A}_s = \{pick,\ discard\}$. If the agent decides to pick the current impression, it will transit to $(t-1, b-1, \boldsymbol{x}_{t-1})$ and can transit to $(t-1, b, \boldsymbol{x}_{t-1})$ if the agent decides to discard

Table 1: A summary of our notations.

Notation	Description
$\boldsymbol{x}$	The feature vector that represents a bid request.
$p_x(\boldsymbol{x})$	The probability density function of $\boldsymbol{x}$.
$r(\boldsymbol{x})$	Relative ranking score of the impression $\boldsymbol{x}$.
$p_r(r)$	The probability density function of the ranking score r.
$V(t, b, \boldsymbol{x})$	The expected total reward with starting state $(t, b, \boldsymbol{x})$, taking the optimal policy.
$V(t, b)$	The expected total reward with starting state (t, b), taking the optimal policy.
$a(t, b, \boldsymbol{x})$	The optimal action in state $(t, b, \boldsymbol{x})$.
B	The budget of the allowed impression number in each period.
T	The volume size of the ad request flow in each period.
$\mathcal{R} = B/T$	The top proportion ratio of the budget over the volume size.

the impression. Furthermore, since we hope the agent to pick top impressions within each episode, the agent is rewarded with $r(\boldsymbol{x}_t) = 1 - r$ if the agent chooses to pick the impression $\boldsymbol{x}_t$, where r is the relative rank defined in Sec. 3.2. As such, the agent can only get the maximum reward if it picks top B impressions at each episode. The summarized state transition function and reward function are given as:

$$
\begin{aligned}
&P\Big((t, b, \boldsymbol{x}_t), (t-1, b-1, \boldsymbol{x}_{t-1}), pick\Big) = p_x(\boldsymbol{x}_{t-1}),\\
&P\Big((t, b, \boldsymbol{x}_t), (t-1, b, \boldsymbol{x}_{t-1}), discard\Big) = p_x(\boldsymbol{x}_{t-1}),\\
&R\Big((t, b, \boldsymbol{x}_t), pick\Big) = r(\boldsymbol{x}_t); \quad R\Big((t, b, \boldsymbol{x}_t), discard\Big) = 0.
\end{aligned} \tag{7}
$$

In this work, we hope to optimize a deterministic policy π, a mapping from each state $s \in \boldsymbol{S}$ to action $a \in \boldsymbol{A}_s$, i.e. $a = \pi(s)$, which corresponds to the strategy for ad impression selection in our scenario. With the policy π, we have the value function $V^\pi(s)$: the expected cumulative reward upon starting in state s and following the policy π, which satisfies the Bellman equation with the discount factor $\gamma = 1$ in our cases:

$$
V^\pi(s) = \sum_{s' \in \boldsymbol{S}} P(s, s', \pi(s)) \times V^\pi(s') + R(s, \pi(s)). \tag{8}
$$

The optimal value function is $V^*(s) = \max_\pi V^\pi(s)$ while the target optimal policy is given as

$$
\pi^*(s) = \underset{a \in \boldsymbol{A}_s}{\operatorname{argmax}} \Big\{ \sum_{s' \in \boldsymbol{S}} P(s, s', a) \times V^*(s') + R(s, a) \Big\}. \tag{9}
$$

For notation simplicity, in later sections, we use $V(s)$ to represent the optimal value function and $a(s)$ to represent the optimal policy.

Dynamic Programming Solution. With the reward function and state transition modeled as shown in Eq. (7), the optimal policy (Eq. (9)) can be derived using a dynamic programming approach, specifically value iteration in this work. As $(t, b, \boldsymbol{x})$ represents the state s, we have the optimal value function $V(t, b, \boldsymbol{x})$. Meanwhile, we need to consider situations where the next feature vector is not observed, so we introduce another optimal value function $V(t, b)$ by marginalizing out $\boldsymbol{x}$: $V(t, b) = \int_{\boldsymbol{x}} p_x(\boldsymbol{x})\, V(t, b, \boldsymbol{x})\, d\boldsymbol{x}$. Also the optimal policy is expressed as $a(t, b, \boldsymbol{x})$.

