Real-Time Bidding by Reinforcement Learning in Display Advertising

[†]Han Cai, [†]Kan Ren, [†]Weinan Zhang,[*]
[‡]Kleanthis Malialis, [‡]Jun Wang, [†]Yong Yu, [♯]Defeng Guo
[†]Shanghai Jiao Tong University, [‡]University College London, [♯]Vlion Inc.
{hcai,kren,wnzhang}@apex.sjtu.edu.cn, j.wang@cs.ucl.ac.uk

ABSTRACT

The majority of online display ads are served through real-time bidding (RTB) — each ad display impression is auctioned off in real-time when it is just being generated from a user visit. To place an ad automatically and optimally, it is critical for advertisers to devise a learning algorithm to cleverly bid an ad impression in real-time. Most previous works consider the bid decision as a static optimization problem of either treating the value of each impression independently or setting a bid price to each segment of ad volume. However, the bidding for a given ad campaign would repeatedly happen during its life span before the budget runs out. As such, each bid is strategically correlated by the constrained budget and the overall effectiveness of the campaign (e.g., the rewards from generated clicks), which is only observed after the campaign has completed. Thus, it is of great interest to devise an optimal bidding strategy sequentially so that the campaign budget can be dynamically allocated across all the available impressions on the basis of both the immediate and future rewards. In this paper, we formulate the bid decision process as a reinforcement learning problem, where the state space is represented by the auction information and the campaign's real-time parameters, while an action is the bid price to set. By modeling the state transition via auction competition, we build a Markov Decision Process framework for learning the optimal bidding policy to optimize the advertising performance in the dynamic real-time bidding environment. Furthermore, the scalability problem from the large real-world auction volume and campaign budget is well handled by state value approximation using neural networks. The empirical study on two large-scale real-world datasets and the live A/B testing on a commercial platform have demonstrated the superior performance and high efficiency compared to state-of-the-art methods.

Keywords

Bid Optimization, Reinforcement Learning, Display Ads

[*]Weinan Zhang is the corresponding author of this paper.

WSDM 2017, February 06-10, 2017, Cambridge, United Kingdom

DOI: http://dx.doi.org/10.1145/3018661.3018702

1. INTRODUCTION

The increased availability of big data and the improved computational power have advanced machine learning and artificial intelligence for various prediction and decision making tasks. In particular, the successful application of reinforcement learning in certain settings such as gaming control [19] has demonstrated that machines not only can predict, but also have a potential of achieving comparable human-level control and decision making. In this paper, we study *machine bidding* in the context of display advertising. Auctions, particularly real-time bidding (RTB), have been a major trading mechanism for online display advertising [30, 7, 23]. Unlike the keyword-level bid decision in sponsored search [1], the advertiser needs to make the impression-level bid decision in RTB, i.e., bidding for every single ad impression in real time when it is just being generated by a user visit [18, 30] . Machine based bid decision, i.e., to calculate the strategic amount that the advertiser would like to pay for an ad opportunity, constitutes a core component that drives the campaigns' ROI [11, 32]. By calculating an optimal bid price for each ad auction (also considering the remaining budget and the future availability of relevant ad impressions in the ad exchange) and then observing the auction result and user feedback, the advertiser would be able to refine their bidding strategy and better allocate the campaign budget across the online page view volume.

A straightforward bidding strategy in RTB display advertising is mainly based on the truthfulness of second-price auctions [6], which means the bid price for each ad impression should be equal to its true value, i.e., the action value (e.g., click value) multiplied by the action rate (e.g., click-through rate) [14]. However, for budgeted bidding in repeated auctions, the optimal bidding strategy may not be truthful but depends on the market competition, auction volume and campaign budget [31]. In [18, 32], researchers have proposed to seek the optimal bidding function that directly maximizes the campaign's key performance indicator (KPI), e.g., total clicks or revenue, based on the static distribution of input data and market competition models. Nevertheless, such static bid optimization frameworks may still not work well in practice because the RTB market competition is highly dynamic and it is fairly common that the true data distribution heavily deviates from the assumed one during the model training [4], which requires additional control step such as budget pacing [28] to constrain the budget spending.

In this paper, we solve the issue by considering bidding as a sequential decision, and formulate it as a *reinforcement*

learning to bid (RLB) problem. From an advertiser's perspective, the whole ad market and Internet users are regarded as the environment. At each timestep, the advertiser's bidding agent observes a state, which consists of the campaign's current parameters, such as the remaining lifetime and budget, and the bid request for a specific ad impression (containing the information about the underlying user and their context) . With such state (and context), the bidding agent makes a bid action for its ad. After the ad auction, the winning results with the cost and the corresponding user feedback will be sent back to the bidding agent, which forms the reward signal of the bidding action. Thus, the bid decision aims to derive an optimal bidding policy for each given bid request.

With the above settings, we build a Markov Decision Process (MDP) framework for learning the optimal bidding policy to optimize the advertising performance. The value of each state will be calculated by performing dynamic programming. Furthermore, to handle the scalability problem for the real-world auction volume and campaign budget, we propose to leverage a neural network model to approximate the value function. Besides directly generalizing the neural network value function, we also propose a novel *coarse-to-fine episode segmentation model* and *state mapping models* to overcome the large-scale state generalization problem.

In our empirical study, the proposed solution has achieved 16.7% and 7.4% performance gains against the state-of-the-art methods on two large-scale real-world datasets. In addition, our proposed system has been deployed into a commercial RTB platform. We have performed an online A/B testing, where a 44.7% improvement in click performance was observed against a most widely used method in the industry.

2. BACKGROUND AND RELATED WORK

Reinforcement Learning. An MDP provides a mathematical framework which is widely used for modelling the dynamics of an environment under different actions, and is useful for solving reinforcement learning problems [21]. An MDP is defined by the tuple $\langle S, A, P, R\rangle$. The set of all states and actions are represented by S and A respectively. The reward and transition probability functions are given by R and P. Dynamic programming is used in cases where the environment's dynamics, i.e., the reward function and transition probabilities are known in advance. Two popular dynamic programming algorithms are policy iteration and value iteration. For large-scale situations, it is difficult to experience the whole state space, which leads to the use of function approximation that constructs an approximator of the entire function [8, 22]. In this work, we use value iteration for small-scale situations, and further build a neural network approximator to solve the scalability problem.

RTB Strategy. In the RTB process [32], the advertiser receives the bid request of an ad impression with its real-time information and the very first thing to do is to estimate the *utility*, i.e., the user's response on the ad if winning the auction. The distribution of the *cost*, i.e., the market price [1, 5], which is the highest bid price from other competitors, is also forecasted by the *bid landscape forecasting* component. Utility estimation and bid landscape forecasting are described below. Given the estimated utility and cost factors, the *bidding strategy* [30] decides the final bid price with accessing the information of the remaining budget and auction volume. Thus it is crucial to optimize the final bidding strategy considering the market and bid request information with budget constraints. A recent comprehensive study on the data science of RTB display advertising is posted in [24].

Utility Estimation. For advertisers, the utility is usually defined based on the user response, i.e., click or conversion, and can be modeled as a probability estimation task [14]. Much work has been proposed for user response prediction, e.g., click-through rate (CTR) [16], conversion rate (CVR) [14] and post-click conversion delay patterns [3]. For modeling, linear models such as logistic regression [14] and nonlinear models such as boosted trees [10] and factorization machines [17] are widely used in practice. There are also online learning models that immediately perform updating when observing each data instance, such as Bayesian probit regression [9], FTRL learning in logistic regression [16]. In our paper, we follow [14, 32] and adopt the most widely used logistic regression for utility estimation to model the reward on agent actions.

