Budget Constrained Bidding by Model-free Reinforcement Learning in Display Advertising

Di Wu, Xiujun Chen, Xun Yang, Hao Wang, Qing Tan, Xiaoxun Zhang, Kun Gai
Alibaba Group*

Abstract

Real-time bidding (RTB) is almost the most important mechanism in online display advertising, where proper bid for each page view plays a vital and essential role for good marketing results. Budget constrained bidding is a typical scenario in RTB mechanism where the advertisers hope to maximize total value of winning impressions under a pre-set budget constraint. However, the optimal strategy is hard to be derived due to complexity and volatility of the auction environment. To address the challenges, in this paper, we formulate budget constrained bidding as a Markov Decision Process. Quite different from prior model-based work, we propose a novel framework based on model-free reinforcement learning which sequentially regulates the bidding parameter rather than directly producing bid. Along this line, we further innovate a reward function which deploys a deep neural network to learn appropriate reward and thus leads the agent to deliver the optimal policy effectively; we also design an adaptive ϵ-greedy strategy which adjusts the exploration behaviour dynamically and further improves the performance. Experimental results on real dataset demonstrate the effectiveness of our framework.

1 Introduction

In recent years, online display advertising is attracting more and more attention, and Real-Time Bidding (RTB) [Wang and Yuan, 2015; Yuan *et al.*, 2013] has been playing an important role. In particular, one slot display advertising scenario, in which only the highest bidder wins the opportunity, is quite common across different advertising platforms. A typical optimization goal for advertisers is to maximize the total value of winning impressions under a budget constraint, and how to bid properly is critical. Budget constrained bidding is an automated bidding technique to address this need, and advertisers can simply set their optimization goal and budget constraint, then the platform will bid on behalf of advertisers to get the best return. This bidding strategy is widely provided by advertising platforms such as Google and Facebook, and it has improved the marketing efficiency significantly.

Budget constrained bidding is formally defined and abstracted as a knapsack problem [Lin *et al.*, 2016], and [Zhang *et al.*, 2014; Zhang *et al.*, 2016] give a solution with the optimal bid equalling $impression\_value/\lambda$ in second-price auctions. This bidding formula has a very straightforward interpretation: the better impression value is, the higher bid is offered, and the impression value is scaled by a parameter λ to get the bid. Impression value is closely related to the optimization goals, such as maximizing clicks, or optimizing conversions, and usually click through rate [Qu *et al.*, 2016; McMahan *et al.*, 2013] and conversion rate [Lee *et al.*, 2012] need to be estimated. The scaling parameter λ is critical: high λ may cause the bid lower than the optimal one, so budget may not be fully used, and similarly low λ may make the bid higher than expected, which may run out of budget early and miss the potential valuable impressions. Its value is usually calculated from historical data, but the complexity and volatility of the auction environment makes it a much more complicated problem. During the RTB process, other participants and their bids may change rapidly, and in the meantime, advertisers may change the campaign settings, such as budget and target audience. This requires the bidding algorithm to have better adaptation ability to the environment.

There are two categories of existing work on budget constrained bidding. The first category makes use of the optimal bidding formula, and tunes the parameter λ. One solution is to use $bid = impression\_value/f(budget)$, and this comes from the observation that λ determines the bid and eventually affects how the budget is spent, and thus budget consumption speed can be used as a signal to adjust λ. However, how to determine the best budget consumption speed is still an open question [Zhou *et al.*, 2008]. In the second category, the auction process is considered as a Markov Decision Process [Sutton and Barto, 1998]. Based on this formulation, [Cai *et al.*, 2017] tries to use reinforcement learning (RL) algorithm to solve the bidding problem, and trains RL agent to produce bids directly given impression value. However, model-based RL algorithm used in [Cai *et al.*, 2017] requires valid and accurate state transition matrix, which is difficult to get in practical problems.

In our work, we propose a novel method for budget constrained bidding by model-free reinforcement learning, namely Deep Reinforcement Learning to Bid (DRLB). Given the domain knowledge that the optimal bid is linear with re-

*Corresponding author: di.wudi@alibaba-inc.com

spect to impression value, we train an agent to sequentially regulate the bidding parameter λ instead of producing bids directly, and this formulation significantly reduces the problem complexity. Also, instead of using direct reward from the environment, we propose a new kind of reward function, and use a deep network (RewardNet) for reward approximation, which accelerates the convergence, and improves training efficiency in practice. What is more, we design an adaptive ϵ-greedy policy to adjust the exploration probability based on how the state-action value is distributed. We conducted experiments on large-scale real dataset, and observed improvement on the key metric compared with the baselines. Our contributions can be summarized below.

