Deep Reinforcement Learning for Sponsored Search Real-time Bidding

Jun Zhao*
Alibaba Group
jun.zhaozj@alibaba-inc.com

Guang Qiu
Alibaba Group
guang.qiug@alibaba-inc.com

Ziyu Guan
Xidian University
zyguan@xidian.edu.cn

Wei Zhao*
Xidian University
ywzhao@mail.xidian.edu.cn

Xiaofei He
Zhejiang University
xiaofei.he@cad.zju.edu.cn

ABSTRACT

Bidding optimization is one of the most critical problems in online advertising. Sponsored search (SS) auction, due to the randomness of user query behavior and platform nature, usually adopts keyword-level bidding strategies. In contrast, the display advertising (DA), as a relatively simpler scenario for auction, has taken advantage of real-time bidding (RTB) to boost the performance for advertisers. In this paper, we consider the RTB problem in sponsored search auction, named SS-RTB. SS-RTB has a much more complex dynamic environment, due to stochastic user query behavior and more complex bidding policies based on multiple keywords of an ad. Most previous methods for DA cannot be applied. We propose a reinforcement learning (RL) solution for handling the complex dynamic environment. Although some RL methods have been proposed for online advertising, they all fail to address the "environment changing" problem: the state transition probabilities vary between two days. Motivated by the observation that auction sequences of two days share similar transition patterns at a proper aggregation level, we formulate a robust MDP model at hour-aggregation level of the auction data and propose a control-by-model framework for SS-RTB. Rather than generating bid prices directly, we decide a bidding model for impressions of each hour and perform real-time bidding accordingly. We also extend the method to handle the multi-agent problem. We deployed the SS-RTB system in the e-commerce search auction platform of Alibaba. Empirical experiments of offline evaluation and online A/B test demonstrate the effectiveness of our method.

CCS CONCEPTS

• **Information systems** → **Sponsored search advertising**; • **Computing methodologies** → **Sequential decision making**; **Q-learning**; *Multi-agent reinforcement learning*; *Neural networks*;

---

* Corresponding author.

KEYWORDS

Bidding Optimization; Sponsored Search Auction; Reinforcement Learning

1 INTRODUCTION

Bidding optimization is one of the most critical problems for maximizing advertisers' profit in online advertising. In the sponsored search (SS) scenario, the problem is typically formulated as an optimization of the advertisers' objectives (KPIs) via seeking the best settings of keyword bids [2, 9]. The keyword bids are usually assumed to be fixed during the online auction process. However, the sequence of user queries (incurring *impressions* and *auctions* for online advertising) creates a complicated dynamic environment where a real-time bidding strategy could significantly boost advertisers' profit. This is more important on e-commerce auction platforms since impressions more readily turn into purchases, compared to traditional web search. In this work, we consider the problem, Sponsored Search Real-Time Bidding (SS-RTB), which aims to generate proper bids at the impression level in the context of SS. To the best of our knowledge, there is no publicly available solution to SS-RTB.

The RTB problem has been studied in the context of display advertising (DA). Nevertheless, SS-RTB is intrinsically different from RTB. In DA, the impressions for bidding are concerned with ad placements in publishers' web pages, while in SS the targets are ranking lists of dynamic user queries. The key differences are: (1) for a DA impression only the winning ad can be presented to the user (i.e. a 0-1 problem), while in the SS context multiple ads which are ranked high can be exhibited to the query user; (2) In SS, we need to adjust bid prices on multiple keywords for an ad to achieve optimal performance, while an ad in DA does not need to consider such a keyword set. These differences render popular methods for RTB in DA, such as predicting winning market price [29] or winning rate [31], inapplicable in SS-RTB. Moreover, compared to ad placements in web pages, user query sequences in SS are stochastic and highly dynamic in nature. This calls for a complex model for SS-RTB, rather than the shallow models often used in RTB for DA [31].

One straightforward solution for SS-RTB is to establish an optimization problem that outputs the optimal bidding setting for each impression independently. However, each impression bid is strategically correlated by several factors given an ad, including the ad's budget and overall profit, and the dynamics of the underlying environment. The above greedy strategy often does not lead to a good overall profit [6]. Thus, it is better to model the bidding strategy as a sequential decision on the sequence of impressions in order to optimize the overall profit for an ad by considering the factors mentioned above. This is exactly what reinforcement learning (RL) [17, 21] does. By RL, we can model the bidding strategy as a dynamic interactive control process in a complex environment rather than an independent prediction or optimization process. The budget of an ad can be dynamically allocated across the sequence of impressions, so that both immediate auction gain and long-term future rewards are considered.

Researchers have explored using RL in online advertising. Amin *et al.* constructed a Markov Decision Process (MDP) for budget optimization in SS [1]. Their method deals with impressions/auctions in a batch model and hence cannot be used for RTB. Moreover, the underlying environment for MDP is "static" in that all states share the same set of transition probabilities and they do not consider impression-specific features. Such a MDP cannot well capture the complex dynamics of auction sequences in SS, which is important for SS-RTB. Cai *et al.* developed a RL method for RTB in DA [6]. The method combines an optimized reward for the current impression (based on impression-level features) and the estimate of future rewards by a MDP for guiding the bidding process. The MDP is still a static one as in [1]. Recently, a deep reinforcement learning (DRL) method was proposed in [28] for RTB in DA. Different from the previous two works, their MDP tries to fully model the dynamics of auction sequences. However, they also identified an "environment changing" issue: the underlying dynamics of the auction sequences from two days could be very different. For example, the auction number and the users' visits could heavily deviate between days. A toy illustration is presented in Figure 1. Compared to a game environment, the auction environment in SS is itself stochastic due to the stochastic behaviors of users. Hence, the model learned from Day 1 cannot well handle the data from Day 2. Although [28] proposed a sampling mechanism for this issue, it is still difficult to guarantee we obtain the same environment for different days. Another challenge that existing methods fail to address is the multi-agent problem. That is, there are usually many ads competing with one another in auctions of SS. Therefore, it is important to consider them jointly to achieve better global bidding performance.