Bid Landscape Forecasting. Bid landscape forecasting refers to modeling the market price distribution for auctions of specific ad inventory, and its c.d.f. is the winning probability given each specific bid price [5]. The authors in [15, 32, 5] presented some hypothetical winning functions and learned the parameters. For example, a log-normal market price distribution with the parameters estimated by gradient boosting decision trees was proposed in [5]. Since advertisers only observe the winning impressions, the problem of censored data [1, 33] is critical. Authors in [27] proposed to leverage censored linear regression to jointly model the likelihood of observed market prices in winning cases and censored ones with losing bids. Recently, the authors in [25] proposed to combine survival analysis and decision tree models, where each tree leaf maintains a non-parametric survival model to fit the censored market prices. In this paper, we follow [1, 33] and use a non-parametric method to model the market price distribution.

Bid Optimization. As has been discussed above, bidding strategy optimization is the key component within the decision process for the advertisers [18]. The auction theory [12] proved that truthful bidding is the optimal strategy under the second-price auction. However, truthful bidding may perform poorly when considering the multiple auctions and the budget constraint [32]. In real-world applications, the linear bidding function [18] is widely used. The authors in [32] empirically showed that there existed non-linear bidding functions better than the linear ones under variant budget constraints. When the data changes, however, the heuristic model [18] or hypothetical bidding functions [31, 32] cannot depict well the real data distribution. The authors in [1, 29] proposed the model-based MDPs to derive the optimal policy for bidding in sponsored search or ad selection in contextual advertising, where the decision is made on keyword level. In our work, we investigate the most challenging impression-level bid decision problem in RTB display advertising that is totally different from [1, 29]. We also tackle the scalability problem, which remains unsolved in [1], and demonstrate the efficiency and effectiveness of our method in a variety of experiments.

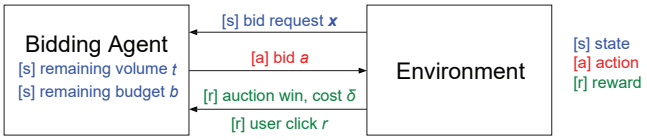

Figure 1: Diagram of reinforcement learning to bid.

3. PROBLEM AND FORMULATION

In a RTB ad auction, each bidding agent acts on behalf of its advertiser and generates bids to achieve the campaign's specific target. Our main goal is to derive the optimal bidding policy in a reinforcement learning fashion. For most performance-driven campaigns, the optimization target is to maximize the user responses on the displayed ads if the bid leads to auction winning. Without loss of generality, we consider clicks as our targeted user response objective, while other KPIs can be adopted similarly. The diagram of interactions between the bidding agent and the environment is shown in Figure 1.

3.1 Problem Definition

Mathematically, we consider bidding in display advertising as an episodic process [1]; each episode comprises T auctions which are sequentially sent to the bidding agent. Each auction is represented by a high dimensional feature vector $\boldsymbol{x}$, which is indexed via one-hot binary encoding. Each entry of $\boldsymbol{x}$ corresponds to a category in a field, such as the category `London` in the field `City`, and the category `Friday` in the field `Weekday`. The fields consist of the campaign's ad information (e.g., ad creative ID and campaign ID) and the auctioned impression contextual information (e.g., user cookie ID, location, time, publisher domain and URL).

At the beginning, the agent is initialized with a budget B, and the advertising target is set to acquire as many clicks as possible during the following T auctions. Three main pieces of information are considered by the agent (i) the remaining auction number $t \in \{0, \cdots, T\}$; (ii) the unspent budget $b \in \{0, \cdots, B\}$ and (iii) the feature vector $\boldsymbol{x}$. During the episode, each auction will be sent to the agent sequentially and for each of them the agent needs to decide the bid price according to the current information t, b and $\boldsymbol{x}$.

The agent maintains the remaining number of auctions t and the remaining budget b. At each timestep, the agent receives an auction $\boldsymbol{x} \in \boldsymbol{X}$ (the feature vector space), and determines its bid price a. We denote the market price p.d.f. given $\boldsymbol{x}$ as $m(\delta, \boldsymbol{x})$, where δ is the market price and m is its probability. If the agent bids at price $a \geq \delta$, then it wins the auction and pays δ, and the remaining budget changes to $b - \delta$. In this case, the agent can observe the user response and the market price later. Alternatively, if losing, the agent gets nothing from the auction. We take predicted CTR (pCTR) $\theta(\boldsymbol{x})$ as the expected reward, to model the action utility. After each auction, the remaining number of auctions changes to $t - 1$. When the auction flow of this episode runs out, the current episode ends and both the remaining auction number and budget are reset.

3.2 MDP Formulation of RTB

A Markov Decision Process (MDP) provides a framework that is widely used for modeling agent-environment interactions [21]. Our notations are listed in Table 1. An MDP can

Table 1: A summary of our notations.

Notation	Description
$\boldsymbol{x}$	The feature vector that represents a bid request.
$\boldsymbol{X}$	The whole feature vector space.
$p_x(\boldsymbol{x})$	The probability density function of $\boldsymbol{x}$.
$\theta(\boldsymbol{x})$	The predicted CTR (pCTR) if winning the auction of $\boldsymbol{x}$.
$m(\delta, \boldsymbol{x})$	The p.d.f. of market price δ given $\boldsymbol{x}$.
$m(\delta)$	The p.d.f. of market price δ.
$V(t, b, \boldsymbol{x})$	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 $(t, b, \boldsymbol{x})$.

be represented by a tuple $(\boldsymbol{S}, \{\boldsymbol{A}_s\}, \{P^a_{ss'}\}, \{R^a_{ss'}\})$, where $\boldsymbol{S}$ and $\boldsymbol{A}_s$ are two sets of all states and all possible actions in state $s \in \boldsymbol{S}$, respectively. $P^a_{ss'}$ represents the state transition probability from state $s \in \boldsymbol{S}$ to another state $s' \in \boldsymbol{S}$ when taking action $a \in \boldsymbol{A}_s$, which is denoted by $\mu(a, s, s')$. Similarly, $R^a_{ss'}$ is the reward function denoted by $r(a, s, s')$ that represents the reward received after taking action a in state s and then transiting to state s'.

We consider $(t, b, \boldsymbol{x}_t)$ as a state s[1] and assume the feature vector $\boldsymbol{x}_t$ is drawn i.i.d. from the probability density function $p_x(\boldsymbol{x})$. The full state space is $\boldsymbol{S} = \{0, \cdots, T\} \times \{0, \cdots, B\} \times \boldsymbol{X}$. And if $t = 0$, the state is regarded as a terminal state which means the end of the episode. The set of all actions available in state $(t, b, \boldsymbol{x}_t)$ is $\boldsymbol{A}_{(t,b,\boldsymbol{x}_t)} = \{0, \cdots, b\}$, corresponding to the bid price. Furthermore, in state $(t, b, \boldsymbol{x}_t)$ where $t > 0$, the agent, when bidding a, can transit to $(t-1, b-\delta, \boldsymbol{x}_{t-1})$ with probability $p_x(\boldsymbol{x}_{t-1})m(\delta, \boldsymbol{x}_t)$ where $\delta \in \{0, \cdots, a\}$ and $\boldsymbol{x}_{t-1} \in \boldsymbol{X}$. That is the case of winning the auction and receiving a reward $\theta(\boldsymbol{x}_t)$. And the agent may lose the auction whereafter transit to $(t-1, b, \boldsymbol{x}_{t-1})$ with probability $p_x(\boldsymbol{x}_{t-1}) \sum_{\delta=a+1}^{\infty} m(\delta, \boldsymbol{x}_t)$, where $\boldsymbol{x}_{t-1} \in \boldsymbol{X}$. All other transitions are impossible because of the auction process. In summary, transition probabilities and reward function can be written as:

$$
\begin{aligned}
\mu\Big(a, (t, b, \boldsymbol{x}_t), (t-1, b-\delta, \boldsymbol{x}_{t-1})\Big) &= p_x(\boldsymbol{x}_{t-1}) m(\delta, \boldsymbol{x}_t), \\
\mu\Big(a, (t, b, \boldsymbol{x}_t), (t-1, b, \boldsymbol{x}_{t-1})\Big) &= p_x(\boldsymbol{x}_{t-1}) \sum_{\delta=a+1}^{\infty} m(\delta, \boldsymbol{x}_t), \\
r\Big(a, (t, b, \boldsymbol{x}_t), (t-1, b-\delta, \boldsymbol{x}_{t-1})\Big) &= \theta(\boldsymbol{x}_t), \\
r\Big(a, (t, b, \boldsymbol{x}_t), (t-1, b, \boldsymbol{x}_{t-1})\Big) &= 0,
\end{aligned} \tag{1}
$$

where $\delta \in \{0, \cdots, a\}$, $\boldsymbol{x}_{t-1} \in \boldsymbol{X}$ and $t > 0$. Specifically, the first equation is the transition when giving a bid price $a \geq \delta$, while the second equation is the transition when losing the auction.