1. To the best of our knowledge, our work is the first model-free RL algorithm for budget constrained bidding.
2. A novel reward function design is proposed to lead the RL agent to its optimum, which shows superiority in our constrained scenario.
3. An adaptive ϵ-greedy policy is introduced, which is more efficient in exploitation and exploration for DRLB.

The rest of this paper is organized as follows: Section 2 introduces the formulation of budget constrained bidding task and some basic knowledge of reinforcement learning. Then in section 3, we present our method and give detailed analysis on how to design reward function and improve ϵ-greedy policy. Experimental results are introduced in Section 4. Section 5 concludes this paper.

2 Background

2.1 Budget Constrained Bidding

The goal of budget constrained bidding is to design a bidding strategy for maximizing the total value of winning impressions under a budget constraint, which could be described as follows.

During a time period, one day for instance, there exists a candidate impression stream sequence. For each candidate impression, the strategy predicts its value and gives a bid to compete with other unknown bidders. If our bid is the highest one in the auction, we will get the predicted value as reward at the cost of the second highest price in second-price auction [Edelman *et al.*, 2007]. Otherwise, we lose the auction and get no reward from this impression. The whole bidding process will stop when the cost is over budget or time terminates. We hope the strategy could get reward as much as possible. [Lin et al., 2016] formalized budget constrained bidding as a knapsack problem, and [Zhang *et al.*, 2014; Zhang *et al.*, 2016] proved Eq. (1) is an optimal solution in a constant environment.

$$bid = impression\_value/\lambda \tag{1}$$

At the end of each time period, the best λ^* can be easily calculated under any budget constraint. Unfortunately, the strategy needs to bid in real time without knowing the remaining of the stream. What is more, the bidding environment is complicated and not stable, which makes the budget constrained bidding problem a difficult task.

2.2 Reinforcement Learning Basics

Reinforcement learning is an effective solution to finite Markov Decision Process (S, A, P, R). S and A represent states and actions respectively, and the transition probability matrix and reward are given by P and R. The interaction process between agent and environment is summarized as follows. The agent takes an action a_t according to the policy at each discrete time step t based on the observation s_t of the environment. As a response, the environment gives a reward r_{t+1} signal, and then changes to next state s_{t+1}. Since the time is finite, the decision process will terminate in T steps. The goal of the RL agent is to maximize the cumulative discounted reward $return = \Sigma_{t=1}^{T} \gamma^{t-1} r_t, \gamma \in [0, 1]$.

A famous model-free RL algorithm to solve this problem is called Q-learning [Watkins and Dayan, 1992; Strehl *et al.*, 2006]. The ϵ-greedy policy selects an action according to the state-action value function $Q(s_t, a_t)$, which represents the expectation of the return under state s_t when taking action a_t. The agent iteration process is shown in Eq. (2).

$$\begin{aligned} Q(s_t, a_t) \leftarrow & Q(s_t, a_t) \\ & + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \end{aligned} \tag{2}$$

Q-learning can be interpreted as the state-action value function Q consistently updated towards the $Q\ target$ $(r_{t+1} + \gamma \max_a Q(s_{t+1}, a))$ for a small step controlled by α, which makes Q finally converge to the optimal Q^* [Watkins and Dayan, 1992]. Besides, the difference value between $Q\ target$ and Q is called temporal-difference (TD) error.

DQN and Double DQN

Deep Q network (DQN) proposed by [Mnih *et al.*, 2015] uses a deep neural network to represent $Q(S, A; \theta)$, in which θ are the network parameters. DQN has two main modifications based on Q-learning. First, fixed $Q\ target$ is introduced, which means the parameters of $Q\ target$ is updated every C iterations with the parameters of Q. Second, the experience replay technique is applied, which means the historical training data is stored in a fixed-size queue, and then mini-batch of the data is randomly sampled to update the agent at each step. The fixed $Q\ target$ erases the instability of neural network while the experience replay enhances the data utility and decreases the data correlation.