Motivated by the challenges above, this paper proposes a new DRL method for the SS-RTB problem. In our work, we captured various discriminative information in impressions such as market price and conversion rate (CVR), and also try to fully capture the dynamics of the underlying environment. The core novelty of our proposed method lies in how the environment changing problem is handled. We solve this problem by observing the fact that statistical features of proper aggregation (e.g. by hour) of impressions have strong periodic state transition patterns in contrast to impression level data. Inspired by this, we design a robust MDP at the hour-aggregation level to represent the sequential decision process in the SS auction environment. At each state of the MDP, rather than

generating bid prices directly, we decide a bidding model for impressions of that hour and perform real-time bidding accordingly. In other words, the robust MDP aims to learn the optimal parameter policy to control the real-time bidding model. Different from the traditional "control-by-action" paradigm of RL, we call this scheme "control-by-model". By this control-by-model learning scheme, our system can do real-time bidding via capturing impression-level features, and meanwhile also take the advantage of RL to periodically control the bidding model according to the real feedback from the environment. Besides, considering there are usually a considerable number of ads, we also design a massive-agent learning algorithm by combining competitive reward and cooperative reward together.

The contribution of this work is summarized as follows: (1) We propose a novel research problem, Sponsored Search Real-Time Bidding (SS-RTB), and properly motivate it. (2) A novel deep reinforcement learning (DRL) method is developed for SS-RTB which can well handle the environment changing problem. It is worth to note that, the robust MDP we proposed is also a general idea that can be applied to other applications. We also designed an algorithm for handling the massive-agent scenario. (3) We deploy the DRL model in the Alibaba search auction platform, one of the largest e-commerce search auction platforms in China, to carry out evaluation. The offline evaluation and standard online A/B test demonstrate the superiority of our model.

2 RELATED WORK

In this section, we briefly review two fields related to our work: reinforcement learning and bidding optimization.

2.1 Reinforcement Learning

In reinforcement learning (RL) theory, a control system is formulated as a Markov Decision Process (MDP). A MDP can be mathematically represented as a tuple $<\mathcal{S}, \mathcal{A}, p, r>$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action space respectively, $p(\cdot)$ denotes the transition probability function, and $r(\cdot)$ denotes the feedback reward function. The transition probability from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$ is $p(s, a, s')$. The reward received after taking action a in state s is $r(s, a)$. The goal of the model is to learn an optimal policy (a sequence of decisions mapping state s to action a), so as to maximize the expected accumulated long term reward.

Remarkably, deep neural networks coupled with RL have achieved notable success in diverse challenging tasks: learning policies to play games [17, 18, 23], continuous control of robots and autonomous vehicles [11, 12, 16], and recently, online advertising [1, 6, 28]. While the majority of RL research has a consistent environment, applying it to online advertising is not a trivial task since we have to deal with the environment changing problem mentioned previously. The core novelty of our work is that we propose a solution which can well handle this problem.

Moreover, when two or more agents share an environment, the performance of RL is less understood. Theoretical proofs or guarantees for multi-agent RL are scarce and only restricted to specific types of small tasks [5, 22, 25]. The authors in [25] investigated how two agents controlled by independent Deep Q-Networks (DQN) interact with each other in the game of Pong. They used the environment as the sole source of interaction between agents. In their
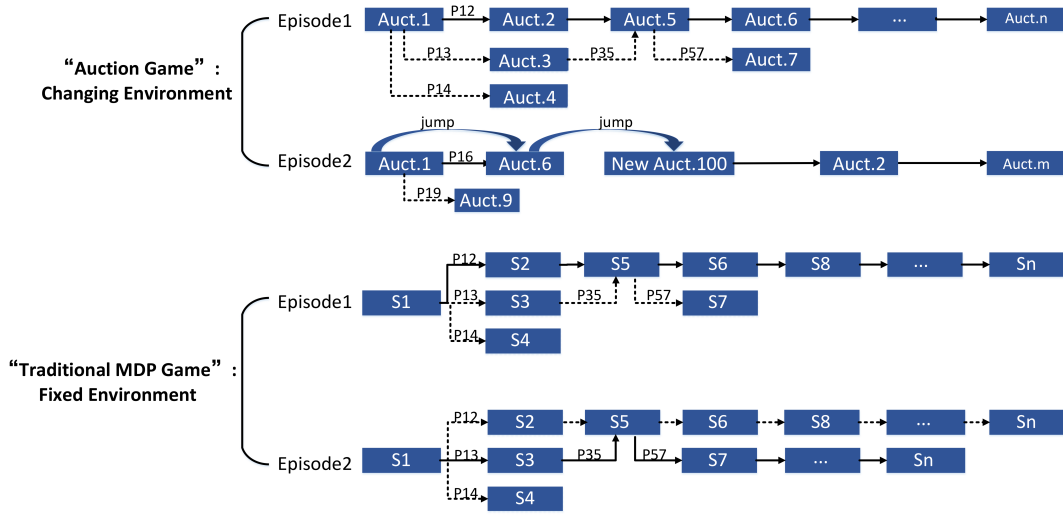
Figure 1: Auction Environment vs Game Environment

study, by changing reward schema from competition to cooperation, the agents would learn to behave accordingly. In this paper, we adopt a similar idea to solve the multi-agent problem. However, the scenario in online advertising is quite different. Not only the agent number is much larger, but also the market environment is much more complex. No previous work has explored using cooperative rewards to address the multi-agent problem in such scenarios.

2.2 Bidding optimization

In sponsored search (SS) auctions, bidding optimization has been well studied. However, most previous works focused on the keyword-level auction paradigm, which is concerned with (but not limited to) budget allocation [3, 9, 19], bid generation for advanced match [4, 8, 10], keywords' utility estimation [2, 13].

Unlike in SS, RTB has been a leading research topic in display advertising (DA) [27, 30]. Different strategies of RTB have been proposed by researchers [6, 7, 14, 29, 31]. In [31], the authors proposed a functional optimization framework to learn the optimal bidding strategy. However, their model is based on an assumption that the auction winning function has a consistent concave shape form. Wu *et al.* proposed a fixed model with censored data to predict the market price in real-time [29]. Although these works have shown significant advantage of RTB in DA, they are not applicable to the SS context due to the differences between SS and DA discussed in Section 1.

In addition to these prior studies, recently, a number of research efforts in applying RL to bidding optimization have been made [1, 6, 28]. In [1], Amin *et al.* combined the MDP formulation with the Kaplan-Meier estimator to learn the optimal bidding policy, where decisions were made on keyword level. In [6], Cai *et al.* formulated the bidding decision process as a similar MDP problem, but taking one step further, they proposed a RTB method for DA by employing the MDP as a long term reward estimator. However, both of the two works considered the transition probabilities as static and failed to capture impression-level features in their MDPs. More

recently, the authors in [28] proposed an end-to-end DRL method with impression-level features formulated in the states and tried to capture the underlying dynamics of the auction environment. They used random sampling to address the environment changing problem. Nevertheless, random sampling still cannot guarantee an invariant underlying environment. Our proposed method is similar to that of [28] in that we use a similar DQN model and also exploit impression-level features. However, the fundamental difference is that we propose a robust MDP based on hour-aggregation of impressions and a novel control-by-model learning scheme. Besides, we also try to address the multi-agent problem in the context of online advertising, which has not been done before.

