Deep Policy Optimization for E-commerce Sponsored Search Ranking Strategy

Li He
Alibaba Group
hl121322@alibaba-inc.com

Liang Wang
Alibaba Group
liangbo.wl@alibaba-inc.com

Kaipeng Liu
Alibaba Group
zhiping.lkp@taobao.com

Weinan Zhang
Shanghai Jiao Tong University
wnzhang@sjtu.edu.cn

ABSTRACT

In the E-commerce platform, the sponsored search engine not only takes the role of revenue contributor, but also contributes to the long-term growth of the platform by improving user experiences and facilitating advertisers' commercial goals. The key to the satisfactory of the platform, the user and the advertiser is the decision of the list of advertisements to show and the charging prices for the advertisers. In the sponsored search platform, these decisions are made according to a ranking function. In the E-commerce platform, advertisements showing positions under different queries from different users may be associated with advertisement candidates of different bid price distributions and click probability distributions, which requires the ranking functions to be optimized adaptively to the traffic characteristics. In this work, we proposed a generic framework to optimize the ranking functions by deep reinforcement learning methods. Experimental results on a large-scale sponsored search platform (Alibaba sponsored search engine) confirm the effectiveness of the proposed method.

KEYWORDS

Sponsored search, ranking strategy, reinforcement learning

1 INTRODUCTION

Sponsored search is a multi-billion dollar business model which has been widely used in the industrial area [5, 11]. In the commonly employed pay-per-click model, advertisers are charged for users' clicks on their advertisements. The sponsored search platform ranks the advertisements by a ranking function and selects the top ranked ones to present to the users. The prices charged from the advertisers of these presented advertisements are computed by the generalized second price (GSP) auction mechanism [8]. Traditionally, the ranking function is set to be the advertisements' expected revenue to the platform, computed as the product between the advertisers' bidding price and the predicted click-through-rate (CTR) of the user.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
WOODSTOCK'97, July 1997, El Paso, Texas USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

Following this expected revenue based ranking function design, most of the existing methods focus on designing elaborate models to predict the CTR [13, 30]. In this work, instead of designing a CTR prediction model, we try an alternative way to model the ranking function design as a policy optimization problem. Specifically, the proposed method is designed to work for the sponsored search engine of E-commerce platforms like Taobao or Amazon, on which the primary search intention for users are browser and compare between products to help them make the final decision. Hence, the interaction between the users and the platform are sequentially in nature. In this paper, we use the reinforcement learning [26] approach to learn the ranking function parameters as the action (policy) of the learning agent. On the E-commerce platform, we can get the full stack user behavior data including the browsing, click and purchase behaviors. Hence, the rewards of reinforcement learning can be defined according to the benefits of the platform (the revenue), the users (their clickness on the advertisements) and the advertisers (their sales from showing the advertisements).

Since the reinforcement learning method optimize the agents' policy through an explore and exploit strategy, in the existing literature, it is mostly used in games and robotics applications like [17, 24] and [14] where the exploration can be done in a simulated or artificial environment. When utilizing the algorithms in advertising platforms, we need to consider the cost for the exploration. In this paper, we solve this problem by building a simulated learning environment. The simulated environment stores the replay information for each advertisement showing chance, enabling the simulation of the advertisement selection decision and user response under different ranking functions. The user response is computed by reward shaping methods [20]. However, the learnt ranking function may not be optimal due to the data distribution gap between the simulated environment and the online platform. To bridge this gap, we further apply an online policy optimization module to tune the learnt ranking function.

Contributions. In this work, we present our work of learning the ranking function for E-commerce sponsored search platform by policy optimization methods. Our main contributions are summarized as follows:

- We first introduce a policy optimization framework to learn the ranking function based on the full observation of the user behavior on the E-commerce platforms;
- We propose to initialize the ranking function by conducting reinforcement learning in a simulated sponsored search environment. In this way, the reinforcement learning can

explore adequately without hurting the performance of the commercial platform;
- We further present an online policy optimization module to optimize the ranking function adaptively to online data distribution.