DQN is model-free and off-policy, for the algorithm does not need the environment model and the learning policy is different from the behaviour policy (ϵ-greedy policy). [Mnih *et al.*, 2015] shows that DQN has promising result in Atari 2600 domain, and the ability of processing high dimensional non-linear features is also inspiring.

[Hasselt, 2010] shows that the original DQN is commonly suffered from state-action value overestimation problem. This is mainly because of the max operator in $Q\ target$, which uses the same values both to select and to evaluate an action. This makes it tend to select overestimated values. To solve this problem, the proposed double DQN (DDQN) in [Hasselt, 2010] rewrites the $Q\ target$ from $r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta^-)$ to $r_{t+1} +$

$\gamma Q(s_{t+1}, \text{argmax}_{a'} Q(s_{t+1}, a'; \theta); \theta^-)$ and receives positive result.

3 Method

In this section, budget constrained bidding is formulated as a λ control problem based on Eq. (1). The optimal λ is hard to obtain because the real auction environment is complex and volatile. To solve this problem, a model-free deep reinforcement learning method is proposed. Firstly, we give the problem modelling and general idea of our method. Then the design of reward function and adaptive ϵ-greedy policy are presented, followed by the framework of DRLB method.

3.1 Modelling

The process of adjusting λ is a finite Markov Decision Process with $\gamma = 1$, and we set the episode length to T according to the λ adjustment frequency. We solve this problem through reinforcement learning, in which the main concepts related to our formulation are listed as follows.

- **state**: partial observation of the auction environment, including time, initial bidding parameters, budget consumption, gained reward, level of auction environment competition and campaign information.
- **action**: adjust λ to a lower or higher one.
- **reward**: value of winning impressions after transition from state s_t to state s_{t+1} with action a.
- **environment**: the auction system.

For most reinforcement learning applications, designing reward function is always a critical part, especially when the agent needs to precisely take actions in a complex task. A good reward function will make the agent learn more efficiently and gain better results. Contrarily, agent with poor reward function may get slow convergence or even undesirable results. In our bidding scenario, the reward function should be well designed because every action rewarded is critical to the final result under the limited budget constraint. It should not only represent the value of current action, but also indicate the potential return under budget constraint. In the next subsection, we will introduce the our reward function called RewardNet.

Moreover, the original ϵ-greedy policy should be improved due to the degradation of exploration when the ϵ anneals. To address this problem, we propose an adaptive ϵ-greedy policy to dynamically amplify the exploration probability, which is also shown in the subsections.

Figure 1 presents the main idea of Deep Reinforcement Learning to bid, and DDQN is used to solve this problem. The deep neural network in DDQN can well depict the nonlinear relationship between state features and the state-action value, and the efficiency of data utilization strategy such as prioritized experience replay makes the agent learn more efficiently. Based on RewardNet and adaptive ϵ-greedy policy, the DDQN in our bidding scenario converges more efficiently and receives superior results.

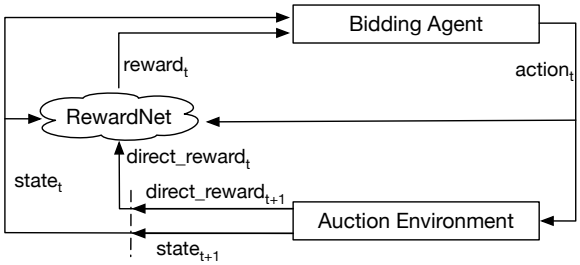


Figure 1: Illustration of Deep Reinforcement Learning to Bid.

3.2 Reward Function Design

Reward Design Trap

During the interaction between agent and environment, agent continuously updates itself according to the reward. We intuitively consider direct reward (total value of winning impressions at $step_i$) as the reward, and the goal of the algorithm is to maximize the $return = \Sigma_{t=1}^{T} direct\_reward_t$. Unfortunately, we find it difficult for the direct reward to lead the agent to its optimum in experiments, which is called Reward Design Trap in this paper.

The reason why agent is hard to quickly converge to the optimum is strongly related to our scenario. In budget constrained bidding scenario, the episodes with good return are hard to be explored because any improper actions may significantly affect the result. Even if the best return is achieved in some rare explorations, the improvement of the agent would be insignificant according to the current DDQN updating process (the chance of being sampled in memory is very low). What is more, the direct reward is monotonously increasing while the λ is decreasing (and vice versa), which means adjusting the λ downward always result in a higher direct reward. Thus, during iterations, the policy may tend to take a decreasing-λ action to get more direct reward which may exhaust the budget and make the episodes with good return even harder to be explored.