3 PROBLEM DEFINITION

In this section, we will mathematically formulate the problem of real-time bidding optimization in sponsored search auction platforms (SS-RTB).

In a simple and general scenario, an ad has a set of keyword tuples $\{kwinf_1, kwinf_2, \ldots, kwinf_m\}$, where each tuple $kwinf_i$ can be defined as *<belong_ad, keyword, bidprice>*. Typically, the *bidprice* here is preset by the advertiser. The process of an auction *auct* could then be depicted as: everytime a user u visits and types a query, the platform will retrieve a list of relevant keyword tuples, $[kwinf_1, kwinf_2, \ldots, kwinf_r]^1$, from the ad repository for auction. Each involved ad is then assigned a ranking score according to its retrieved keyword tuple as *bidscore* ∗ *bidprice*. Here, *bidscore* is obtained from factors such as relevance and personalization, etc. Finally, top ads will be presented to the user u.

For SS-RTB, the key problem is to find another *opt_bidprice* rather than *bidprice* for the matched keyword tuples during real-time auction, so as to maximize an ad's overall profit. Since we carry out the research in an e-commerce search auction platform, in the following we will use concepts related to the e-commerce

---

[1]Typically, an ad can only have one keyword tuple (i.e. the most relevant one) in the list

search scenario. Nevertheless, the method is general and could be adapted to other search scenarios. We define an ad's goal as maximizing the purchase amount PUR_AMT_d as income in a day d, while minimizing the cost $COST_d$ as expense in d, with a constraint that the PUR_AMT_d should not be smaller than the advertiser's expected value g. We can formulate the problem as:

$$\begin{aligned} \max \quad & PUR\_AMT_d / COST_d \\ s.t. \quad & PUR\_AMT_d >= g \end{aligned} \quad (1)$$

Observing that PUR_AMT_d has highly positive correlation with $COST_d$, we can change it to:

$$\begin{aligned} \max \quad & PUR\_AMT_d \\ s.t. \quad & COST_d = c \end{aligned} \quad (2)$$

Eq. (2) is equivalent to Eq. (1) when $COST_d$ is positively correlated with PUR_AMT_d (as is the usual case). We omit the proof due to space limitation. The problem we study in this paper is to decide $opt\_bidprice$ in real-time for an ad in terms of objective (2).

4 METHODOLOGY

4.1 A Sketch Model of Reinforcement Learning

Based on the problem we defined in section 3, we now formulate it into a sketch model of RL:

State s: We design a general representation for states as $s = <b, t, \overrightarrow{auct}>$, where b denotes the budget left for the ad, t denotes the step number of the decision sequence, and $\overrightarrow{auct}$ is the auction (impression) related feature vector that we can get from the advertising environment. It is worth to note that, for generalization purpose, the b here is not the budget preset by the advertiser. Instead, it refers to the cost that the ad expects to expend in the left steps of auctions.

Action a: The decision of generating the real-time bidding price for each auction. Here, this is just a sketch model, we leave the details of the decision definition in Section 4.2 for further discussion.

Reward $r(s, a)$: The income (in terms of PUR_AMT) gained according to a specific action a under state s.

Episode ep: In this paper, we always treat one day as an episode.

Finally, our goal is to find a policy $\pi(s)$ which maps each state s to an action a, to obtain the maximum expected accumulated rewards: $\sum_{i=1}^{n} \gamma^{i-1} r(s_i, a_i)$. $\{\gamma^i\}$ is the set of discount coefficients used in a standard RL model [24].

Since the MDP model always formulates an episode of a control system as a sequence of states and actions, we also need to define whether we enter into a same state for the MDP process or not. In online advertising, auction $\overrightarrow{auct}$ is one of the key information in the state representation, which could be viewed as the part of the visual image of a game system. For this reason, to define the auction equivalence is important for the MDP formulation.

However, due to the randomness of user query behavior, one might never see two auctions with exactly the same feature vector. Hence, in previous work of online advertising, there is a fundamental assumption for MDP models: two auctions with similar features can be viewed as the same [1, 6, 28]. Here we provide a mathematical form of this assumption as follows.

ASSUMPTION 1. *Two auction $\overrightarrow{auct}_i$ and $\overrightarrow{auct}_j$ can be a substitute to each other as they were the same if and only if they meet the following condition:*

$$\frac{\|\overrightarrow{auct}_i - \overrightarrow{auct}_j\|^2}{\min(\|\overrightarrow{auct}_i\|^2, \|\overrightarrow{auct}_j\|^2)} < 0.01$$

This kind of substitution will not affect the performance of a MDP-based control system.

However, the above sketch model is defined on the auction level. This cannot handle the environment change problem discussed previously (Figure 1). In other words, given day 1 and day 2, the model trained on ep_1 cannot be applied to ep_2 since the underlying environment changes. In the next, we present our solution to this problem.

4.2 The Robust MDP model

Our solution is inspired by a series of regular patterns observed in real-data. We found that, by viewing the sequences of auctions at an aggregation level, the underlying environments of two different days share very similar dynamic patterns.

We illustrate a typical example in Figure 2, which depicts the number of clicks of an ad at different levels of aggregation (from second-level to hour-level) in Jan. 28th, 2018 and Jan. 29th, 2018 respectively. It can be observed that, the second-level curves does not exhibit a similar pattern (Figure 2(a) and (b)), while from both minute-level and hour-level we can observe a similar wave shape. In addition, it also suggests that the hour-level curves are more similar than those of minute-level. We have similar observations on other aggregated measures.

The evolving patterns of the same measure are very similar between two days at hour-level, indicating the underlying rule of this auction game is the same. Inspired by these regular patterns, we will take advantage of hour-aggregated features rather than auction-level features to formulate the MDP model. Intuitively, if we treat each day as a 24 steps of auction game, an episode of any day would always have the same general experiences. For example, it will meet an auction valley between 3:00AM to 7:00AM, while facing a heavy competitive auction peak at around 9:00AM and a huge amount of user purchases at around 8:00PM. We override the sketch MDP model as follows.