A deterministic policy π is 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 bidding strategy in RTB display advertising. According to the policy π, we have the value function $V^\pi(s)$: the expected sum of rewards upon starting in state s and obeying policy π. This satisfies the Bellman equation with the discount factor $\gamma = 1$ since in our scenario the total click number is the optimization target, regardless of the click time.

$$
V^\pi(s) = \sum_{s' \in \boldsymbol{S}} \mu(\pi(s), s, s') \Big(r(\pi(s), s, s') + V^\pi(s') \Big) \tag{2}
$$

The optimal value function is defined as $V^*(s) = \max_\pi V^\pi(s)$.

[1]For simplicity, we slightly abuse the notation by including t in the state.

We also have the optimal policy as:

$$\pi^*(s) = \underset{a \in \boldsymbol{A}_s}{\operatorname{argmax}} \Big\{ \sum_{s' \in \boldsymbol{S}} \mu(a, s, s') \Big(r(a, s, s') + V^*(s') \Big) \Big\}, \quad (3)$$

which gives the optimal action at each state s and $V^*(s) = V^{\pi^*}(s)$. The optimal policy $\pi^*(s)$ is exactly the optimal bidding strategy we want to find. For notation simplicity, in later sections, we use $V(s)$ to represent the optimal value function $V^*(s)$.

One may consider the possibility of model-free approaches [26, 20] to directly learn the bidding policy from experience. However, such model-free approaches may suffer from the problems of transition dynamics of the enormous state space, the sparsity of the reward signals and the highly stochastic environment. Since there are many previous works on modeling the utility (reward) and the market price distribution (transition probability) as discussed in Section 2, we take advantage of them and propose our model-based solution for this problem.

4. DYNAMIC PROGRAMMING SOLUTION

In a small scale, Eq. (3) can be solved using a dynamic programming approach. As defined, we have the optimal value function $V(t, b, \boldsymbol{x})$, where $(t, b, \boldsymbol{x})$ represents the state s. Meanwhile, we consider the situations where we do not observe the feature vector $\boldsymbol{x}$; so another optimal value function is $V(t, b)$: the expected total reward upon starting in (t, b) without observing the feature vector $\boldsymbol{x}$ when the agent takes the optimal policy. It satisfies $V(t, b) = \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) V(t, b, \boldsymbol{x})\, d\boldsymbol{x}$. Also that, we have the optimal policy π^* and express it as the optimal action $a(t, b, \boldsymbol{x})$.

From the definition, we have $V(0, b, \boldsymbol{x}) = V(0, b) = 0$ as the agent gets nothing when there are no remaining auctions. Combined with the transition probability and reward function described in Eq. (1), the definition of $V(t, b)$, $V(t, b, \boldsymbol{x})$ can be expressed with $V(t-1, \cdot)$ as

$$\begin{aligned} V(t, b, \boldsymbol{x}) = \max_{0 \le a \le b} \Big\{ & \sum_{\delta=0}^{a} \int_{\boldsymbol{X}} m(\delta, \boldsymbol{x}) p_x(\boldsymbol{x}_{t-1}) \cdot \\ & \Big(\theta(\boldsymbol{x}) + V(t-1, b-\delta, \boldsymbol{x}_{t-1}) \Big)\, d\boldsymbol{x}_{t-1} + \\ & \sum_{\delta=a+1}^{\infty} \int_{\boldsymbol{X}} m(\delta, \boldsymbol{x}) p_x(\boldsymbol{x}_{t-1}) V(t-1, b, \boldsymbol{x}_{t-1})\, d\boldsymbol{x}_{t-1} \Big\} \\ = \max_{0 \le a \le b} \Big\{ & \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) \Big(\theta(\boldsymbol{x}) + V(t-1, b-\delta) \Big) + \\ & \sum_{\delta=a+1}^{\infty} m(\delta, \boldsymbol{x}) V(t-1, b) \Big\}, \end{aligned} \quad (4)$$

where the first summation[2] is for the situation when winning the auction and the second summation is that when losing. Similarly, the optimal action in state $(t, b, \boldsymbol{x})$ is

$$\begin{aligned} a(t, b, \boldsymbol{x}) = \underset{0 \le a \le b}{\operatorname{argmax}} \Big\{ & \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) \Big(\theta(\boldsymbol{x}) + V(t-1, b-\delta) \Big) + \\ & \sum_{\delta=a+1}^{\infty} m(\delta, \boldsymbol{x}) V(t-1, b) \Big\}, \end{aligned} \quad (5)$$

[2]In practice, the bid prices in various RTB ad auctions are required to be integer.

where the optimal bid action $a(t, b, \boldsymbol{x})$ involves three terms: $m(\delta, \boldsymbol{x})$, $\theta(\boldsymbol{x})$ and $V(t-1, \cdot)$. $V(t, b)$ is derived by marginalizing out $\boldsymbol{x}$:

$$\begin{aligned} V(t, b) = & \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) \max_{0 \le a \le b} \Big\{ \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) \Big(\theta(\boldsymbol{x}) + V(t-1, b-\delta) \Big) \\ & + \sum_{\delta=a+1}^{\infty} m(\delta, \boldsymbol{x}) V(t-1, b) \Big\}\, d\boldsymbol{x} \\ = & \max_{0 \le a \le b} \Big\{ \sum_{\delta=0}^{a} \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x}) \theta(\boldsymbol{x})\, d\boldsymbol{x} + \sum_{\delta=0}^{a} V(t-1, b-\delta) \cdot \\ & \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x})\, d\boldsymbol{x} + V(t-1, b) \sum_{\delta=a+1}^{\infty} \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x})\, d\boldsymbol{x} \Big\} \\ = & \max_{0 \le a \le b} \Big\{ \sum_{\delta=0}^{a} \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x}) \theta(\boldsymbol{x})\, d\boldsymbol{x} + \\ & \sum_{\delta=0}^{a} m(\delta) V(t-1, b-\delta) + V(t-1, b) \sum_{\delta=a+1}^{\infty} m(\delta) \Big\}. \end{aligned} \quad (6)$$

To settle the integration over $\boldsymbol{x}$ in Eq. (6), we consider an approximation $m(\delta, \boldsymbol{x}) \approx m(\delta)$ by following the dependency assumption $\boldsymbol{x} \to \theta \to a \to w$(winning rate) in [32]. Thus

$$\begin{aligned} \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x}) \theta(\boldsymbol{x})\, d\boldsymbol{x} & \approx m(\delta) \int_{\boldsymbol{X}} p_x(\boldsymbol{x}) \theta(\boldsymbol{x})\, d\boldsymbol{x} \\ & = m(\delta) \theta_{\text{avg}}\ , \end{aligned} \quad (7)$$

where θ_{avg} is the expectation of the pCTR θ, which can be easily calculated with historical data. Taking Eq. (7) into Eq. (6), we get an approximation of the optimal value function $V(t, b)$:

$$\begin{aligned} V(t, b) \approx \max_{0 \le a \le b} \Big\{ & \sum_{\delta=0}^{a} m(\delta) \theta_{\text{avg}} + \sum_{\delta=0}^{a} m(\delta) V(t-1, b-\delta) + \\ & \sum_{\delta=a+1}^{\infty} m(\delta) V(t-1, b) \Big\}. \end{aligned} \quad (8)$$