2 RELATED WORKS

The auction mechanism has been widely used in Internet companies like Google, Yahoo [8] and Alibaba, and extensively studied in the research area: Bejamin *et al.* in [8] investigate the properties of generalized second price (GSP) and compare it with the VCG [27] mechanism in terms of the equilibrium behavior. In [16], the authors formulate the advertisement allocation problem as a matching problem with budget constraints, and provide theoretical proof that the algorithm can achieve a higher competitive ratio than the greedy algorithm. To solve the ineffectiveness in *next-price* auction, the authors in [1] designs a truth-telling keyword auction mechanism. In [18] followed by [9], the reserved price problem is studied, including its welfare effects and its relation to equilibrium selection criteria. A field analysis on setting reserve prices in sponsored search platforms of Internet companies is presented in [21]. Existing work generally focused on the revenue and efficiency of the auction mechanism and treat the user click-through-rate in a "single-parameter" setting [23]. In our E-commerce platform, in consideration of long term return, instead of maximizing the platform revenue only, we also add the user experience and advertiser utility terms into the ranking function. And since we have the full observation of users' browsing, clicking and purchasing behavior, we can optimize towards the realistic benefits of the platform, users and advertisers.

Recently, policy optimization methods exemplified by reinforcement learning are able to work in the environment with high-dimensional observations by feeding large-scale experience data and training with powerful computational machines [14], and make breakthrough in many different areas. However, most of the existing applications are conducted on simulated non-profitable platforms, where the experience data are easy to acquire and there is no restrict to try any agent policies and training schemes [24, 29]. For online advertising with reinforcement learning, the authors in [2] first propose to tune sponsored search keyword bid price in an MDP framework, where the state space is represented by the auction information, the advertisement campaign's remaining budgets and life-time, while the actions refer to the bid price to set. Then in [3], the authors formulate the sequential bid decision making process in the real-time bidding display advertising. The method is based on assumptions that the winning rate depends only on the bid price and the actual clicks can be well estimated by the predicted CTR. Hence, enabling the best bidding strategy can be computed in an offline fashion. The work most similar to us is [4], which studies the impression allocation problems using reinforcement learning methods in E-commerce platform. However, they make the "single-parameter" assumption in CTR estimation and they only concern the allocation of the impressions whereas in our sponsored search scenario, we also need to optimize the pricing of the impressions. In our work, we model the ranking function as policy and optimize it by conducting reinforcement learning on the offline simulated data, followed by an online model update procedure to make the model consistent with the online data distribution.

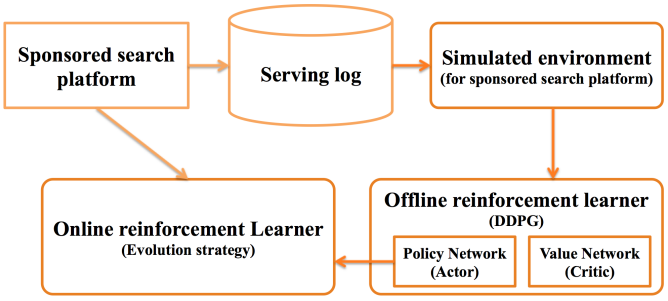

Figure 1: System flow chart.

2.1 System Overview

As shown in Fig. 1, the whole system is composed of three modules: the offline sponsored search environment simulation module, the offline reinforcement learning module and the online optimization module. The environment simulation module is used to simulate the effect caused by changing the ranking function parameters, including re-ranking the advertisement candidates, showing the new top-ranked advertisement, and generating users' response with respect to such changes. To allow adequate exploration, the offline reinforcement learning module collects training data by deploying randomly generated ranking functions on the simulated environment. An actor-critic deep reinforcement learner [15] is then trained on top of these training data. Thereafter, to bridge the gap between the offline simulated data and the online user-advertiser-platform interaction, we build an online learning module to update the model by the online serving results.