Firstly, at terminal states, the value functions should be zero, i.e. $V(0, b, \boldsymbol{x}) = V(0, b) = 0$, from the definition. And the Bellman

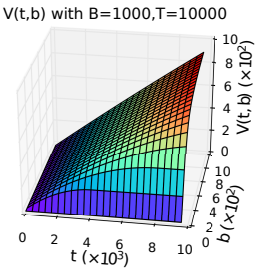

Figure 3: $V(t, b)$ in relative ranking setting.

equation of the optimal value function $V(t, b, \boldsymbol{x})$ is given as:

$$
\begin{aligned}
V(t, b, \boldsymbol{x}) &= \max \Big\{ \int_{\boldsymbol{x}_{t-1}} p_x(\boldsymbol{x}_{t-1}) V(t-1, b-1, \boldsymbol{x}_{t-1})\, d\boldsymbol{x}_{t-1} + r(\boldsymbol{x}),\\
&\qquad \int_{\boldsymbol{x}_{t-1}} p_x(\boldsymbol{x}_{t-1}) V(t-1, b, \boldsymbol{x}_{t-1})\, d\boldsymbol{x}_{t-1} \Big\}\\
&= \max \Big\{ V(t-1, b-1) + r(\boldsymbol{x}),\, V(t-1, b) \Big\},
\end{aligned} \tag{10}
$$

where the first term corresponds to picking the current ad impression while the second term corresponds to discarding the impression. $V(t, b)$ is derived by marginalizing out $\boldsymbol{x}$, and we define $\beta = V(t-1, b) - V(t-1, b-1)$:

$$
\begin{aligned}
V(t, b) &= \int_{\boldsymbol{x}} p_x(\boldsymbol{x}) \max \Big\{ V(t-1, b-1) + r(\boldsymbol{x}),\, V(t-1, b) \Big\}\, d\boldsymbol{x}\\
&= \int_{\beta < r(\boldsymbol{x}) \le 1} p_x(\boldsymbol{x}) \Big(V(t-1, b-1) + r(\boldsymbol{x}) \Big)\, d\boldsymbol{x} +\\
&\qquad \int_{0 \le r(\boldsymbol{x}) \le \beta} p_x(\boldsymbol{x}) V(t-1, b)\, d\boldsymbol{x},
\end{aligned} \tag{11}
$$

Furthermore, since $r(\boldsymbol{x})$ is a deterministic mapping from $\boldsymbol{x}$ to the relative rank, the relationship of their probability density function is determined as:

$$
p_r(r(\boldsymbol{x})) = \frac{p_x(\boldsymbol{x})}{||\nabla r(\boldsymbol{x})||}. \tag{12}
$$

With Eq. (12), the integration over the feature vector $\boldsymbol{x}$ in Eq. (11) can be transformed into integration over the relative rank r:

$$
\begin{aligned}
&\int_{\beta < r(\boldsymbol{x}) \le 1} p_x(\boldsymbol{x}) \Big(V(t-1, b-1) + r(\boldsymbol{x}) \Big)\, d\boldsymbol{x}\\
&= \int_{\beta < r(\boldsymbol{x}) \le 1} \Big(V(t-1, b-1) + r(\boldsymbol{x}) \Big) p_r(r(\boldsymbol{x}))\, ||\nabla r(\boldsymbol{x})||\, d\boldsymbol{x}\\
&= \int_{\beta < r(\boldsymbol{x}) \le 1} \Big(V(t-1, b-1) + r(\boldsymbol{x}) \Big) p_r(r(\boldsymbol{x}))\, dr(\boldsymbol{x})\\
&= \int_{\beta}^{1} \Big(V(t-1, b-1) + r \Big) p_r(r)\, dr,
\end{aligned} \tag{13}
$$

where $p_r(r) = 1$ for that $p_r(r) \sim U(0, 1)$. Similarly, we have

$$
\int_{0 \le r(\boldsymbol{x}) \le \beta} p_x(\boldsymbol{x}) V(t-1, b)\, d\boldsymbol{x} = \int_0^\beta p_r(r) V(t-1, b)\, dr. \tag{14}
$$

As such, the optimal value function $V(t, b)$ can be expressed as:

$$
\begin{aligned}
V(t, b) &= \int_\beta^1 \Big(V(t-1, b-1) + r \Big) p_r(r)\, dr + \int_0^\beta V(t-1, b) p_r(r)\, dr\\
&= (1 - \beta) V(t-1, b-1) + \frac{1 - \beta^2}{2} + \beta V(t-1, b).
\end{aligned} \tag{15}
$$