Noticing that $\sum_{\delta=0}^{\infty} m(\delta, \boldsymbol{x}) = 1$, Eq. (5) is rewritten as

$$\begin{aligned} a(t, b, \boldsymbol{x}) = & \underset{0 \le a \le b}{\operatorname{argmax}} \Big\{ \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) \Big(\theta(\boldsymbol{x}) + V(t-1, b-\delta) \Big) - \\ & \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) V(t-1, b) \Big\} \\ = & \underset{0 \le a \le b}{\operatorname{argmax}} \Big\{ \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) \Big(\theta(\boldsymbol{x}) + V(t-1, b-\delta) - V(t-1, b) \Big) \Big\} \\ \equiv & \underset{0 \le a \le b}{\operatorname{argmax}} \Big\{ \sum_{\delta=0}^{a} m(\delta, \boldsymbol{x}) g(\delta) \Big\}, \end{aligned} \quad (9)$$

where we denote $g(\delta) = \theta(\boldsymbol{x}) + V(t-1, b-\delta) - V(t-1, b)$. From the definition, we know $V(t-1, b)$ monotonically increases w.r.t. b, i.e., $V(t-1, b) \ge V(t-1, b')$ where $b \ge b'$. As such, $V(t-1, b-\delta)$ monotonically decreases w.r.t. δ. Thus $g(\delta)$ monotonically decreases w.r.t. δ. Moreover, $g(0) = \theta(\boldsymbol{x}) \ge 0$ and $m(\delta, \boldsymbol{x}) \ge 0$. Here, we care about the value of $g(b)$. (i) If $g(b) \ge 0$, then $g(b') \ge g(b) \ge 0$ where $0 \le b' \le b$, so $m(\delta, \boldsymbol{x}) g(\delta) \ge 0$ where $0 \le \delta \le b$. As a result, in this case, we have $a(t, b, \boldsymbol{x}) = b$. (ii) If $g(b) < 0$, then there must exist an integer A such that $0 \le A < b$ and $g(A) \ge 0$, $g(A+1) < 0$. So $m(\delta, \boldsymbol{x}) g(\delta) \ge 0$ when $\delta \le A$ and

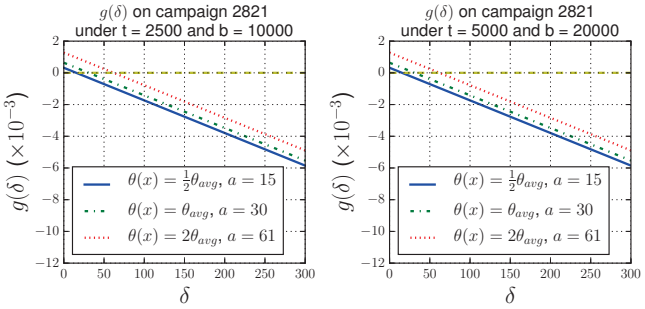

Figure 2: $g(\delta)$ on campaign 2821 as an example.

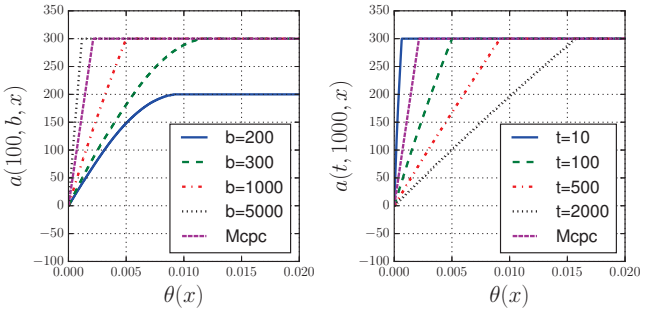

Figure 3: Example derived bidding functions.

Algorithm 1 Reinforcement Learning to Bid

Input: p.d.f. of market price $m(\delta)$, average CTR θ_{avg}, episode length T, budget B
Output: value function $V(t, b)$
1: initialize $V(0, b) = 0$
2: **for** $t = 1, 2, \cdots, T-1$ **do**
3: **for** $b = 0, 1, \cdots, B$ **do**
4: enumerate $a_{t,b}$ from 0 to $\min(\delta_{\max}, b)$ and set $V(t, b)$ via Eq. (8)
5: **end for**
6: **end for**

Input: CTR estimator $\theta(\boldsymbol{x})$, value function $V(t, b)$, current state $(t_c, b_c, \boldsymbol{x}_c)$
Output: optimal bid price a_c in current state
1: calculate the pCTR for the current bid request: $\theta_c = \theta(\boldsymbol{x}_c)$
2: **for** $\delta = 0, 1, \cdots, \min(\delta_{\max}, b_c)$ **do**
3: **if** $\theta_c + V(t_c - 1, b_c - \delta) - V(t_c - 1, b_c) \geq 0$ **then**
4: $a_c \leftarrow \delta$
5: **end if**
6: **end for**

$m(\delta, \boldsymbol{x})g(\delta) < 0$ when $\delta > A$. Consequently, in this case, we have $a(t, b, \boldsymbol{x}) = A$. In conclusion, we have

$$a(t, b, \boldsymbol{x}) = \begin{cases} b & \text{if } g(b) \geq 0 \\ A \quad g(A) \geq 0 \text{ and } g(A+1) < 0 & \text{if } g(b) < 0 \end{cases}. \quad (10)$$

Figure 2 shows examples of $g(\delta)$ on campaign 2821 from iPinYou real-world dataset. Additionally, we usually have a maximum market price $\delta_{\max}$, which is also the maximum bid price. The corresponding RLB algorithm is shown in Algorithm 1.

Discussion on Derived Policy. Contrary to the linear bidding strategies which bids linearly w.r.t. the pCTR with a static parameter [18], such as MCPC and LIN discussed in Section 5.3, our derived policy (denoted as RLB) adjusts its bidding function according to current t and b. As shown in Figure 3, RLB also introduces a linear form bidding function when b is large, but decreases the slope w.r.t. decreasing b and increasing t. When b is small (such as $b < 300$), RLB introduces a non-linear concave form bidding function.

Discussion on the Approximation of $V(t, b)$. In Eq. (7), we take the approximation $m(\delta, \boldsymbol{x}) \approx m(\delta)$ by following the dependency assumption $\boldsymbol{x} \to \theta \to a \to w$(winning rate) in [32] and consequently get an approximation of the optimal value function $V(t, b)$ in Eq. (8). Here, we consider a more general case where such assumption does not hold in the whole feature vector space $\boldsymbol{X}$, but holds within each individual subset. Suppose $\boldsymbol{X}$ can be explicitly divided into several segments, i.e., $\boldsymbol{X} = \sqcup_i \boldsymbol{X}_i$. The segmentation can be built by publisher, user demographics etc. For each segment $\boldsymbol{X}_i$, we take the approximation $m(\delta, \boldsymbol{x}) \approx m_i(\delta)$ where $\boldsymbol{x} \in \boldsymbol{X}_i$. As such, we have

$$\int_{\boldsymbol{X}} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x}) \theta(\boldsymbol{x}) \, d\boldsymbol{x} = \sum_i \int_{\boldsymbol{X}_i} p_x(\boldsymbol{x}) m(\delta, \boldsymbol{x}) \theta(\boldsymbol{x}) \, d\boldsymbol{x}$$
$$\approx \sum_i m_i(\delta) \int_{\boldsymbol{X}_i} p_x(\boldsymbol{x}) \theta(\boldsymbol{x}) \, d\boldsymbol{x} = \sum_i m_i(\delta) (\theta_{\text{avg}})_i P(\boldsymbol{x} \in \boldsymbol{X}_i).$$

4.1 Handling Large-Scale Issues

Algorithm 1 gives a solution to the optimal policy. However, when it comes to the real-world scale, we should also consider the complexity of the algorithm. Algorithm 1 consists of two stages. The first one is about updating the value function $V(t, b)$, while the second stage is about taking the optimal action for current state based on $V(t, b)$. We can see that the main complexity is on the first stage. Thus we focus on the first stage in this section. Two nested loops in the first stage lead the time complexity to $O(TB)$. As for the space complexity, we need to use a two-dimensional table to store $V(t, b)$, which will later be used when taking action. Thus the space complexity is $O(TB)$.