3 METHOD

3.1 Ranking Function Learning as Policy Optimization

We define the ranking function prototype here for two reasons. One is to introduce what the ranking function looks like because it determines how to select advertisement from candidates. The other is to specify how we make platform performance goals tunable through this function. The following ranking function is used to compute the rank score for advertisement *ad*

$$\phi(s,a,ad) = \underbrace{f_{a_1}(CTR)\cdot bid}_{\text{platform}} + a_2 \cdot \underbrace{f_{a_3}(CTR,CVR)}_{\text{user}} + a_4 \cdot \underbrace{f_{a_5}(CVR,price)}_{\text{advertiser}} \quad (1)$$

where s represents the search context including the search query, user demographic information and status of advertisement candidates for the advertisement showing chance. Since the traffic status including the bidding price distribution and user engagement on different search context may be different, it is preferable to optimize the ranking function adaptively. *bid* and *price* are the bid price and product price set by the advertiser for the advertisement *ad* on the

current query. CTR and CVR of ad are predicted by the platform. f_{a_i} ($i \in \{1, 3, 5\}$) performs nonlinear monotonic projection on CTR and CVR, and scalar a_2, a_3 are used to balance the weights between the three terms. Because our sponsored search platform charges the advertisers by 'click', the first term $f_{a_1}(CTR) \cdot bid$ can be seen as the expected revenue of the platform. We refer the users' preference to the presented advertisements as their response ratio (CTR and CVR). Hence, the second term indicates the engagement of the users. The third term computes the expected return (expected user purchase amount) of the advertiser by showing the advertisement, which measures the gain of the advertisers. Compared with the ranking function proposed in [12], which is generally used by Google and Yahoo, our ranking function Eq. (1) involves more parameters and commercial factors to consider, which can be used to optimize for more comprehensive commercial goals.

The ranking function is parameterized by a in consideration of the following issues: first, we charge the advertisers according to second price auction mechanism, which is lower than the bid price; second, the three terms may not be the same in numeric scale. The ranking function optimization problem can be formulated as to predict the best parameter a given the search context s as

$$\pi(s) = \arg\max_a R(\phi(s, a)) \tag{2}$$

where $R(\phi(s, a))$ is the reward given the ranking function $\phi(s, a)$. The reward can be defined as the sum of purchase amount, number of click and platform revenue or any weighted combinations of the three terms during a certain period after the ranking function is operated, depending on the platform performance goal. In this work, we formulate the learning of the ranking function as a policy optimization problem with the state defined as s and the action as a. On the E-commerce platform, after posing a query, a user would sequentially browse the search results, click to read the product details and compare them to make purchase decision. The state s would continuously changed according to the existing user behavior and the advertisement showing position changes, and the ranking function should also be changed accordingly to maximize the cumulated reward. The reinforcement learning methods are designed to solve the sequential decision making problem in an explore and exploit manner, which is suitable to solve the ranking function learning problem. However, the performance of the reinforcement learning is guaranteed by adequate exploration, and the exploration may bring uncertainty in the ranking functions' behavior and have performance cost. In this work, we minimize the exploration cost by conducting the reinforcement learning in a simulated sponsored search environment, then update the policy online by an evolution strategy based online policy optimization. In the following sections, we will introduce the detail algorithms of the method.

3.2 Environment Simulation Module

In this work, the environment simulation is performed by replaying the existing advertisement serving processes and making them adjustable in terms of ranking function setting. For each online advertisement showing chance, we store the bidding information and predicted CTR, CVR of all the associated advertisement candidates. According to Eq. (1), the ranking orders and click prices

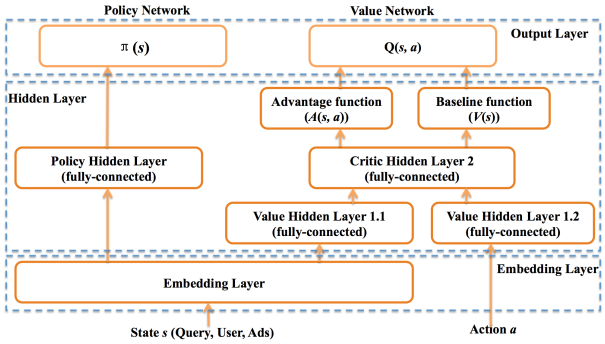

Figure 2: Illustration of the Actor and Critic network architectures used in our work.

for these advertisement candidates can be computed out of these replay information. The reward for showing an advertisement is simulated by the reward shaping approach [20] as the user response (like clicking the advertisement or purchasing the product). For example, if our goal is to get more platform revenue and user clicks, the intermediate reward can be computed as

$$r(s_t, a_t) = CTR \cdot click\_price + \lambda \cdot CTR \tag{3}$$

where λ is a manually tunable parameter to balance between the expectation of more user engagement (click behavior) or more platform revenue. The current state s_t is defined as the search context including the query related features, user behavior features and current advertisement position etc. The action a_t is the ranking function parameter in Eq. (1). To simulate the state transition to s_{t+1}, the user behavior features can be updated by adding the expected user response calculated out of the predicted CTR and CVR. The advertisement positions is updated as the next advertisement showing position.