State Transition. The auctions of an episode will be grouped into m ($m = 24$ in our case) groups according to the timestamp. Each group contains a batch of auctions in the corresponding period. A state s is re-defined as $<b, t, \vec{g}>$, where b is the left budget, t is the specific time period, $\vec{g}$ denotes the feature vector containing aggregated statistical features of auctions in time period t, e.g. number of click, number of impression, cost, click-through-rate (CTR), conversion rate (CVR), pay per click (PPC), etc. In this way, we obtain an episode with fixed steps m. In the following, we show that the state transition probabilities are consistent between two days.
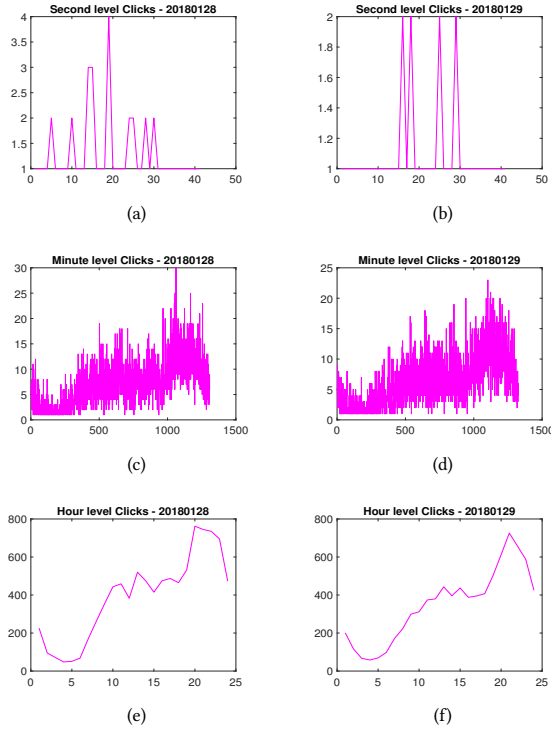
Figure 2: Patterns for Different Aggregation Levels of Clicks

Suppose given state s and action a, we observe the next state s'. We rewrite the state transition probability function $p(s, a, s')$ as

$$
\begin{aligned}
p(s, a, s') &= p(S = s'|S = s, A = a) \\
&= p(S = (b - \delta, t + 1, \vec{g}')|S = (b, t, \vec{g}), A = a) \\
&= p(T = t + 1|T = t, A = a) \\
&\quad \cdot p(B = b - \delta, G = \vec{g}'|B = b, G = \vec{g}, A = a) \\
&= p(B = b - \delta, G = \vec{g}'|B = b, G = \vec{g}, A = a)
\end{aligned}
\tag{3}
$$

where upper case letters represent random variables and δ is the cost of the step corresponding to the action a. Since B is only affected by δ which only depends on the action a, Eq. (3) could be rewritten as:

$$
p(s, a, s') = p(Cost = \delta, G = \vec{g}'|G = \vec{g}, A = a)
\tag{4}
$$

Because $Cost$ is also designed as a feature of $\vec{g}'$, Eq. (4) then becomes:

$$
\begin{aligned}
p(s, a, s') &= p(G = \vec{g}'|G = \vec{g}, A = a) \\
&= \prod_i p(g_i'|g_i, A = a)
\end{aligned}
\tag{5}
$$

where each g_i represents an individual aggregated statistical feature of the feature set (i is the index to the statistical features) and we use the property that features are independent with one another. By inspecting auctions from adjacent days, we have the following empirical observation.

OBSERVATION 1. *Let $g_{i,t}$ be the aggregated value of feature i at step t. When $g_{i,t}$ and the action a_t are fixed, $g_{i,t+1}$ will meet:*

$$
g_{i,t+1} \in [(1 - \eta)\bar{g}, (1 + \eta)\bar{g}]
$$

Where $\bar{g}$ is the sample mean of $g_{i,t+1}$ when $g_{i,t}$ and a_t are fixed, and η is a small value that meets $\eta < 0.03$.

This indicates that the aggregated features change with very similar underlying dynamics within days. When $g_{i,t}$ and a_t are fixed, for any two possible values $\hat{g}$ and $\tilde{g}$ of $g_{i,t+1}$, we have:

$$
\begin{aligned}
\frac{\|\hat{g} - \tilde{g}\|^2}{\min(\|\hat{g}\|^2, \|\tilde{g}\|^2)} &\leq \left(\frac{2\eta}{1 - \eta}\right)^2 \\
&\leq 0.01
\end{aligned}
\tag{6}
$$

According to Assumption 1, we can deem any possible value of $g_{i,t+1}$ to be the same, which means $p(g_i'|g_i, A = a) = 1$. According to (5), finally we get:

$$
p(s, a, s') = 1
\tag{7}
$$

This means the state transition is consistent among different days, leading to a robust MDP.

Action Space. With established state and transition, we now need to formulate the decision action. Most previous works used reinforcement learning to control the bid directly, so the action is to set bid prices (costs). However, applying the idea to our model would result in setting a batch cost for all the auctions in the period. It is hard to derive impression level bid price and more importantly, this cannot achieve real-time bidding.

Instead of generating bid prices, we take a control-by-model scheme: we deploy a linear approximator as the real-time bidding model to fit the optimal bid prices, and we utilize reinforcement learning to learn an optimal policy to control the real-time bidding model. Hence, the action here is the parameter control of the linear approximator function, rather than the bid decision itself.

Previous studies have shown that the optimal bid price has a linear relationship with the impression-level evaluation (e.g. CTR) [15, 20]. In this paper, we adopt the predicted conversion rate (PCVR) as the independent variable in the linear approximator function for real-time bidding which is defined as:

$$
opt\_bidprice = f(PCVR) = \alpha \cdot PCVR
\tag{8}
$$

where the $PCVR$ can be simply learned by any popular regression model (e.g. Logistic regression) on historical data.

To sum up, the robust MDP we propose is modeled as follows:

state	$< b, t, \vec{g} >$
action	set α
reward	PUR_AMT gained in one step
episode	a single day

4.3 Algorithm

In this work, we take a value-based approach as our solution. The goal is to find an optimal policy $\pi(s_t)$ which can be mathematically written as:

$$
\pi(s_t) = \arg\max_a Q(s_t, a_t)
\tag{9}
$$

Where,

$$
Q(s_t, a_t) = \mathbb{E}[R_t|S = s_t, A = a_t]
\tag{10}
$$

$$R_t = \sum_{k=0}^{m-t} \gamma^k r(s_{t+k}, a_{t+k}) \tag{11}$$

R_t in Eq. (11) is the accumulated long term reward that needs to be maximized. $Q(s_t, a_t)$ in Eq. (10) is the standard action value function [21, 24] which captures the expected value of R_t given s_t and a_t. By finding the optimal Q function for each state s_t iteratively, the agent could derive an optimal sequential decision.