In consideration of the space complexity and the time complexity, Algorithm 1 can only be applied to small-scale situations. When we confront the situation where T and B are very large, which is a common case in real world, there will probably be not enough resource to get the exact value of $V(t, b)$ for every $(t, b) \in \{0, \cdots, T\} \times \{0, \cdots, B\}$.

With restricted computational resources, one may not be able to go through the whole value function update. Thus we propose some parameterized models to fit the value function on small data scale, i.e., $\{0, \cdots, T_0\} \times \{0, \cdots, B_0\}$, and generalize to the large data scale $\{0, \cdots, T\} \times \{0, \cdots, B\}$.

Good parameterized models are supposed to have low deviation to the exact value of $V(t, b)$ for every $(t, b) \in \{0, \cdots, T\} \times \{0, \cdots, B\}$. That means low root mean square error (RMSE) in the training data and good generalization ability.

Basically, we expect the prediction error of $\theta(\boldsymbol{x}) + V(t-1, b-\delta) - V(t-1, b)$ from Eq. (9) in the training data to be low in comparison to the average CTR θ_{avg}. For most (t, b), $V(t, b)$ is much larger than θ_{avg}. For example, if the budget b is large enough, $V(t, b)$ is with the same scale of $t \times \theta_{\text{avg}}$. Therefore, if we take $V(t, b)$ as our target to approximate, it is difficult to give a low deviation in comparison to θ_{avg}. Actually, when calculating $a(t, b, \boldsymbol{x})$ in Eq. (9), we care about the value of $V(t-1, b-\delta) - V(t-1, b)$ rather than $V(t-1, b-\delta)$ or $V(t-1, b)$. Thus here we introduce a new function of value differential $D(t, b) = V(t, b+1) - V(t, b)$ to replace the

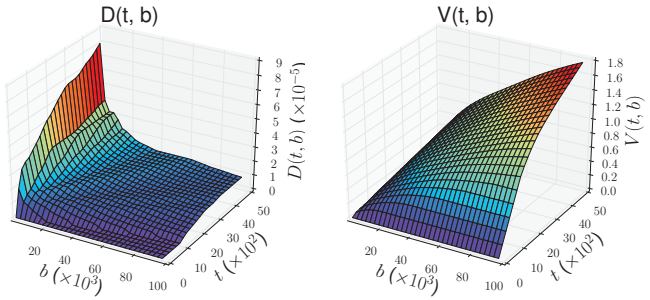

Figure 4: $D(t,b)$ and $V(t,b)$ on campaign 3427.

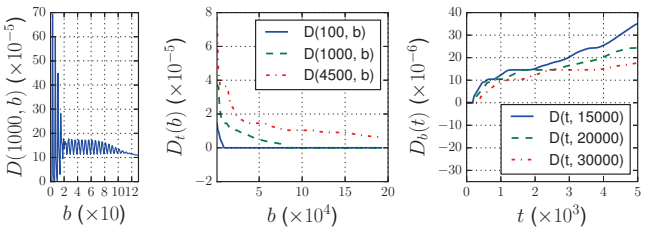

Figure 5: Analysis of $D(t,b)$ on campaign 3386.

role of $V(t,b)$ by

$$V(t-1,b-\delta)-V(t-1,b)=-\sum_{\delta'=1}^{\delta} D(t-1,b-\delta'). \quad (11)$$

Figure 4 illustrates the value of $D(t,b)$ and $V(t,b)$ on the data of an example campaign. In Figure 5, we use the campaign 3386 from iPinYou real-world dataset as an example and show some interesting observations of $D(t,b)$ (other campaigns are similar). At first, for a given t, we consider $D(t,b)$ as a function of b and denote it as $D_t(b)$. $D_t(b)$ fluctuates heavily when b is very small, and later keeps decreasing to 0. Similarly, for a given b, we have $D_b(t)$ as a function of t and it keeps increasing. Moreover, both $D_t(b)$ and $D_b(t)$ are obviously nonlinear. Consequently, we apply the neural networks to approximate them for large-scale b and t.

As a widely used solution [2, 21], here we take a fully connected neural network with several hidden layers as a nonlinear approximator. The input layer has two nodes for t and b. The output layer has one node for $D(t,b)$ without activation function. As such, the neural network corresponds to a non-linear function of t and b, denoted as $\mathrm{NN}(t,b)$.

Coarse-to-fine Episode Segmentation Model. Since the neural networks do not guarantee good generalization ability and may suffer from overfitting, and also to avoid directly modeling $D(t,b)$ or $V(t,b)$, we explore the feasibility of mapping unseen states ($t > T_0$ and $b > B_0$) to acquainted states ($t \le T_0$ and $b \le B_0$) rather than giving a global parameterized representation. Similar to budget pacing, we have the first simple implicit mapping method where we can divide the large episode into several small episodes with length T_0 and within each large episode we allocate the remaining budget to the remaining small episodes. If the agent does not spend the budget allocated for the small episode, it will have more allocated money for the rest of the small episodes in the large episode.

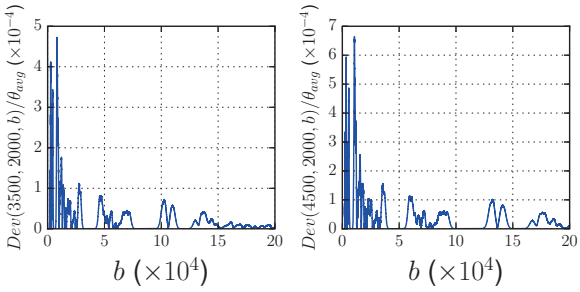

Figure 6: $Dev(t,T_0,b)$ on campaign 3427 as an example.

State Mapping Models. Also we consider explicit mapping methods. At first, because $D_t(b)$ keeps decreasing and $D_b(t)$ keeps increasing, then for $D(t,b)$ where t and b are large, there should be some points $\{(t',b')\}$ where $t' \le T_0$ and $b' \le B_0$ such that $D(t',b') = D(t,b)$ as is shown in Figure 4, which confirms the existence of the mapping for $D(t,b)$. Similarly, $a(t,b,\boldsymbol{x})$ decreases w.r.t. t and increases w.r.t. b, which can be seen in Figure 3 and is consistent with intuitions. Thus the mapping for $a(t,b,\boldsymbol{x})$ also exists. From the view of practical bidding, when the remaining number of auctions are large and the budget situation is similar, given the same bid request, the agent should give a similar bid price (see Figure 2). We consider a simple case that b/t represents the budget condition. Then here we have two linear mapping forms: (i) map $a(t,b,\boldsymbol{x})$ where $t > T_0$ to $a(T_0, \frac{b}{t} \times T_0, \boldsymbol{x})$. (ii) map $D(t,b)$ where $t > T_0$ to $D(T_0, \frac{b}{t} \times T_0)$. Denote $|D(t,b) - D(T_0, \frac{b}{t} \times T_0)|$ as $Dev(t,T_0,b)$. Figure 6 shows that the deviations of the simple linear mapping method are low enough ($< 10^{-3}\theta_{\text{avg}}$).