Algorithm 1 Reinforcement Learning to Select

Input: episode length T, ad impression number budget B
Output: value function $V(t, b)$
initialize $V(0, b) = 0$
for $t = 1, 2, \cdots, T-1$ **do**
  for $b = 0, 1, \cdots, B$ **do**
    set $V(t, b)$ via Eq. (15)
  end for
end for
Input: ranking function $\theta(\boldsymbol{x})$, value function $V(t, b)$, rank score function M, current state $(t_c, b_c, \boldsymbol{x}_c)$
Output: decision a_c on whether to pick or discard in current state
calculate the ranking score for the current ad request: $\theta_c = \theta(\boldsymbol{x}_c)$
compute the relative ranking score $r_c = M(\theta_c)$
if $r_c + V(t_c - 1, b_c - 1) - V(t_c - 1, b_c) \geq 0$ **then**
  $a_c \leftarrow pick$
else
  $a_c \leftarrow discard$
end if

Algorithm 2 Dynamic Negative Sampling

Input: negative (non-clicked) samples S^-, positive (clicked) samples S^+, negative down sampling ratio r
Output: ranking function $\theta(\boldsymbol{x})$, trained to optimize Precision@$\mathcal{R}$
randomly initialize the ranking function $\theta(\boldsymbol{x})$
while not converged **do**
  (sampling step)
  sample r proportion of negative samples from S^- according to the strategy described in Sec. 3.2, i.e. S_i^-
  (training step)
  train the ranking function $\theta(\boldsymbol{x})$ with S_i^- and S^+
end while

The derived optimal value function $V(t, b)$ is illustrated in Figure 3. With the derived optimal value function $V(t, b)$, the optimal policy can be expressed as:

$$a(t, b, \boldsymbol{x}) = \begin{cases} pick & \text{if } r(\boldsymbol{x}) + V(t-1, b-1) \geq V(t-1, b) \\ discard & \text{if } r(\boldsymbol{x}) + V(t-1, b-1) < V(t-1, b) \end{cases}. \quad (16)$$

The final algorithm is shown in Algorithm 1.

4 EXPERIMENTS

In this section, we first describe detailed experiment settings, including datasets, compared models, evaluation metrics. Then we present performance comparison between our proposed DNS method for Precision@$\mathcal{R}$ optimization, RL based model for sequential volume selection and some other baselines. At last we conduct a case study of hyperparameter tuning. And in the real scenario, the model will be updated after receiving a bunch of new bid request with user response ground truth. For experiment reproducibility we publish our code[2].

4.1 Datasets

We use two real-world datasets, iPinYou and YOYI, for the offline experiments. We also conduct the experiments on BEBI, a commercial programmatic display advertising platform.

iPinYou is a leading demand side platform (DSP) in China, the dataset released for the research comprises 19.5M impressions and 14.79K clicks and 16.0K CNY expense on 9 different campaigns over 10 days in 2013. The feature engineering and the splitting of the train/test sets follows [47].

YOYI is another mainstream DSP in China, and mainly focuses on the multi-device display advertising in China. And the dataset [32] comprises 441.7M impressions, 416.9K clicks and 319.5K CNY expense during 8 days in Jan. 2016. The first 7 days are set as the training data while the last day is set as the test data.

BEBI Media Limited[3] is a commercial programmatic display advertising platform in Hong Kong, on which we test our model. Note that, the ad agent on BEBI will accept all the ad requests and deliver ad impressions, which establishes an ideal test environment for our A/B testing. Specifically, the data involved in the test set comprises 21.3M impressions and 34.2K clicks during 9 days from Feb. 22, 2017 to Mar. 2, 2017. We take two largest campaigns from BEBI, namely BEBI-1 and BEBI-2.

4.2 Experiment Settings

As is mentioned previously, we focus on programmatic direct insertion order, where the advertiser and publisher would sign a contact of running a display ad campaign, including the campaign life period, the guaranteed delivery ad volume and the predefined cost, etc. Different from private marketplace or real-time bidding, the advertiser does not need to bid with others, they just need to make a binary decision, i.e., to pick or discard the given ad request. In this scenario, if the advertiser decide to pick (as an ad impression), they will directly response the ad request and show the ad, otherwise they will miss the request. The ad flow from the publishers is extremely large, while the budget (here is the total number of impressions in the contract) is relatively limited for the agent. So, for the advertiser agent, the main goal can be defined as to obtain as many user clicks as possible from the ads flow with the constrained impression budget B.