By Bellman equation [21, 24], we could get:

$$Q^*(s_t, a_t) = \mathbb{E}[r(s, a) + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})|S = s_t, A = a_t] \tag{12}$$

Eq. (12) reveals the relation of Q^* values between step t and step $t + 1$, where the Q^* value denotes the optimal value for $Q(s, a)$ given s and a. For small problems, the optimal action value function can be exactly solved by applying Eq. (12) iteratively. However, due to the exponential complexity of our model, we adopt a DQN algorithm similar to [17, 18] which employs a deep neural network (DNN) with weights θ to approximate Q. Besides, we also map the action space into 100 discrete values for decision making. Thus, the deep neural network can be trained by minimizing the loss functions in the following iteratively:

$$L_i(\theta_i) = \mathbb{E}[(y_i - Q_{train}(s_t, a_t; \theta_i))^2] \tag{13}$$

Where, $y_i = r(s, a) + \gamma \max_{a_{t+1}} Q_{target}(s_{t+1}, a_{t+1})$

The core idea of minimizing the loss in Eq. (13) is to find a DNN that can closely approximate the real optimal Q function, using Eq. (12) as an iterative solver. Similar to [17], the target network is utilized for stable convergence, which is updated by train network every C steps.

Algorithm 1 DQN Learning

1: **for** *epsode* = 1 to *n* **do**
2:     Initialize replay memory D to capacity N
3:     Initialize action value functions $(Q_{train}, Q_{ep}, Q_{target})$ with weights $\theta_{train}, \theta_{ep}, \theta_{target}$
4:     **for** t = 1 to m **do**
5:         With probability ϵ select a random action a_t
6:         otherwise select $a_t = \arg\max_a Q_{ep}(s_t, a; \theta_{ep})$
7:         Execute action a_t to auction simulator and observe state s_{t+1} and reward r_t
8:         if budget of s_{t+1} < 0, then continue
9:         Store transition (s_t, a_t, r_t, s_{t+1}) in D
10:        Sample random mini batch of transitions (s_j, a_j, r_j, s_{j+1}) from D
11:         **if** $j = m$ **then**
12:             Set $y_j = r_j$
13:         **else**
14:             Set $y_j = r_j + \gamma \arg\max_{a'} Q_{target}(s_{j+1}, a, , \theta_{target})$
15:         **end if**
16:         Perform a gradient descent step on the loss function $(y_j - Q_{train}(s_j, a_j; \theta_{train}))^2$
17:     **end for**
18:     Update θ_{ep} with θ
19:     Every C steps, update θ_{target} with θ
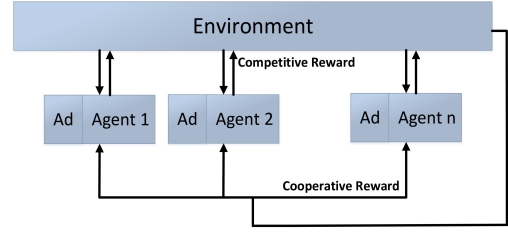20: **end for**

Figure 3: Massive-agent Framework

The corresponding details of the algorithm is presented in Algorithm 1. In addition to train network and target network, we also introduce an episode network for better convergence. Algorithm 1 works in 3 modes to find the optimal strategy: 1) Listening. The agent will record each $<s_t, a_t, r(s_t, a_t), s_{t+1}>$ into a replay memory buffer; 2) Training. The agent grabs a mini-batch from the replay memory and performs gradient descent for minimizing the loss in Eq. (13). 3) Prediction. The agent will generate an action for the next step greedily by the Q-network. By iteratively performing these 3 modes, an optimal policy could be found.

4.4 The Massive-agent Model

The model in Section 4.3 works well when there are only a few agents. However, in the scenario of thousands or millions of agents, the global performance would decrease due to competition. Hence, we proposed an approach for handling the massive-agent problem.

The core idea is to combine the private competitive objective with a public cooperative objective. We designed a cooperative framework: for each ad, we deploy an independent agent to learn the optimal policy according to its own states. The learning algorithm for each agent is very similar to Algorithm 1. The difference is, after all agents made decisions, each agent will receive a competitive feedback representing its own reward and a cooperative feedback representing the global reward of all the agents. The learning framework is illustrated in Figure 3.

4.5 System Architecture

In this subsection, we provide an introduction to the architecture of our system depicted by Figure 4. There are mainly three parts: Data Processor, Distributed Tensor-Flow Cluster, Search Auction Engine.

Data Processor The core component of this part is the simulator, which is in charge of RL exploration. In real search auction platforms, it is usually difficult to obtain precise exploration data. On one hand, we cannot afford to perform many random bidding predictions in online system; on the other hand, it is also hard to generate precise model-based data by pure prediction. For this reason, we build a simulator component for trial and error, which utilizes both model-free data such as real auction logs and model-based data such as predicted conversion rate to generate simulated statistical features for learning. The advantage of the simulator is, auctions with different bid decision could be simulated rather than predicted, since we have the complete auction records for all ads. In particular, we don't need to predict the market price, which is quite hard to predict in SS auction. Besides, for corresponding effected
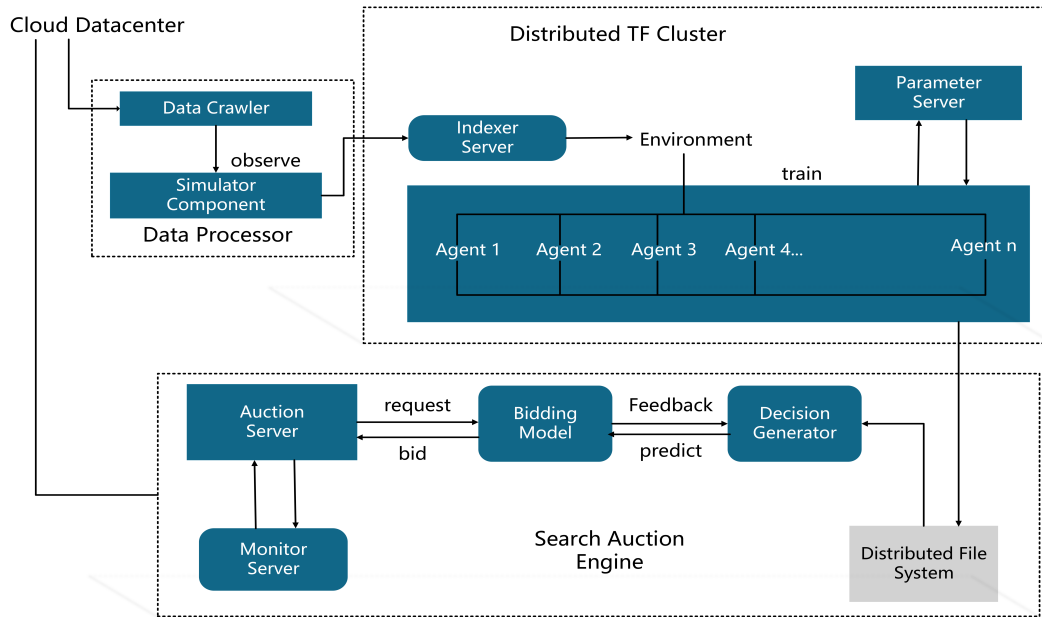
Figure 4: System Achitecture