5. EXPERIMENTAL SETUP

5.1 Datasets

Two real-world datasets are used in our experimental study, namely iPinYou and YOYI.

iPinYou is one of the mainstream RTB ad companies in China. The whole dataset comprises 19.5M impressions, 14.79K clicks and 16.0K CNY expense on 9 different campaigns over 10 days in 2013. We follow [31] for splitting the train/test sets and feature engineering.

YOYI is a leading RTB company focusing on multi-device display advertising in China. YOYI dataset 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.

For experiment reproducibility we publicize our code[3]. In the paper we mainly report results on iPinYou dataset, and further verify our algorithms over the YOYI dataset as supplementary.

[3]The experiment code is available at https://github.com/han-cai/rlb-dp and iPinYou dataset is available at http://data.computational-advertising.org.

5.2 Evaluation Methods

The evaluation is from the perspective of an advertiser's campaign with a predefined budget and lifetime (episode length).

Evaluation metrics. The main goal of the bidding agent is to optimise the campaign's KPI (e.g., clicks, conversions, revenue, etc.) given the campaign budget. In our work, we consider the number of acquired clicks as the KPI, which is set as the primary evaluation measure in our experiments. We also analyze other statistics such as win rate, cost per mille impressions (CPM) and effective cost per click (eCPC).

Evaluation flow. We mostly follow [32] when building the evaluation flow, except that we divide the test data into episodes. Specifically, the test data is a list of records, each of which consists of the bid request feature vector, the market price and the user response (click) label. We divide the test data into episodes, each of which contains T records and is allocated with a budget B. Given the CTR estimator and the bid landscape forecasting, the bidding strategy goes through the test data episode by episode. Specifically, the bidding strategy generates a price for each bid request (the bid price cannot exceed current budget). If the bid price is higher than or equal to the market price of the bid request, the advertiser wins the auction and then receives the market price as cost and the user click as reward and then updates the remaining auction number and budget.

Budget constraints. Obviously, if the allocated budget B is too high, the bidding strategy can simply give a very high bid price each time to win all clicks in the test data. Therefore, in evaluation, budget constraints should not be higher than the historic total cost of the test data. We determine the budget B in this way: $B = \text{CPM}_{\text{train}} \times 10^{-3} \times T \times c_0$, where $\text{CPM}_{\text{train}}$ is the cost per mille impressions in the training data and c_0 acts as the budget constraints parameter. Following previous work [32, 31], we run the evaluation with $c_0 = 1/32, 1/16, 1/8, 1/4, 1/2$.

Episode length. The episode auction number T influences the complexity of our algorithms. When T is high, the original Algorithm 1 is not capable of working with limited resources, which further leads to our large-scale algorithms. For the large-scale evaluation, we set T as 100,000, which corresponds to a real-world 10-minute auction volume of a medium-scale RTB ad campaign. And for the small-scale evaluation, we set the episode length as 1,000. In addition, we run a set of evaluations with $c_0 = 0.2$ and the episode length $T = 200, 400, 600, 800, 1000$ to give a more comprehensive performance analysis.

5.3 Compared Methods

The following bidding policies are compared with the same CTR estimation component which is a logistic regression model and the same bid landscape forecasting component which is a non-parametric method, as described in Section 2:

SS-MDP is based on [1], considering the bid landscape but ignoring the feature vector of bid request when giving the bid price. Although we regard this model as the state-of-the-art, it is proposed to work on keyword-level bidding in sponsored search, which makes it not fine-grained enough to compare with RTB display advertising strategies.

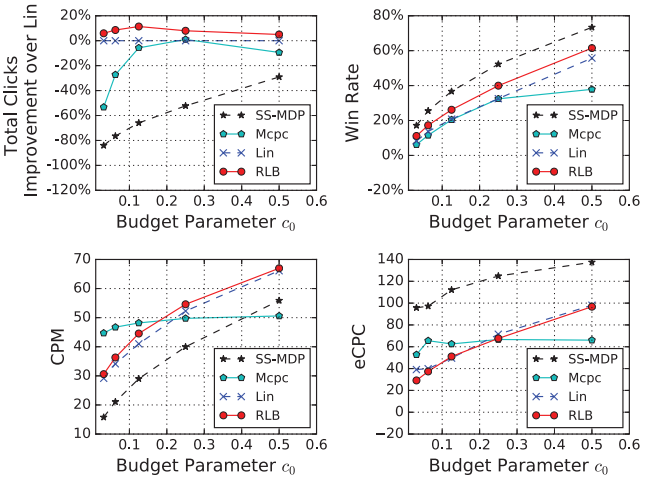

Figure 7: Overall performance on iPinYou under $T = 10^3$ and different budget conditions.

Mcpc gives its bidding strategy as $a_{\text{Mcpc}}(t, b, \boldsymbol{x}) = \text{CPC} \times \theta(\boldsymbol{x})$, which matches some advertisers' requirement of maximum CPC (cost per click).

Lin is a linear bidding strategy w.r.t. the pCTR: $a_{\text{Lin}}(t, b, \boldsymbol{x}) = b_0 \frac{\theta(\boldsymbol{x})}{\theta_{\text{avg}}}$, where b_0 is the basic bid price and is tuned using the training data [18]. This is the most widely used model in industry.

RLB is our proposed model for the small-scale problem as shown in Algorithm 1.

RLB-NN is our proposed model for the large-scale problem, which uses the neural network $\text{NN}(t, b)$ to approximate $D(t, b)$.

RLB-NN-Seg combines the neural network with episode segmentation. For each small episode, the allocated budget is $B_s = B_r/N_r$ where B_r is the remaining budget of the current large episode and N_r is the remaining number of small episodes in the current large episode. Then RLB-NN is run for the small episode. It corresponds to the coarse-to-fine episode segmentation model discussed in Section 4.1.

RLB-NN-MapD combines the neural network with the mapping of $D(t, b)$. That is: (i) $D(t, b) = \text{NN}(t, b)$ where $t \leq T_0$. (ii) $D(t, b) = \text{NN}(T_0, \frac{b}{t} \times T_0)$ where $t > T_0$.

RLB-NN-MapA combines the neural network with the mapping of $a(t, b, \boldsymbol{x})$. That is: $a(t, b, \boldsymbol{x}) = a(T_0, \frac{b}{t} \times T_0, \boldsymbol{x})$ where $t > T_0$. The last two models correspond to the state mapping models discussed in Section 4.1.

6. EXPERIMENTAL RESULTS

In this section we present the experimental results on small- and large-scale data settings respectively.

Table 2: Click improvement of RLB over Lin for each campaign under $T = 10^3$ and different budget conditions.

iPinYou	1/32	1/16	1/8	1/4	1/2
1458	4.66%	3.96%	3.25%	0.21%	1.02%
2259	114.29%	35.29%	9.09%	32.56%	22.22%
2261	25.00%	6.25%	-3.70%	6.82%	0.00%
2821	20.00%	11.86%	27.27%	29.36%	12.97%
2997	23.81%	54.55%	85.26%	13.04%	3.18%
3358	2.42%	3.30%	0.87%	3.02%	0.40%
3386	8.47%	22.47%	13.24%	14.57%	13.40%
3427	7.58%	10.04%	12.28%	6.88%	5.34%
3476	-4.68%	-3.79%	2.50%	5.43%	0.72%
Average	22.39%	15.99%	16.67%	12.43%	6.58%
YOYI	3.89%	2.26%	7.41%	3.48%	1.71%

6.1 Small-Scale Evaluation

The performance comparison on iPinYou dataset under $T = 1000$ and different budget conditions are reported in Figure 7. In the comparison on total clicks (upper left plot), we find that (i) our proposed model RLB performs the best under every budget condition, verifying the effectiveness of the derived algorithm for optimizing attained clicks. (ii) LIN has the second best performance, which is a widely used bidding strategy in industry [18]. (iii) Compared to RLB and LIN, MCPC does not adjust its strategy when the budget condition changes. Thus it performs quite well when $c_0 \geq 1/4$ but performs poorly on very limited budget conditions, which is consistent with the discussion in Section 2. (iv) SS-MDP gives the worst performance, since it is unaware of the feature information of each bid request, which shows the advantages of RTB display advertising.