Our experiments will also be split up into 2 phases as discussed in Sec. 3. First,we will show the results of the ranking function learned with our DNS method in top $\mathcal{R}$ Precision. And then we make discussions on the sequential volume selection with RL.

Compared Models. In display advertising, the researchers always use the historical impression log to train a user response estimator for each ad campaign. In our experiments, the targeted user response is click and the problem turns out to be CTR estimation [11, 34]. And we choose logistic regression (LR) as the baseline ranking function, which is effective in handling large-scale data and widely adopted in the industry. In our work, we take two-step DNS, in the sampling phase the top-1-of-k sampling strategy (DNS-k) as described in Sec. 3.2 is adopted.

Evaluation and Data Flow. Recall that, the main goal of our method is to maximize the gained clicks under the given budget B of ad impressions. We follow [48] when building the evaluation flow, except that we will divide the test data into episodes. In each period, the ad request in historical ad logs will be sent to the agent, then the agent will decide the action of whether to pick or discard

[2]The experiment code is available at https://goo.gl/HqcZiF.

[3]https://www.bebi.com/

the request. The performance of obtained click number and other measurement results will be calculated after the period ends.

Episode Length and Budget. The episode length T is the total volume size of the ad requests defined in Sec. 3.3. In the following experiments, we set T as 100,000, which corresponds to about 10-minute period of received display requests of a medium publisher. The impression budget B is always much smaller than the size of the whole ad volume T. To present this condition, in experiments, we set $B = \mathcal{R} \cdot T$, where $\mathcal{R}$ is 0.8%, 2% for different budget constraint settings. According to the different budget settings, the optimization goal $\mathcal{R}$ in Sec. 3 is different. And we can tune the parameter k in top-1-of-k sampling according to the correlation between R and k, which is discussed in Sec. 3.2.

4.3 Experimental Results

In this section, we will present the experiment results on different datasets. At first, we discuss the performance comparison of the ranking models about top-optimization. Then, the results of the sequential volume selection model based on RL will be described in detail.

4.3.1 Ranking performance. We put more concerns on the ranking performance at top $\mathcal{R}$ proportion, rather than the global pairwise rank metric such as AUC. Thus we choose some top rank metrics for our evaluation.

Precision@$\mathcal{R}$ [20] is the fraction of positive samples at top $\mathcal{R}$ in the ranking list, which determines the upper bound of how many clicks we can acquire with this ranking function.

NDCG@$\mathcal{R}$ [23] is the normalized discounted cumulative gain which gives the more attention on the top of the ranking list.

As is described above, we take LR with uniformly negative down sampling as our baseline. And we set the sample number k of our method as 5 then the refered model is DNS-5.

Table 2 illustrates the Precision and NDCG performance with different $\mathcal{R}$. It shows that DNS-5 outperforms LR in the Precision@$\mathcal{R}$ and NDCG@$\mathcal{R}$, especially when the top proportion $\mathcal{R}$ is low. Figure 4 shows the improvement in ranking metrics at the top $\mathcal{R}$ in iPinYou dataset. We can find that, when the top proportion $\mathcal{R}$ is smaller, DNS-5 achieves relatively better performance. And with the increase of the top proportion $\mathcal{R}$, performance of DNS-5 will decline, and finally perform even inferior to LR. Associated with the AUC metric in Table 2, it shows that for a determined k, DNS-k can not improve the global rank performance, in contrary, it will sacrifice the global ranking performance, while getting better performance at the top of the ranking list. However, it is reasonable since we care more about the top ranked samples for volume selection, which will be discussed later.

4.3.2 Sequential Selection Performance. In this part, we will discuss the performance of different sequential selection strategies. And in the process of ads selection, we use the following evaluation metrics:

Clicks - the number of the user clicks from the selected ad requests flow. The goal of the selection is to get as many clicks as possible.

CTR - the click-through rate, which means the ratio of clicks against the total ad impressions in all the selected impressions. In our programmatic direct scenario, as discussed in Sec. 1, each impression is paid in a fixed pre-negotiated price, which means we can use Clicks and CTR to take place of the price related metrics, e.g. cost per click (CPC), to evaluate the effectiveness of our DNS algorithm.