It is necessary to design a better reward function which can avoid the drawbacks of direct reward and still leads the agent to converge to the same optimal agent (maximizing $return = \Sigma_{t=1}^{T} direct\_reward_t$).

Reward Function Learning by Neural Network

The defect of the direct reward can be ascribed to the ignorance of the budget cost. Therefore, the reward function should provide extra information about the potential return, and lead the agent to focus on the long-term result. At a specific state, every action could be tried numerous times, and each time there exists a corresponding $return = \Sigma_{t=1}^{T} direct\_reward_t$. We consider each action's best return as its potential contribution to the long-term result.

In our method, we assume the state transition is deterministic. For any state-action pair (s, a), the reward R_n is delivered by the reward function defined by Eq. (3). Compared with the original reward function, the action ignoring the budget consumption will yield poor return and never be encouraged. Besides, the optimal policy would remain the same if we replace the original reward function with the new one. The proof is

presented in Theorem 1.

$$R_n(s,a) = max_{e \in E(s,a)} return_e \quad (3)$$

, where $E(s,a)$ represents the set of episodes that take action a at state s

Because it is impossible to store all the states and actions with their reward, we deploy a neural network to predict it. Meanwhile, the forecast ability of the neural network towards unseen states and actions will further boost the convergence. The training dataset is the $(start\_state, current\_state, action)$ tuples with their historical maximum return in a time window. The iteration process is shown in Algorithm 1.

Theorem 1. *Let $\pi^*_{M_n}$ be an optimal policy for the new MDP $M_n = (S, A, P, R_n)$ with the reward R_n from rewardNet. Then, $\pi^*_{M_n}$ is guaranteed to be an optimal policy $\pi^*_{M_d}$ for the original MDP $M_d = (S, A, P, R_d)$ with direct reward R_d.*

Proof. Let $\{(s_i, \pi^*_{M_n}(s_i), R_n(s_i, \pi^*_{M_n}(s_i)))\}, i = 1, .., T$ be an episode produced by policy $\pi^*_{M_n}$ at a starting state s_1, where $\pi^*_{M_n}(s_i)$ is the action at state s_i and T is the episode length. Let $V^{\pi}_M(s)$ denote the return obtained at state s when applying policy π to MDP M. The discounting factor $\gamma = 1$.

Since the initial bidding parameter and campaign information are included in the state according to Section 3.1, the episodes that go through any of the above state s_i should also start from s_1. Thus according to Eq. (3) we have

$$R_n(s_i, \pi^*_{M_n}(s_i)) \leq V^{\pi^*_{M_d}}_{M_d}(s_1), \forall i = 1, .., T. \quad (4)$$

, where $V^{\pi^*_{M_d}}_{M_d}(s_1)$ is denoted as the maximal return for M_d. Therefore, we can infer that the return $V^{\pi^*_{M_n}}_{M_n}(s_1)$, which is the sum of reward R_n, is no larger than $T \cdot V^{\pi^*_{M_d}}_{M_d}(s_1)$, i.e.,

$$V^{\pi^*_{M_n}}_{M_n}(s_1) = \Sigma^T_{i=1} R_n(s_i, \pi^*_{M_n}(s_i)) \leq T \cdot V^{\pi^*_{M_d}}_{M_d}(s_1), \quad (5)$$

, with equality if and only if $R_n(s_i, \pi^*_{M_n}(s_i)) = V^{\pi^*_{M_d}}_{M_d}(s_1)$, $\forall i = 1, ..., T$.

Besides, the episode produced by $\pi^*_{M_d}$ obtains the maximal return $V^{\pi^*_{M_d}}_{M_d}(s_1)$ for M_d. Therefore at each state of this episode, the reward R_n, which is the maximal return of episodes according to Eq. (3), is also $V^{\pi^*_{M_d}}_{M_d}(s_1)$,i.e.,

$$V^{\pi^*_{M_d}}_{M_n}(s_1) = \Sigma^T_{i=1} R_n(s_i, \pi^*_{M_d}(s_i)) = T \cdot V^{\pi^*_{M_d}}_{M_d}(s_1) \quad (6)$$