3.3 Offline Reinforcement Learning for Ranking Function Initialization

Learning from the simulated environment introduced above gives rise to many special requirements for the learning algorithm. First of all, the simulation method above could only generate temporally independent state-action pairs. This is because the state-action sequence of user-platform interaction is tangled. The current user behavior is correlated with the previously presented advertisements and occurred user responses. Hence, the reinforcement learning method should support off-policy learning [7]. Moreover, since the action space $\mathcal{A}$ is continuous (refer to Eq. (2)), it is practical to define the deterministic policy function [25]. Taking these requirements into consideration, we use the Deep Deterministic Policy Gradient (DDPG) learning method in [15] . The method supports off-policy learning, and combines the learning power of deep neural networks with the deterministic policy function property in Actor-Critic architecture.

The architectures of our DDPG model is shown in Fig. 2. We represent all the features of s_t as ID features, and use a shared embedding layer to convert each of these ID features into a fixed-length

dense vector and concatenate them to form the feature representation of s_t. For the policy network, we use clip method as [17] to clip the output into a valid range to avoid over-learning. For the value network, we add a dueling network architecture [28] before the output layer which divides the value function ($Q(s,a)$) into the sum of a state value function ($V(s)$) and a state-dependent action advantage function ($A(s,a)$), such that $Q(s,a) = V(s) + A(s,a)$. According to the insight of [28], the dueling architecture makes the learning of the value network efficient by identifying the highly rewarded states and the states where the selected actions do not affect the rewards much. To train the DDPG model, we employ the asynchronous training strategy [19] using the (s, a, r, s') tuples generated by simulation system .

3.4 Online Policy Optimization for Ranking Function Updating

The offline simulated environment is still inconsistent with the real online environment due to the dynamic data distribution and sequential correlation between the continuous user behavior. This inconsistency poses the online updating requirement for the learned ranking function. However, directly using the asynchronous training framework in Section 3.3 is not proper due to the speciality of the online updating: (1) the data distribution is different, where the online rewards are sparse and discrete; (2) there are latency in reward collection. Regarding to these specialities, we introduce the evolution strategy [22] to update the parameters of the policy model. We perform the following steps to online update the policy networks $\pi_{\theta_\pi}(s_t)$: (i) stochastically perturb the parameters θ_π by a Gaussian noise generator with zero mean and variance σ^2. Denote the set of n perturbed parameters as $\Theta_{\pi,\epsilon} = \{\theta_\pi + \epsilon_1, \theta_\pi + \epsilon_2, ..., \theta_\pi + \epsilon_n\}$. (ii) Hash the the online traffic into bins according to dimensions like user ID and IP address. For each parameter $\theta_{\pi,i} \in \Theta_{\pi,\epsilon}$, we deploy a policy network $\pi_{\theta_{\pi,i}}(s_t)$ on a traffic bin and get the reward according to Eq. (3) as the weighted sum of platform revenue and the click number in this bin $R_i = total\_click\_price + \lambda \cdot click\_number$. However, in reality, the number of advertisement showing should not be exactly the same for each bin. We compute the relative value of the reward by dividing it with the number of served advertisements as $\overline{R}_i = \frac{R_i}{served\_ad\_number}$. (iii) Update the parameter θ_π by the weighted sum of the perturbations as

$$\theta'_\pi = \theta_\pi + \eta \frac{1}{n\sigma} \sum_{i=1}^{n} \overline{R}_i \epsilon_i \tag{4}$$

where η is the learning rate.

The evolution strategy based method has several merits under our scenario. First of all, it is derivative-free. Since the rewards are discrete, it is hard to compute the gradient from the reward to the policy network parameters. Secondly, by fixing the seed of the random number generator, we just need to communicate the reward (a scalar) between the policy networks in local traffic bins and the central parameter servers. Thirdly, the method does not have intermediate reward requirement due to the homogeneity of these online traffic bins. Thus it can be deployed to optimize conversion related performance.