RemainingVolume - the fraction of the remaining ad request volume over the whole request flow after the budget runs out.

RemainingBudgets - the fraction of the remaining budget over the whole ad impression budget when the ad display episode ends up.

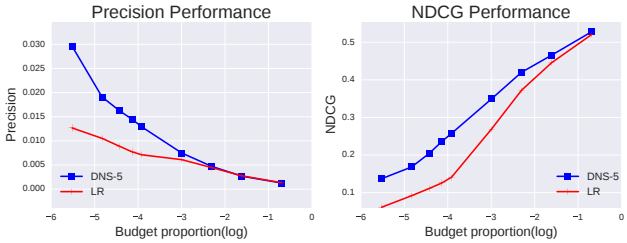

Figure 4: *Precision* and *NDCG* with different top proportion $\mathcal{R}$ in iPinYou(all).

Table 2: Detailed Precision and NDCG on our datasets with different top proportion $\mathcal{R}$.

Dataset	Algorithm	AUC(10^{-2})	Precision@$\mathcal{R}$ (10^{-3})			NDCG@$\mathcal{R}$ (10^{-2})		
			0.4%	1.2%	2%	0.4%	1.2%	2%
iPinYou(all)	LR	85.21	12.68	8.86	7.11	6.14	11.14	14.13
	DNS-5	83.21	29.63	16.20	12.88	13.71	20.48	25.77
YOYI	LR	91.08	22.26	17.79	14.67	9.12	17.93	23.15
	DNS-5	86.00	25.44	15.59	12.99	9.88	15.92	20.68
BEBI	LR	74.96	11.66	9.59	8.54	2.71	5.80	8.15
	DNS-5	68.94	18.13	10.35	8.36	4.22	6.55	8.36
BEBI-1	LR	82.73	5.35	3.90	3.51	2.95	5.50	7.66
	DNS-5	77.86	7.81	4.96	4.16	5.62	8.42	10.62
BEBI-2	LR	91.53	3.95	4.03	4.02	2.76	7.28	11.38
	DNS-5	92.18	7.90	6.18	5.84	5.65	11.53	17.00

In this part, our compared baseline is a threshold-based method which sets the threshold according to the ranking score function derived from the train data. For example, if budget is $\mathcal{R}$ percentage, then the goal is to find the top $\mathcal{R}$ percentage impressions in the test data. Therefore, the relative rank threshold will be set as $\mathcal{R}$. For each given ad request, if the estimated relative rank score is lower than the threshold, the ad request will be discarded. Otherwise we will deliver the ad impression. We regard this method as the constant threshold (CT) method. And our method has been described in Algorithm 1, and will be referred to as DP, named as dynamic programming in the following experiments. To represent the selection models with different ranking score functions, we concatenate the model names. For example, when referring to the DP selection strategy with the ranking score function made by DNS-5, we name it as DNS-5-DP.

Tables 3, 4, 5, 6 and 7 depict the performance of different models with different selection strategies over the datasets. From the tables, we can find that: (i) Methods with DNS-5 always achieve relatively higher CTR and Clicks numbers than LR. It is reasonable since, from the analysis in Sec. 4.3.1, we have already known that DNS can effectively improve the precision at top. (ii) DP will spend all the budget and meet all the display opportunities. Algorithm 1 can be regarded as a budget pacing strategy. More discussions can be seen in Sec. 4.4. (iii) The DNS-CT method always spends budget too aggressively to see all the ad requests. However, the LR-CT

Table 3: Performance in Different Selection Strategies with budget $B = 0.8\% \cdot T$ and $2\% \cdot T$ in iPinYou .

Strategy	$B = 0.8\% \cdot T$			
	Clicks	CTR (10^{-2})	RemainingVolume	RemainingBudgets
DNS-5-DP	860	2.62	0%	0%
DNS-5-CT	663	2.02	0%	0%
LR-DP	513	1.56	0 %	0.02%
LR-CT	361	1.10	9.97%	0%
	$B = 2\% \cdot T$			
DNS-5-DP	1137	1.39	0%	0%
DNS-5-CT	1096	1.34	0.7%	0%
LR-DP	835	1.02	0 %	0%
LR-CT	692	0.84	10.24%	0%

Table 4: Performance in Different Selection Strategies with budget $B = 0.8\% \cdot T$ and $2\% \cdot T$ in YOYI .