user behavior which can't be simply simulated (e.g. purchase), we use a mixed method with simulation and PCVR prediction, With this method, the simulator can generate various effects: ranking, clicks, costs, etc.

Distributed Tensor-Flow Cluster This is a distributed cluster deployed on tensor-flow. The DRL model will be trained here in a distributed manner with parameter servers to coordinate weights of networks. Since DRL usually needs huge amounts of samples and episodes for exploration, and in our scenario thousands of agents need to be trained parallelly, we deployed our model on 1000 CPUs and 40 GPUs, with capability of processing 200 billion sample instances within 2 hours.

Search Auction Engine The auction engine is the master component. It sends requests and impression-level features to the bidding model and get bid prices back in real-time. The bidding model, in turn, periodically sends statistical online features to and get optimal policies outputted by the trained Q-network from the decision generator.

5 EXPERIMENTAL SETUP

Our methods are tested both by offline evaluation (Section 6) and via online evaluation (Section 7) on a large e-commerce search auction platform of Alibaba Corporation with real advertisers and auctions. In this section, we introduce the dataset, compared methods and parameter setting.

5.1 Dataset

We randomly select 1000 big ads in Alibaba's search auction platform, which on average cover 100 million auctions per day on the platform, for offline/online evaluation. The offline benchmark dataset is extracted from the search auction log for two days of late December, 2017. Each auction instance contains (but not limited to)

the bids, the clicks, the auction ranking list and the corresponding predicted features such as PCVR that stand for the predicted utility for a specific impression. For evaluation, we use one day collection as the training data and the other day collection for test. Note that we cannot use the test collection directly for test since the bidding actions have already been made therein. Hence, we perform evaluation by simulation based on the test collection. Both collections contain over 100 million auctions. For online evaluation, a standard A/B test is conducted online. Over 100 million auction instances of the 1000 ads are collected one day in advance for training, and the trained models are used to generate real bidding decisions online.

In order to better evaluate the effectiveness of our single-agent model, we also conduct separate experiments on 10 selected ads with disjoint keyword sets, for both offline and online evaluation. Besides, we use the data processor in our system to generate trial-and-error training data for RL methods. In particular, 200 billion simulated auctions are generated by the simulator (described in Section 4.5) for training in offline/online evaluation. The simulated datasets are necessary for boosting the performance of RL methods.

5.2 Compared Methods and Evaluation Metric

The compared bidding methods in our experiments include:

Keyword-level bidding (KB): KB bids based on keyword-level. It is a simple and efficient online approach adopted by many search auction platforms, including the Alibaba's search auction platform. We treat this algorithm as the fundamental baseline of the experiments.

RL with auction-level MDP (AMDP): AMDP optimizes sequence bidding decisions by auction-level DRL algorithm [28]. As in [28], this algorithm samples an auction in every 100 auctions interval as the next state.

Table 1: Hyper-parameter settings

Hyper-parameter	Setting
Target network update period	10000 steps
Memory size	1 million
Learning rate	0.0001 with RMSProp method [26]
Batch size	300
Network structure	4-layer DNN
DNN layer sizes	[15, 300, 200, 100]

RL with robust MDP (RMDP): This is the algorithm we proposed in this paper. RMDP is single-agent oriented, without considering the competition between ads.

Massive-agent RL with robust MDP (M-RMDP): This is the algorithm extended from RMDP for handling the massive-agent problem.

To evaluate the performance of the algorithms, we use the evaluation metric $PUR\_AMT/COST$ under the same $COST$ constraint. It should be noted that, due to Alibaba's business policy, we temporarily cannot expose the absolute performance values for the KB algorithm. Hence, we use relative improvement values with respect to KB instead. This will not affect performance comparison.

5.3 Hyper-parameter Setting

To facilitate the implementation of our method, we provide the settings of some key hyper-parameters in Table 1. It is worth to note that, for all agents we use the same hyper-parameter setting and DNN network structure.

6 OFFLINE EVALUATION

The purpose of this experiment is to answer the following questions. (i) How does the DRL algorithm works for search auction data? (ii) Does RMDP outperform AMDP under changing environment? (iii) Is the multi-agent problem well handled by M-RMDP?

6.1 Single-agent Analysis

The performance comparison in terms of $PUR\_AMT/COST$ is presented in Table 2, where all the algorithms are compared under the same cost constraint. Thus, the performance in Table 2 actually depicts the capability of the bidding algorithm in obtaining more PUR_AMT under same cost, compared to KB. It shows that, on the test data RMDP outperforms KB and AMDP. However, if we compare the performance on the training dataset, the AMDP algorithm is the best since it models decision control at the auction level. Nevertheless, AMDP performs poorly on the test data, indicating serious overfitting to the training data. This result demonstrates that the auction sequences cannot ensure a consistent transition probability distribution between different days. In contrast, RMDP shows stable performance between training and test, which suggests that RMDP indeed captures consistent transition patterns under environment changing by hour-aggregation of auctions.