4 EXPERIMENTAL RESULTS

We conduct experiments on an E-commerce sponsored search platform, which serves several billions of advertisements to hundreds of millions of users per-day. There are about tens of millions of advertisements covering diverse categories.

To fully study the effectiveness of the proposed ranking strategy optimization method, both analytical experiments on offline data and empirical experiments by deploying the learned ranking strategy online are carried out. On the platform, the search results are presented in a streaming fashion, and the advertisements are allowed to be shown on fixed positions within the streamed content. Since the search results are tangled with the advertisements, besides the platform revenue, one important issue we need to deal with is the user experience. In the experiments, we set the immediate reward r to be

$$r = click\_price \cdot is\_click + \lambda \cdot is\_click \tag{5}$$

where *click_price* is the amount we charge the advertisers according to generalized second price auction, and *is_click* is a binary number indicating whether the advertisement is clicked (1) or not (0). λ is manually set according to the average *click_price* to balance between the platform revenue goal and the user experience goal. But because we test our model on a small percentage of traffic online (2% traffic), the purchase amount is highly varied according to our observation. In the current experiments, we do not show the purchase optimized results and leave it as a future work when we ramp up our test traffic amount.

4.1 Experiments on Offline Data

In the offline experiments, we study the convergence property of the proposed method and the effect of using different architectures and different hyper parameters on the speed of convergence. We employ an analytical method to verify whether the proposed method can converge to the 'right' ranking function. In the experiment, a simple state representation (only query + advertisement position) is utilized such that from the simulated data, it is computational feasible to perform brute force search to find the best parameters of the ranking function. The brute force method proceeds by uniformly sampling the ranking function parameters in a at a fixed step size for each replay sample, computing the rewards (refer Eq. (5)), and finding the best parameter a^* from these samples according to the aggregated rewards. For training the reinforcement learning model, we encode both the query IDs and advertisement position IDs into 8-dimension embedding vectors. As a result, our embedding layer consists of a 16-dimension feature vector. For the critic hidden layer, we utilize two full-connection units, each of which has 500 nodes, and ELU [6] as the activation function. We use the same settings for the actor hidden layer except using 100 nodes in each layer. At the training stage, we use an exponential decaying learning rate for network parameter. We set the initialized value to 1.0e-4 and decay it by 0.96 for each $1M$ steps. The target network learning rate (τ) and regularization loss penalty factor are set to be 0.01 and 1.0e-5 respectively. The λ is set to be the average of the *click_price* calculated out of the data log. Experimental results are presented in Fig. 3. The performance of the proposed method is measured by the squared error between the learnt ranking function parameters

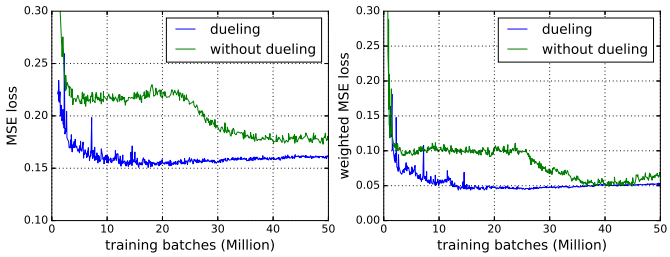

Figure 3: Comparison of the convergence speed of training by utilizing different network architectures. Averaged squared error (left) and advertisement showing number weighted squared error (right) differences between strategy parameters of DDPG and the searched results. '(without) dueling' is the trained result (not) using the dueling network.

Table 1: Hyperparameter setup of Fig. 4.

ID	Learning rate	Regularization	Batch size
decay	exponential decay	1.0e-5	50k
low	**1.0e-5**	1.0e-5	50k
high	**1.0e-4**	1.0e-5	50k
batch size	exponential decay	1.0e-5	**10k**
regular	exponential decay	**1.0e-3**	50k

and the 'best' parameters found by brute force method. From the results, we can see, the proposed method could converge gradually to the best ranking function as the training process goes on. In Fig. 3, we also compare the results of using the dueling architecture (annotated by 'dueling' in the figure) and not ('without dueling') to confirm the performance improvement brought by the dueling architecture. It can be seen that, the dueling architecture improves the convergence speed dramatically. The intuition behind the results is the dueling architecture could help remove the reward variance of the same action under different states by $V(s)$, and guide the action-value network $A(s, a)$ to focus more on differentiating between different actions. As a result, the policy network learning is accelerated.