Strategy	$B = 0.8\% \cdot T$			
	Clicks	CTR (10^{-2})	RemainingVolume	RemainingBudgets
DNS-5-DP	162	1.84	0%	0%
DNS-5-CT	168	1.91	14.58%	0%
LR-DP	137	1.56	0 %	0.68%
LR-CT	69	2.72	0%	71.17%
	$B = 2\% \cdot T$			
DNS-5-DP	305	1.39	0%	0%
DNS-5-CT	304	1.38	14.42%	0%
LR-DP	302	1.37	0 %	0.07%
LR-CT	152	2.01	0%	65.73%

Table 5: Performance in Different Selection Strategies with-budget $B = 0.8\% \cdot T$ and $2\% \cdot T$ in BEBI .

Strategy	$B = 0.8\% \cdot T$			
	Clicks	CTR (10^{-2})	RemainingVolume	RemainingBudgets
DNS-5-DP	275	1.25	0%	0%
DNS-5-CT	269	1.23	6.63%	0%
LR-DP	207	0.95	0 %	0%
LR-CT	122	1.18	0%	53.06%
	$B = 2\% \cdot T$			
DNS-5-DP	460	0.84	0%	0%
DNS-5-CT	445	0.84	0%	3.4%
LR-DP	438	0.80	0 %	0.4%
LR-CT	316	0.96	0%	40.02%

Table 6: Performance in Different Selection Strategies with budget $B = 0.8\% \cdot T$ and $2\% \cdot T$ in BEBI-1 .

Strategy	$B = 0.8\% \cdot T$			
	Clicks	CTR (10^{-2})	RemainingVolume	RemainingBudgets
DNS-5-DP	139	0.50	0%	0%
DNS-5-CT	101	0.37	58.94%	0%
LR-DP	124	0.55	0 %	0%
LR-CT	101	0.51	0%	28.22%
	$B = 2\% \cdot T$			
DNS-5-DP	272	0.39	0%	0%
DNS-5-CT	232	0.34	43.83%	0%
LR-DP	241	0.34	0 %	0.23%
LR-CT	197	0.36	0%	20.68%

method always fails to use up all the budget. This is for that the distributions of the training data and the test data always differ from each other, so the constant threshold set by CT model will suit in training set but probably not on both. (iv) Moreover, it shows that DNS-5-DP always makes the best performance in clicks and CTR because this method focuses on the top precision and executes a smart pacing strategy during the selection process.

Table 7: Performance in Different Selection Strategies with budget $B = 0.8\% \cdot T$ and $2\% \cdot T$ in BEBI-2 .

Strategy	$B = 0.8\% \cdot T$			
	Clicks	CTR (10^{-2})	RemainingVolume	RemainingBudgets
DNS-5-DP	113	0.67	0%	0%
DNS-5-CT	105	0.63	28.41%	0%
LR-DP	71	0.43	0.66 %	0%
LR-CT	50	0.45	0%	33.67%
	$B = 2\% \cdot T$			
DNS-5-DP	243	0.58	0%	0%
DNS-5-CT	227	0.54	16.09%	0%
LR-DP	162	0.39	0 %	0.5%
LR-CT	109	0.38	0%	31.33%

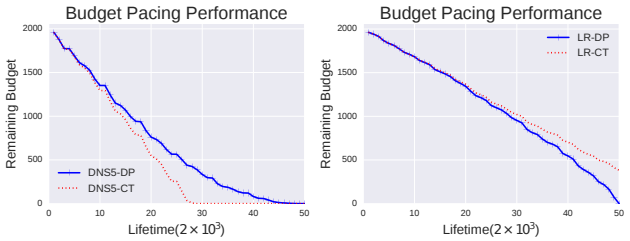

Figure 5: Budget Pacing Performance of Single Episode in BEBI-1 with Budget $B = 2\% \cdot T$.

4.4 Ablation Study

In this section, we make some discussions about the budget pacing of the compared models, and further conduct a case study on the hyperparameter learning of our algorithm.