As for the comparison on win rate, CPM and eCPC, we observe that (i) under every budget condition, SS-MDP keeps the highest win rate. The reason is that SS-MDP considers each bid request equally, thus its optimization target is equivalent to the number of impressions. Therefore, its win rate should be the highest. (ii) LIN and RLB are very close in comparison on CPM and eCPC. RLB can generate a higher number of clicks with comparable CPM and eCPC against LIN because RLB effectively spends the budget according to the market situation, which is unaware of by LIN.

Table 2 provides a detailed performance on clicks of RLB over LIN under various campaigns and budget conditions. Among all 50 settings, RLB wins LIN in 46 (92%), ties in 1 (2%) and loses in 3 (6%) settings. It shows that RLB is robust and significantly outperforms LIN in the vast majority of the cases. Specifically, for 1/8 budget, RLB outperforms LIN by 16.7% on iPinYou data and 7.4% on YOYI data. Moreover, Figure 8 shows the performance comparison under the same budget condition ($c_0 = 0.2$) and different episode lengths. The findings are similar to the above results. Compared to LIN, RLB can attain more clicks with similar eCPC. Note that, in offline evaluations the total auction number is stationary, larger episode length also means smaller episode number. Thus the total click numbers in Figure 8 do not increase largely w.r.t. T.

SS-MDP is the only model that ignores the feature information of the bid request, thus providing a poor overall performance. Table 3 reports in detail the clicks along with the AUC of the CTR estimator for each campaign. We find

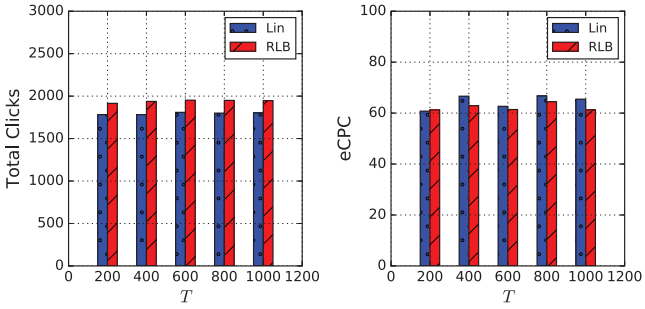

Figure 8: Overall performance comparison on iPinYou under $c_0 = 0.2$ and different T's.

Table 3: Detailed AUC and clicks ($T = 10^3$ and $c_0 = 1/16$).

iPinYou	AUC of $\theta(\boldsymbol{x})$	SS-MDP	MCPC	LIN	RLB
1458	97.73%	42	405	455	473
2259	67.90%	13	11	17	23
2261	62.16%	16	12	16	17
2821	62.95%	49	38	59	66
2997	60.44%	116	82	77	119
3358	97.58%	15	144	212	219
3386	77.96%	24	56	89	109
3427	97.41%	20	178	279	307
3476	95.84%	38	103	211	203
Average	80.00%	37	114	157	170
YOYI	87.79%	120	196	265	271

that when the performance of the CTR estimator is relatively low (AUC $< 70\%$), e.g., campaign 2259, 2261, 2821, 2997, the performance of SS-MDP on clicks is quite good in comparison to MCPC and LIN. By contrast, when the performance of the CTR estimator gets better, other methods which utilize the CTR estimator can attain much more clicks than SS-MDP.

6.2 Large-Scale Evaluation

In this section, we first run the value function update in Algorithm 1 under $T_0 = 10,000$ and $B_0 = \text{CPM}_{\text{train}} \times 10^{-3} \times T_0 \times 1/2$, then train a neural network with the attained data $(t, b, D(t, b))$ (where $(t, b) \in \{0, \cdots, T_0\} \times \{0, \cdots, B_0\}$). Here we use a fully connected neural network with two hidden layers which use tanh activation function. The first hidden layer has 30 hidden nodes and the second one has 15 hidden nodes. Next, we apply the neural network to run bidding under $T = 100,000$ and $B = \text{CPM}_{\text{train}} \times 10^{-3} \times T \times c_0$. In addition, SS-MDP is not tested in this experiment because it suffers from scalability issues and will have a similarly low performance as in the small-scale evaluation.

Table 4 shows the performance of the neural network on iPinYou and YOYI. We can see that the RMSE is relatively low in comparison to θ_{avg}, which means that the neural network can provide a good approximation to the exact algorithm when the agent comes to a state $(t, b, \boldsymbol{x})$ where $(t, b) \in \{0, \cdots, T_0\} \times \{0, \cdots, B_0\}$.

Figure 9 shows the performance comparison on iPinYou under $T = 100,000$ and different budget conditions. We observe that (i) MCPC has a similar performance to that observed in small-scale situations. (ii) For total clicks, RLB-NN performs better than LIN under $c_0 = 1/32, 1/16, 1/8$ and performs worse than LIN under $c_0 = 1/2$, which shows

Table 4: Approximation performance of the neural network.

	iPinYou	YOYI
RMSE ($\times 10^{-6}$)	0.998	1.263
RMSE / θ_{avg} ($\times 10^{-4}$)	9.404	11.954

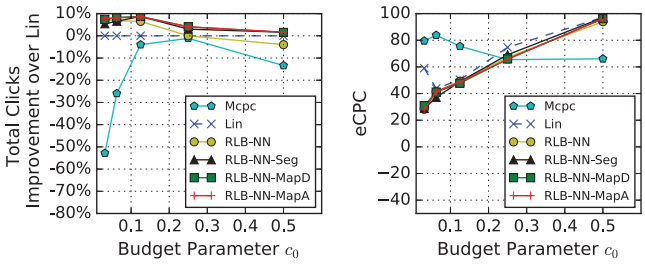

Figure 9: Overall performance on iPinYou under $T = 10^5$ and different budget conditions.

that the generalization ability of the neural network is satisfactory only in small scales. For relatively large scales, the generalization of RLB-NN is not reliable. (iii) Compared to RLB-NN, the 3 sophisticated algorithms RLB-NN-Seg, RLB-NN-MapD and RLB-NN-MapA are more robust and outperform Lin under every budget condition. They do not rely on the generalization ability of the approximation model, therefore their performance is more stable. The results clearly demonstrate that they are effective solutions for the large-scale problem. (iv) As for eCPC, all models except from Mcpc are very close, thus making the proposed RLB algorithms practically effective.

7. ONLINE DEPLOYMENT AND A/B TEST

Our proposed RLB model is deployed and tested in a live environment provided by Vlion DSP. The deployment environment is based on HP ProLiant DL360p Gen8 servers. A 5-node cluster is utilized for the bidding agent, where each node is in CentOS release 6.3, with 6 core Intel Xeon CPU E5-2620 (2.10GHz) and 64GB RAM. The model is implemented in Lua with Nginx.