In Fig. 4, we evaluate the influences of different hyper parameters for training the model, including the learning rate, regularization penalties and batch sizes. The parameter setup is listed in Table 1. As can be observed, (1) larger batch size (compare 'decay' and 'batch size') and lower regularization penalty (compare 'decay' and 'regular') make the learning method converge more closely to the optimal solution. This is because there is strong variance in the rewards. Larger batch size helps to reduce the variance in the batched training data, and lower regularization penalty allows a larger searching space for variables; (2) decaying learning rate ('decay' , 'low' and 'high') makes the learning method converge more smoothly and quickly. A decayed learning rate inherits the merit of higher learning rate by accelerating the convergence at the beginning of training, and benefits from the lower learning rate to smooth the learning in later training rounds.

It should be noted that, when deploying the model online, we would use far more features to encode the sequentially changed

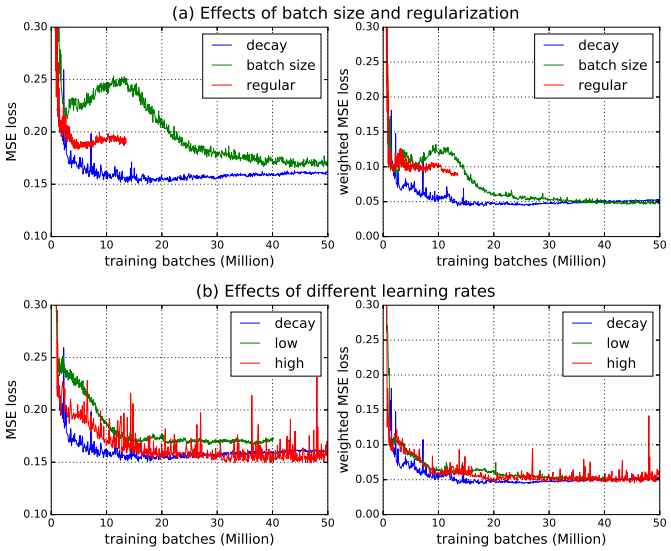

Figure 4: Comparison of the convergence speed of training by using different hyper parameters. The hyper parameter description is shown in Table 1.

search context instead of just two simple features as in above analytical case. Hence, it is computational infeasible to use brute-force method to find a mapping between the high-dimensional search context space to the continuous action space (ranking function parameters).

4.2 Online Serving Experiments

In this section, we present the experimental results of conducting a bucket experiment on Alibaba sponsored search engine. The sponsored search platform charges the advertisers for the click on their advertisements according to GSP auction mechanism. We split a small percentage (about 2%) of traffic from the whole online traffic by hashing the user IDs, IP addresses, etc., and deploy the learned ranking function online for advertisement selection and pricing. The following business metric are measured to see the improvement brought by the proposed method. (1) Revenue-Per-Mille (RPM): the revenue generated per thousand impressions; (2) Price-Per-Click (PPC): the average price per-click determined by the auction; (3) Click-through-rate (CTR). We employ the three business metrics because we are optimizing towards the platform revenue and user experience as in Eq. (5). RPM is determined by the product of CTR and PPC. From the change of CTR and PPC, we can also induce the improvements of the advertisers' saling efficiency, i.e. increase in CTR and decrease in PPC means the advertisers can attract more customers by less spending on advertisement serving.