Budget Pacing Performance. The goal of Algorithm 1 is to select the top ranked volume with the current ranking function in the ad request flow under the impression budget constraint. As we can see from Figure 3, intuitively, sequential selection is just like a budget pacing mechanism. When the budget is loose, DP will improve the selection threshold and when budget is tight, DP will contrarily decrease the selection threshold dynamically. In other words, from the budget perspective, when the spending speed is too fast, DP method will adjust the speed to a lower level and vice versa. Figure 5 shows the varying budget spending w.r.t. the remaining budgets under different strategies in BEBI-1. We may find that DP will adjust the spending speed and make the ad impression activity smooth in the flow of large volume ad requests.

Sampling Parameter. As is discussed in Sec. 3.2, we find the optimal value of the sampling parameter k in Algorithm 2 varies according to the top $\mathcal{R}$ proportion value. This correlation can be observed in Figure 6. It shows when the budget is tight, the optimal value of k is relatively large. With the budget constraint getting looser, the optimal k reduces. Moreover, when the budget is extremely loose, e.g. 50%, the optimal k is just 1, which means that we sample one negative instance at each sampling step (DNS-1) in Algorithm 2. Note that DNS-1 is actually LR model with uniform negative sampling.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we study the volume ranking and sequential selection problem in programmatic display advertising. For volume ranking, we proposed an effective top optimization method through strategic negative down sampling. And we also proposed a solution of sequential selection based on reinforcement learning. We show that

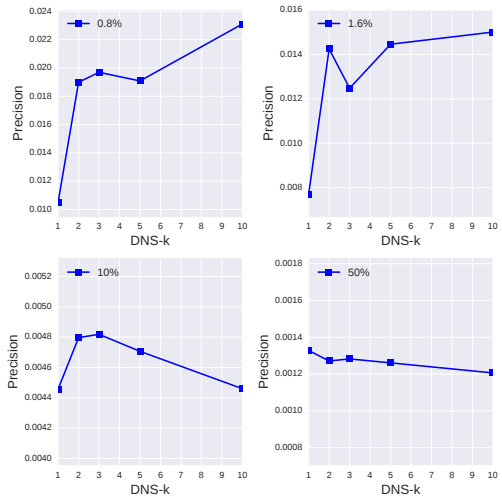

Figure 6: Performance of Different Dynamic Sampling Parameters in Different proportion $\mathcal{R}$ in iPinYou .

these problems can be solved by an approximately top optimization method and a model based reinforcement learning method. Experimental results have shown that the proposed ranking model can improve the top volume ranking metrics, based on which the sequential binary decision maker of ad request selection successfully improves the achieved click number within the impression budget.

For future work, we plan to delve deeper into the investigation of the relationship between the optimal parameter k and $\mathcal{R}$ as described in Sec. 4.4. To solve this problem, we may need to give an approximated closed-form explanation of DNS. Furthermore, we would apply the proposed method in multi-advertiser scenario and the header bidding problem.

Acknowledgement The work is financially supported by NSFC (61632017) and Shanghai Sailing Program (17YF1428200). We thank Ufuk Altinok of Bebi Media to help us run experiments on Bebi ad platform.