Besides, Table 2 also shows that for each of the 10 ads RMDP consistently learns a much better policy than KB. Since we use the same hyper-parameter setting and network structure, it indicates

Table 2: Results for the offline single-agent case. Performance is relative improvement of $PUR\_AMT/COST$ compared to KB.

ad_id	AMDP(train)	AMDP(test)	RMDP(train)	RMDP(test)
740053750	334.28%	5.56%	158.81%	136.86%
75694893	297.27%	-4.62%	95.80%	62.18%
749798178	68.89%	8.04%	38.54%	34.14%
781346990	227.91%	-20.08%	79.52%	57.99%
781625444	144.93%	-72.46%	53.62%	38.79%
783136763	489.09%	38.18%	327.88%	295.76%
750569395	195.42%	-15.09%	130.46%	114.29%
787215770	253.64%	-41.06%	175.50%	145.70%
802779226	158.50%	-44.67%	79.07%	72.71%
805113454	510.13%	-8.86%	236.08%	195.25%
Avg.	250.03%	-10.06%	120.01%	98.74%

the performance of our method is not very sensitive to hyper-parameter settings.

Furthermore, the results showed in Table 2 illuminate us a general thought about reinforcement learning in the online advertising field. The power of reinforcement learning is due to sequential decision making. Normally, the more frequently the model can get feedback and adjust its control policy, the better the performance will be (AMDP on the training data). However, in the scenario of progressively changing environment, high frequent feedback information might contain too much stochastic noise. A promising solution for robust training is to collect statistical data with proper aggregation. Nevertheless, the drawback of this technique is sacrificing adjust frequency. Generally, a good approach of DRL for online advertising is actually the consequence of a good trade-off between feedback frequency and data trustworthiness (higher aggregation levels exhibit more consistent transition patterns and thus are more trustworthy).

6.2 Multi-agent Analysis

To investigate how DRL works with many agents competing with each other, we run KB, RMDP and M-RMDP on all the 1000 ads of the offline dataset. In this experiment and the following online experiments, we do not involve AMDP since it has already been shown to perform even worse than KB in the single-agent case.

The averaged results for the 1000 ads are shown in Figure 5. The first thing we can observe is that, the costs of all the algorithms are similar (note that the y-axis of Figure 5(a) measures the ratio to KB's cost), while the $PUR\_AMT/COST$ results are different. It shows that RMDP still outperforms KB, but the relative improvement is not as high as in the single-agent case. Compared to RMDP, M-RMDP shows a prominent improvement in terms of $PUR\_AMT/COST$. This consolidates our speculation that by equipping each agent with a cooperative objective, the performance could be enhanced.

7 ONLINE EVALUATION

This section presents the results of online evaluation in the real-world auction environment of the Alibaba search auction platform with a standard A/B testing configuration. As all the results are collected online, in addition to the key performance metric
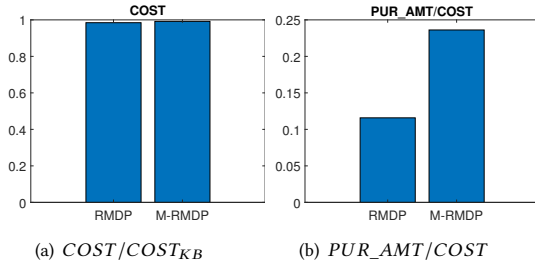
(a) $COST/COST_{KB}$ (b) $PUR\_AMT/COST$

Figure 5: Performance on 1000 ads (offline).

Table 3: Results for the online single-agent case. Performance is relative improvement of RMDP compared to KB.

ad_id	$PUR\_AMT/COST$	CVR	ROI	PPC
740053750	65.19%	60.78%	19.01%	-2.67%
75694893	23.59%	8.83%	12.75%	-11.94%
749798178	4.15%	-7.98%	-0.63%	-11.66%
781346990	41.6%	49.12%	43.86 %	5.33%
781625444	-9.79%	30.95%	7.73%	14.28%
783136763	55.853%	27.03%	52.87%	-18.49%
750569395	2.854%	1.65%	19.60%	8.81%
787215770	21.52%	32.97%	46.94%	-8.61%
802779226	31.44%	46.93%	19.97%	11.78%
805113454	57.08%	78.64%	68.73%	13.74%
Avg.	35.04%	23.11%	21.38%	-5.16%

$PUR\_AMT/COST$, we also show results on metrics including conversion rate (CVR), return on investment (ROI) and cost per click (PPC). These metrics, though different, are related to our objective.

7.1 Single-agent Analysis

We continue to use the same 10 ads for evaluation. The detailed performance of RMDP for each ad is listed in Table 3. It can be observed from the last line that the performance of RMDP outperforms that of KB with an average of 35.04% improvement of $PUR\_AMT/COST$. This is expected as the results are similar to those of the offline experiment. Although the percentages of improvement are different between online and offline cases, this is also intuitive since we use simulated results based on the test collection for evaluation in the offline case. This suggests our RMDP model is indeed robust when deployed to real-world auction platforms. Besides, we also take other related metrics as a reference. We find that there is an average of 23.7% improvement in CVR, an average of 21.38% improvement in ROI and a slight average improvement (5.16%) in PPC (the lower, the better). This means our model could also indirectly improve other correlated performance metrics that are usually considered in online advertising. The slight improvement in PPC means that we help advertisers save a little of his/her cost per click, although not prominent.

7.2 Multi-agent Analysis

A standard online A/B test on the 1000 ads is carried out on Feb. 5th, 2018. The averaged relative improvement results of RMDP and M-RMDP compared to KB are depicted in Table 4. We can see that, similar to the offline experiment, M-RMDP outperforms

Table 4: Results for the online multi-agent case. Performance is relative improvement compared to KB.

Algorithm	$PUR\_AMT/COST$	ROI	CVR	PPC
RMDP	6.29%	26.51%	3.12%	-3.36%
M-RMDP	13.01%	39.12%	12.62%	-0.74%

the online KB algorithm and RMDP in several aspects: i) higher $PUR\_AMT/COST$, which is the objective of our model; ii) higher ROI and CVR, which are related key utilities that advertisers concern. It is again observed that PPC is slightly improved, which means that our model can slightly help advertisers save their cost per click. The performance improvement of RMDP is lower than that in the online single-agent case (Table 3). The reason could be that the competition among the ads affects its performance. In comparison, M-RMDP can well handle the multi-agent problem.