In this work, a new ranking function is proposed by adding the terms reflecting the user engagement and the advertisers' gain, and a offline reinforcement learning method is presented for optimizing the ranking funciton. In the online experiments, we want to verify the performance improvement brought by the new ranking function and the improvement from the reinforcement learning method. To evaluate the effectiveness of the new ranking function,

we compare the proposed ranking function with the one proposed by Lahaie and McAfee [12]. The ranking function in [12] has been used by many companies and proved to be efficient in online advertising auctions. In the experiment, we use this ranking function as the baseline and set its parameter (the exponential term) by brute force searching in the same manner as Section 4.1. For the proposed method, we use both the ranking functions learned by the brute force method in Section 4.1 and the ranking function learned by the reinforcement learning method. It should be noted that, in this experiment, we use the full set of features introduced in Section 3.3 as state features for reinforcement learning, instead of the simple representation employed in Section 4.1. The comparison results on two continuous days' data are shown in Table 2. As is observed, compared to the method [12], our ranking function learned by the brute force method is capable of delivering 2.5% of RPM growth with 1.1% of CTR increase and 1.4% of PPC increase. This is because the proposed function has more parameters to tune and possesses more space for performance improvement. For the ranking function learned by offline reinforcement learning method, we observe a 2.5% RPM growth which is arisen majorly from the CTR increase (2.0% of CTR gain). We interpret the difference between the reinforcement learning method and 'brute force' method by the fact that the reinforcement learning method dose not converge to the exact value of brute force method, and it use a more elaborate set of features to represent states. From the business side, we can find that the new ranking function can improve more user engagement by attracting more user clickness on the presented advertisements. For the platform, the increase of RPM brings in more efficiency in platform profit, and for the advertisers, the RPM growth is driven by the CTR increase with little increase in PPC (for the learned ranking function), this means, the advertisers only need to pay a little more money to attract more potential buyers.

Section 3.4 introduces the online evolution strategy method for optimizing the policy networks based on online data. In this experiment, we evaluate the online performance changes in seven continuous days to confirm the performance increases brought by online learning method. We still use the method of McAfee [12] as baseline. To explore strategy actions in $\pi_{\theta_\pi}(s)$, we add a gaussian noise $G(0, \delta^2)$ with mean 0 and variance $\sigma^2 = 0.05$ to the parameters θ_π of $\pi_{\theta_\pi}(s)$. We split the traffic of the test bucket into 100 splits and apply each perturbed policy networks to one of them. The buckets and the user feedback history are collected from the data logs to compute the updates with learning rate $\eta = 0.05$ in Eq. (4). The average performance of the test bucket are shown in Fig. 5. As can be seen, all the three business metrics improve during the days. Compared with the baseline ranking function [12] (whose parameter stays unchanged during the days), the RPM grows from 2.5% to 4.5%, CTR grows from 2.1% to 2.9%, indicating the effectiveness of the online updating method. We also find the PPC grows because there is no constraints added on it. In the future work, we will try to add constraints on PPC to generate more return for advertisers.

5 CONCLUSIONS

As a commercial sponsored search platform, besides the intermediate platform revenue, the users' engagement and the advertisers' return are also important to the long-term profit of the commercial platform. In this work, we design a new ranking function by incorporating these factors together. However, these additional terms increase the complexity of the ranking function. So, we propose a reinforcement learning framework to optimize the ranking function. To allow adequate exploration without hurting the performance of the commercial platform, we propose to initialize the ranking function by an offline learning procedure conducted in a simulated sponsored search environment, followed by an online learning module which updates the model adaptively to online data distribution. Experimental results confirms the effectiveness of the proposed method.

In the future, we will focus on the following direction: Sequential user behavior simulation. The environment simulation method introduced in Section 3.2 is limited to one time advertisement serving without considering the correlation between sequential user behaviors. We plan to try generative models like GAN [10] to model the continuous user behaviors.

Table 2: Experimental results comparing the performance of the ranking function, the brute force searched ranking function and the one learned by reinforcement learning method in Section.

Metrics (%)	Δ_{rpm}	Δ_{ctr}	Δ_{ppc}
McAfee [12]	0.00	0.00	0.00
brute force	2.55	1.12	1.45
offline learning	2.52	2.08	0.26

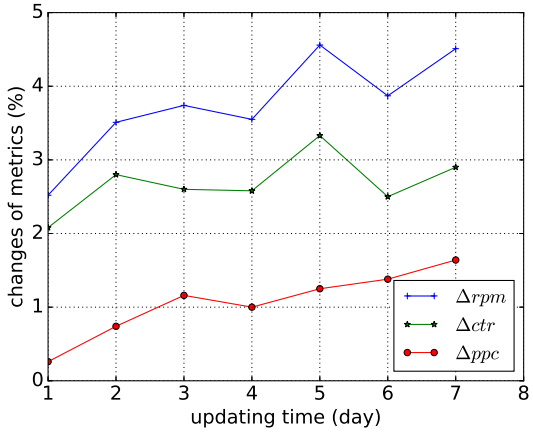

Figure 5: Experimental results illustrating the business metric changes during the online update of the proposed method in Section 3.4.