REFERENCES

[1] Deepak Agarwal, Souvik Ghosh, Kai Wei, and Siyu You. 2014. Budget pacing for targeted online advertisements at linkedin. In *KDD*.
[2] Shivani Agarwal. 2005. A study of the bipartite ranking problem in machine learning. (2005).
[3] Shivani Agarwal. 2011. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *Proceedings of the 2011 SIAM International Conference on Data Mining*.
[4] Dimitri P Bertsekas. 1995. *Dynamic programming and optimal control*.
[5] Stephen Boyd, Corinna Cortes, Chong Jiang, Mehryar Mohri, Ana Radovanovic, and Joelle Skaf. 2012. Large-scale distributed optimization for improving accuracy at the top. In *NIPS Workshop on Optimization for Machine Learning*.
[6] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. 2012. Accuracy at the top. In *NIPS*.
[7] Andrei Broder, Massimiliano Ciaramita, Marcus Fontoura, et al. 2008. To swing or not to swing: learning when (not) to advertise. In *CIKM*.
[8] H Cai, K Ren, W Zhag, K Malialis, and J Wang. 2017. Real-Time Bidding by Reinforcement Learning in Display Advertising. In *WSDM*.
[9] Olivier Chapelle. 2014. Modeling delayed feedback in display advertising. In *KDD*.
[10] Joaquin Fernandez-Tapia. 2015. *An analytical solution to the budget-pacing problem in programmatic advertising*. Technical Report. Technical report.
[11] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*.
[12] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* (1982).
[13] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*.
[14] Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *ICML*.
[15] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *RecSys*.
[16] Nitish Korula, Vahab Mirrokni, and Hamid Nazerzadeh. 2016. Optimizing display advertising markets: Challenges and directions. *IEEE Internet Computing* (2016).
[17] Sébastien Lahaie, David M Pennock, Amin Saberi, and Rakesh V Vohra. 2007. Sponsored search auctions. *Algorithmic game theory* (2007).
[18] Quoc Le and Alexander Smola. 2007. Direct optimization of ranking measures. *arXiv preprint arXiv:0704.3359* (2007).
[19] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *KDD*.
[20] Nan Li, Rong Jin, and Zhi-Hua Zhou. 2014. Top rank optimization in linear time. In *NIPS*.
[21] Nan Li, Ivor W Tsang, and Zhi-Hua Zhou. 2013. Efficient optimization of performance measures by classifier adaptation. *IEEE transactions on pattern analysis and machine intelligence* (2013).
[22] Li-Ping Liu, Thomas G Dietterich, Nan Li, and Zhi-Hua Zhou. 2015. Transductive optimization of top k precision. *arXiv preprint arXiv:1510.05976* (2015).
[23] Nathan N Liu and Qiang Yang. 2008. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR*.
[24] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* (2009).
[25] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *KDD*.
[26] Andrew McStay. 2017. 3.1 Micro-Moments, Liquidity, Intimacy and Automation: Developments in Programmatic Ad-tech. (2017).
[27] Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. 2011. Response prediction using collaborative filtering with hierarchies and side-information. In *KDD*.
[28] Richard J Oentaryo, Ee-Peng Lim, Jia-Wei Low, David Lo, and Michael Finegold. 2014. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *WSDM*.
[29] PwC and Interactive Advertising Bureau. 2017. IAB internet advertising revenue report: 2016 full year. (2017).
[30] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. *arXiv preprint arXiv:1611.00144* (2016).
[31] Alain Rakotomamonjy. 2012. Sparse support vector infinite push. *arXiv preprint arXiv:1206.6432* (2012).
[32] Kan Ren, Weinan Zhang, Yifei Rong, Haifeng Zhang, Yong Yu, and Jun Wang. 2016. User response learning for directly optimizing campaign performance in display advertising. In *CIKM*.
[33] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*.
[34] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*.
[35] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. 2012. TFMAP: optimizing MAP for top-n context-aware recommendation. In *SIGIR*.
[36] George Slefo. 2016. Google Says Programmatic Direct Is Due for a Rise in 2017. http://adage.com/article/digital/google-programmatic-direct-rise-2017/307111/. (2016). [Online;Accessed: 2017-09-02].
[37] Richard S Sutton. 1998. Reinforcement learning. *An introduction* (1998).
[38] Anh-Phuong Ta. 2015. Factorization machines with follow-the-regularized-leader for CTR prediction in display advertising. In *Big Data (Big Data), 2015 IEEE International Conference on*.
[39] Nicolas Usunier, David Buffoni, and Patrick Gallinari. 2009. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th annual international conference on machine learning*.
[40] Jun Wang, Weinan Zhang, and Shuai Yuan. 2016. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting. *arXiv preprint arXiv:1610.03013* (2016).
[41] Hui Yang, Marc Sloan, and Jun Wang. 2014. Dynamic information retrieval modeling. In *SIGIR*.
[42] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *CIKM*.
[43] Shuai Yuan and Jun Wang. 2012. Sequential selection of correlated ads by POMDPs. In *CIKM*.
[44] H Zhang, W Zhang, J Wang, and Y Rong. 2017. Managing Risk of Bidding in Display Advertising. In *WSDM*.
[45] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR*.
[46] Weinan Zhang, Ulrich Paquet, and Katja Hofmann. 2016. Collective Noise Contrastive Estimation for Policy Transfer Learning.. In *AAAI*.
[47] Weinan Zhang and Jun Wang. 2015. Statistical arbitrage mining for display advertising. In *KDD*.
[48] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal real-time bidding for display advertising. In *KDD*.