7.3 Convergence Analysis

We provide convergence analysis of the RMDP model by two example ads in Figure 6. Figures 6(a) and 6(c) show the *Loss* (i.e. Eq. (13)) curves with the number of learning batches processed. Figures 6(b) and 6(d) present the PUR_AMT (i.e. our optimization objective in Eq. (2)) curves accordingly. We observe that, in Figures 6(a) and 6(c) the *Loss* starts as or quickly increases to a large value and then slowly converges to a much smaller value, while in the same batch range PUR_AMT improves persistently and becomes stable (Figures 6(b) and 6(d)). This provides us a good evidence that our DQN algorithm has a solid capability to adjust from a random policy to an optimal solution. In our system, the random probability ϵ of exploration in Algorithm 1 was initially set to 1.0, and decays to 0.0 during the learning process. The curves in Figure 6 demonstrate a good convergence performance of RL.

Besides, we observe that the loss value converges to a relatively small value after about 150 million sample batches are processed, which suggests that the DQN algorithm is data expensive. It needs large amounts of data and episodes to find an optimal policy.

8 CONCLUSIONS

In this paper, we proposed a novel research problem: Sponsored Search Real-Time Bidding (SS-RTB) for the purpose of boosting the performance for advertisers. By analyzing the dynamic and stochastic environment of search auction, we formulate a robust control-by-model reinforcement learning framework and deployed a RTB system in a large e-commerce search auction platform. Empirical experiments of offline evaluation and online A/B test demonstrate the superiority of our model and the capability of RL in controlling the complex and dynamical environment in search auction.

For future work, we will investigate more possibility of control-by-model approaches, to see whether there is a general framework that can well control more complex bidding models and furthermore find the optimal form of bidding. Moreover, since trial-and-error makes RL as a data expensive model, we want to search for better methods that could obtain the optimal decision more efficiently.
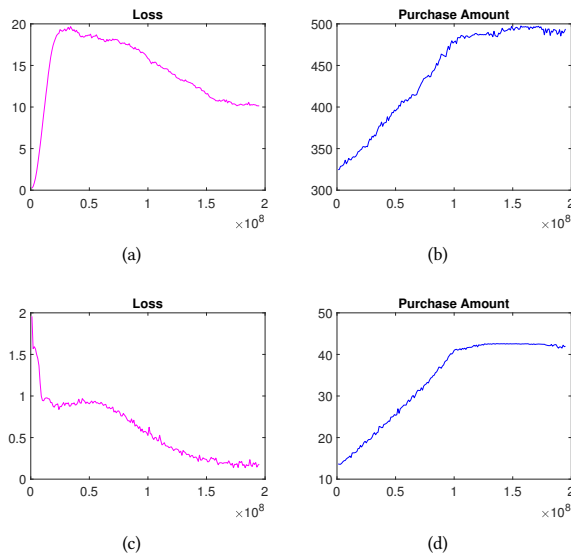
(a)

(b)

(c)

(d)

Figure 6: convergence Performance.

ACKNOWLEDGMENTS

REFERENCES

[1] Kareem Amin, Michael Kearns, Peter Key, and Anton Schwaighofer. 2012. Budget optimization for sponsored search: Censored learning in MDPs. *arXiv preprint arXiv:1210.4847* (2012).

[2] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Kamal Jain, Omid Etesami, and Mohammad Mahdian. 2007. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th international conference on World Wide Web.* ACM, 531–540.

[3] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Mohammad Mahdian, and Amin Saberi. 2005. Multi-unit auctions with budget-constrained bidders. In *Proceedings of the 6th ACM conference on Electronic commerce.* ACM, 44–51.

[4] Andrei Broder, Evgeniy Gabrilovich, Vanja Josifovski, George Mavromatis, and Alex Smola. 2011. Bid generation for advanced match in sponsored search. In *Proceedings of the fourth ACM international conference on Web search and data mining.* ACM, 515–524.

[5] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Systems, Man, and Cybernetics, Part C* 38, 2 (2008), 156–172.

[6] Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo. 2017. Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining.* ACM, 661–670.

[7] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R Devanur. 2011. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1307–1315.

[8] Eyal Even Dar, Vahab S Mirrokni, S Muthukrishnan, Yishay Mansour, and Uri Nadav. 2009. Bid optimization for broad match ad auctions. In *Proceedings of the 18th international conference on World wide web.* ACM, 231–240.

[9] Jon Feldman, S Muthukrishnan, Martin Pal, and Cliff Stein. 2007. Budget optimization in search-based advertising auctions. In *Proceedings of the 8th ACM conference on Electronic commerce.* ACM, 40–49.

[10] Ariel Fuxman, Panayiotis Tsaparas, Kannan Achan, and Rakesh Agrawal. 2008. Using the wisdom of the crowds for keyword generation. In *Proceedings of the 17th international conference on World Wide Web.* ACM, 61–70.

[11] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning.* 2829–2838.

[12] Roland Hafner and Martin Riedmiller. 2011. Reinforcement learning in feedback control. *Machine learning* 84, 1-2 (2011), 137–169.

[13] Brendan Kitts and Benjamin Leblanc. 2004. Optimal bidding on keyword auctions. *Electronic Markets* 14, 3 (2004), 186–201.

[14] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. 2013. Real time bid optimization with smooth budget delivery in online advertising. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising.* ACM, 1.

[15] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 768–776.

[16] Sergey Levine and Pieter Abbeel. 2014. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems.* 1071–1079.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[19] S Muthukrishnan, Martin Pál, and Zoya Svitkina. 2007. Stochastic models for budget optimization in search-based advertising. In *International Workshop on Web and Internet Economics.* Springer, 131–142.

[20] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. 2012. Bid optimizing and inventory scoring in targeted online advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 804–812.

[21] David L Poole and Alan K Mackworth. 2010. *Artificial Intelligence: foundations of computational agents.* Cambridge University Press.

[22] Howard M Schwartz. 2014. *Multi-agent machine learning: A reinforcement approach.* John Wiley & Sons.

[23] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[24] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction.* Vol. 1. MIT press Cambridge.

[25] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

[26] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[27] Jun Wang and Shuai Yuan. 2015. Real-time bidding: A new frontier of computational advertising research. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining.* ACM, 415–416.

[28] Yu Wang, Jiayi Liu, Yuxiang Liu, Jun Hao, Yang He, Jinghe Hu, Weipeng Yan, and Mantian Li. 2017. LADDER: A Human-Level Bidding Agent for Large-Scale Real-Time Online Auctions. *arXiv preprint arXiv:1708.05565* (2017).

[29] Wush Chi-Hsuan Wu, Mi-Yen Yeh, and Ming-Syan Chen. 2015. Predicting winning price in real time bidding with censored data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1305–1314.

[30] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. 2013. Real-time bidding for online advertising: measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising.* ACM, 3.

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