REFERENCES

[1] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. 2006. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM conference on Electronic commerce.*

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

[3] 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 WSDM.* 661–670.

[4] Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. 2017. A deep reinforcement learning framework for allocating buyer impressions in e-commerce websites. *arXiv preprint arXiv:1708.07607* (2017).

[5] Deepayan Chakrabarti, Deepak Agarwal, and Vanja Josifovski. 2008. Contextual advertising by combining relevance with click feedback. In *Proceedings of WWW*. 417–426.

[6] Djork-ArnÃĺ Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *Computer Science* (2015).

[7] Thomas Degris, Martha White, and Richard Sutton. 2012. Off-Policy Actor-Critic. In *Proceedings of International Conference on Machine Learning*.

[8] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2007. Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords. *American Economic Review* 97, 1 (2007), 242–259.

[9] Benjamin Edelman and Michael Schwarz. 2010. Optimal auction design and equilibrium selection in sponsored search auctions. *The American Economic Review* 100, 2 (2010), 597–602.

[10] Ian Goodfellow, Jean Pouget-Abadie, and Mehdi Mirza *et. al.* 2014. Generative adversarial nets. In *Proceedings of NIPS*. 2672–2680.

[11] Cheng Haibin and Erick Cantu-Paz. 2010. Personalized click prediction in sponsored search. In *Proceedings of WSDM*.

[12] Sébastien Lahaie and R Preston McAfee. 2011. Efficient Ranking in Sponsored Search. (2011), 254–265.

[13] Wei Li, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao, and Rong Jin. 2010. Exploitation and exploration in a performance based contextual advertising system. In *Proceedings of the ACM SIGKDD*. 27–36.

[14] Yuxi Li. 2017. Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1701.07274* (2017).

[15] Timothy P. Lillicrap, Jonathan J. Hunt, and Alexander Pritzel *et. al.* 2015. Continuous Control with Deep Reinforcement Learning. In *Proceedings of ICLR*. 1–14.

[16] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. 2007. Adwords and Generalized Online Matching. *J. ACM* 54, 5 (2007).

[17] Volodymyr Mnih, Koray Kavukcuoglu, and David Silver *et al.* 2015. Human-level control through deep reinforcement learning. *Nature* 7540 (2015), 529–533.

[18] Roger Myerson. 1981. Optimal Auction Design. *Mathematics of Operations Research* 6, 1 (1981), 58–73.

[19] Arun Nair, Praveen Srinivasan, and Sam Blackwell *et. al.* 2015. Massively Parallel Methods for Deep Reinforcement Learning. (2015).

[20] Andrew Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of International Conference on Machine Learning*, Vol. 99.

[21] Michael Ostrovsky and Michael Schwarz. 2011. Reserve prices in internet advertising auctions: A field experiment. (2011).

[22] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[23] Weiran Shen and Pingzhong Tang. 2017. Practical versus Optimal Mechanisms. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. 78–86.

[24] D. Silver, A. Huang, and C. J. Maddison *et. al.* [n. d.]. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 ([n. d.]), 484–489.

[25] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proceedings of ICML*. 387–395.

[26] Richard Sutton and Andrew Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge: MIT Press.

[27] William Vickrey. 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance* 16, 1 (1961), 8–37.

[28] Ziyu Wang, Tom Schaul, and Matteo *et. al.* Hessel. 2015. Dueling network architectures for deep reinforcement learning. (2015), 1995–2003.

[29] Y. Wu and Y. Tian. 2017. Training Agent for First-person Shooter Game with Actor-Critic Curriculum Learning. In *Proceedings of ICLR*. 1–8.

[30] H. Yu, C. Hsieh, and C. Lin K. Chang. 2012. Large linear classification when data cannot fit in memory. *ACM Transactions on Knowledge Discovery from Data* 4 (2012), 23–30.