The compared bidding strategy is Lin as discussed in Section 5.3. The optimization target is click. The two compared methods are given the same budget, which is further allocated to episodes. Unlike offline evaluations, the online evaluation flow stops only when the budget is exhausted. Within an episode, a maximum bid number T is set for each strategy to prevent overspending too much. Specifically, T is mostly determined by the allocated budget for the episode B, previous CPM and win rate: $T = B/\text{CPM}/\text{win rate} \times 10^3$. The possible available auction number during the episode is also considered when determining T. The agent keeps the remaining bid number and budget, which we consider as t and b respectively. Note that the remaining budget may have some deviation due to latency. The latency is typically less than 100ms, which is negligible. We test over 5 campaigns during 25-28th of July, 2016. All the methods share the same previous 7-day training data, and the same CTR estimator which is a logistic regression model trained with FTRL. The bid requests of each user are randomly sent to either method. The overall results are presented in Figure

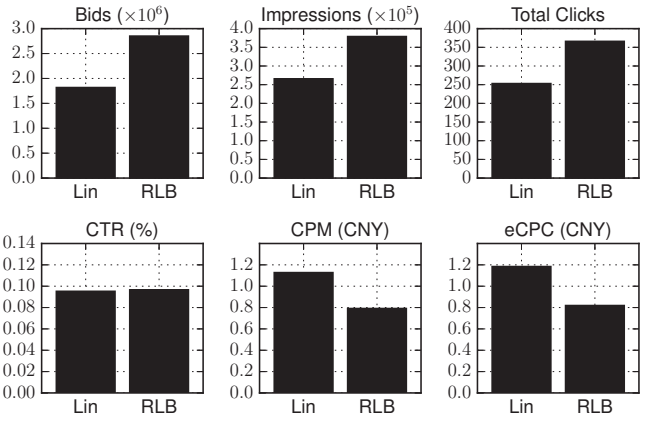

Figure 10: Online A/B testing results.

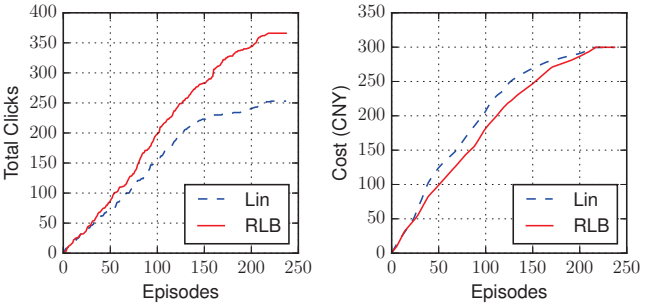

Figure 11: Total clicks and cost increase over episodes.

10, while the click and cost performances w.r.t. time are shown in Figure 11.

From the comparison, we observe the following: (i) with the same cost, RLB achieves lower eCPC than Lin, and thus more total clicks, which shows the cost effectiveness of RLB. (ii) RLB provides better planning than Lin: the acquired clicks and spent budget increase evenly across the time. (iii) With better planning, RLB obtains lower CPM than Lin, yielding more bids and more winning impressions. (iv) With lower CPM on cheap cases, RLB achieves a close CTR compared to Lin, which leads to superior performance. In summary, the online evaluation demonstrates the effectiveness of our proposed RLB model for optimizing attained clicks with a good pacing.

8. CONCLUSIONS

In this paper, we proposed a model-based reinforcement learning model (RLB) for learning the bidding strategy in RTB display advertising. The bidding strategy is naturally defined as the policy of making a bidding action given the state of the campaign's parameters and the input bid request information. With an MDP formulation, the state transition and reward function are captured via modeling the auction competition and user click, respectively. The optimal bidding policy is then derived using dynamic programming. Furthermore, to deal with the large-scale auction volume and campaign budget, we proposed neural network models to fit the differential of the values between two consecutive

states. Experimental results on two real-world large-scale datasets and online A/B test demonstrated the superiority of our RLB solutions over several strong baselines and state-of-the-art methods, as well as their high efficiency to handle large-scale data.

For future work, we will investigate model-free approaches such as Q-learning and policy gradient methods to unify utility estimation, bid landscape forecasting and bid optimization into a single optimization framework and handle the highly dynamic environment. Also, since RLB naturally tackles the problem of budget over- or under-spending across the campaign lifetime, we will compare our RLB solutions with the explicit budget pacing techniques [13, 28].

ACKNOWLEDGMENTS

We sincerely thank the engineers from YOYI DSP to provide us the offline experiment dataset and the engineers from Vlion DSP to help us conduct online A/B tests.

9. REFERENCES

[1] K. Amin, M. Kearns, P. Key, and A. Schwaighofer. Budget optimization for sponsored search: Censored learning in mdps. *UAI*, 2012.

[2] J. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. *NIPS*, pages 369–376, 1995.

[3] O. Chapelle. Modeling delayed feedback in display advertising. In *KDD*, 2014.

[4] Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *KDD*, 2011.

[5] Y. Cui, R. Zhang, W. Li, and J. Mao. Bid landscape forecasting in online ad exchange marketplace. In *KDD*, 2011.

[6] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. Technical report, National Bureau of Economic Research, 2005.

[7] Google. The arrival of real-time bidding, 2011.

[8] G. J. Gordon. Stable function approximation in dynamic programming. In *ICML*, pages 261–268, 1995.

[9] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*, 2010.

[10] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*, 2014.

[11] K. Hosanagar and V. Cherepanov. Optimal bidding in stochastic budget constrained slot auctions. In *EC*, 2008.

[12] V. Krishna. *Auction theory*. Academic press, 2009.

[13] K.-C. Lee, A. Jalali, and A. Dasdan. Real time bid optimization with smooth budget delivery in online advertising. In *ADKDD*, 2013.

[14] K.-c. Lee, B. Orten, A. Dasdan, and W. Li. Estimating conversion rate in display advertising from past erformance data. In *KDD*, 2012.

[15] X. Li and D. Guan. Programmatic buying bidding strategies with win rate and winning price estimation in real time mobile advertising. In *PAKDD*. 2014.

[16] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *KDD*, 2013.

[17] R. J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, and M. Finegold. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *WSDM*, 2014.

[18] C. Perlich, B. Dalessandro, R. Hook, O. Stitelman, T. Raeder, and F. Provost. Bid optimizing and inventory scoring in targeted online advertising. In *KDD*, 2012.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587), 2016.

[20] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. Pac model-free reinforcement learning. In *ICML*, pages 881–888. ACM, 2006.

[21] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[22] G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *ICML*, pages 1017–1024. ACM, 2009.

[23] J. Wang and S. Yuan. Real-time bidding: A new frontier of computational advertising research. In *WSDM*, 2015.

[24] J. Wang, W. Zhang, and S. Yuan. Display advertising with real-time bidding (RTB) and behavioural targeting. *arXiv preprint arXiv:1610.03013*, 2016.

[25] Y. Wang, K. Ren, W. Zhang, J. Wang, and Y. Yu. Functional bid landscape forecasting for display advertising. In *ECML-PKDD*, pages 115–131, 2016.

[26] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[27] W. C.-H. Wu, M.-Y. Yeh, and M.-S. Chen. Predicting winning price in real time bidding with censored data. In *KDD*, 2015.

[28] J. Xu, K.-c. Lee, W. Li, H. Qi, and Q. Lu. Smart pacing for effective online ad campaign optimization. In *KDD*, 2015.

[29] S. Yuan and J. Wang. Sequential selection of correlated ads by pomdps. In *CIKM*, 2012.

[30] S. Yuan, J. Wang, and X. Zhao. Real-time bidding for online advertising: measurement and analysis. In *ADKDD*, 2013.

[31] W. Zhang and J. Wang. Statistical arbitrage mining for display advertising. In *KDD*, 2015.

[32] W. Zhang, S. Yuan, and J. Wang. Optimal real-time bidding for display advertising. In *KDD*, 2014.

[33] W. Zhang, T. Zhou, J. Wang, and J. Xu. Bid-aware gradient descent for unbiased learning with censored data in display advertising. In *KDD*.