Since $\pi^*_{M_n}$ is the optimal policy for the new MDP M_n , it obtains the maximal value for M_n at s_1. Thus we have

$$V^{\pi^*_{M_n}}_{M_n}(s_1) \geq V^{\pi^*_{M_d}}_{M_n}(s_1) = T \cdot V^{\pi^*_{M_d}}_{M_d}(s_1). \quad (7)$$

Based on Eqs. (5) and (7), we have that $V^{\pi^*_{M_n}}_{M_n}(s_1) = T \cdot V^{\pi^*_{M_d}}_{M_d}(s_1)$ and $R_n(s_i, \pi^*_{M_n}(s_i)) = V^{\pi^*_{M_d}}_{M_d}(s_1), \forall i = 1, .., T$.

This means $\pi^*_{M_n}(s_i)$ is an optimal action for M_d at any state s_i. Therefore, the optimal policy for M_n satisfies $\pi^*_{M_n}(s) \in \text{argmax}_{a \in A} Q^*_{M_d}(s, a), \forall s \in S$ and is also optimal for M_d. □

Algorithm 1: Learning RewardNet

```
Initialize replay memory D_1 to capacity N_1;
Initialize reward network R with random weights η;
Initialize reward dictionary M;
for episode = 1 to Z do
    Initialize temporary set S;
    Initialize rtn = 0;
    for t = 1 to T do
        if len(D_1) > K then
            Sample mini-batch of (s_j, a_j, r_{j+1}) from D_1;
            Perform a gradient descent step on
              (R(s_j, a_j; η) − r_{j+1})^2 with respect to the
              network parameters η;
        RL agent executes a_t in the Environment;
        Obtain direct reward r_{t+1} from the Environment;
        Set rtn = rtn + r_{t+1};
        Store pair (s_t, a_t) in S;
    for (s_j, a_j) in S do
        Set M(s_0, s_j, a_j) = max(M(s_0, s_j, a_j), rtn);
        Store pair (s_j, a_j, M(s_0, s_j, a_j)) in D_1;
        Discard old pairs in D_1 out of the time window;
```

3.3 Adaptive ϵ-greedy Policy

The original DDQN uses ϵ-greedy policy to balance exploitation and exploration (E&E), which means the agent selects the best action with probability 1-ϵ, and randomly takes an action with probability ϵ. The ϵ starts with a large value and anneals to a small value over time. It is well known that improper ϵ may result in poor E&E efficiency: the high ϵ annealing speed means insufficient exploration, while the low annealing speed means slow convergence.

[Lin *et al.*, 2016] formulates budget constrained bidding as a knapsack problem. It can be inferred that at any specific phase of an bidding process, there exists an optimal λ^* for the rest of the impressions. Thus, under a specific state at time t, the best action is to adjust the current λ to the best λ^*, and the larger the deviation from the best λ^* is, the less return will be gained.

Based on this inference, the state-action value should be unimodal distributed. If the distribution is not unimodal, the estimation of the state-action value will need to be improved by increasing the exploration probability. Based on the original ϵ-greedy policy, we investigate the state-action value at each step, and dynamically change ϵ to a constant value ζ (larger than the annealed ϵ) when the distribution is not unimodal.

3.4 Deep Reinforcement Learning to Bid

As illustrated in Figure 1, based on the RewardNet and adaptive ϵ-greedy policy above, the whole process of bidding agent interacting with auction environment can be summarized as follows: according to adaptive ϵ-greedy policy, the agent takes an action a_t (adjusting current λ by some degree) under state s_t (indicators of current situation, including level of bidding competition, budget consumption speed, etc.), and

then gives bids to compete with other unknown bidders according to Eq. (1). The auction environment returns direct reward $direct\_reward_{t+1}$ and next state s_{t+1} as response. The agent updates itself according to the TD errors calculated based on $\langle state, next\_state, action, reward\rangle$ mini-batch sampled from historical episodes data, in which the reward is delivered by RewardNet. The episode ends when the agent is out of budget or it reaches the end of the day. The pseudo code is shown in Algorithm 2.

Algorithm 2: Deep Reinforcement Learning to Bid

```
Initialize replay memory D_2 to capacity N_2;
Initialize Q with random weights θ;
Initialize Q target with weights θ^- = θ;
for episode = 1 to Z do
    Initialize λ = λ_0 and start state s_1;
    for t = 1 to T do
        Update RewardNet (8-10 in Algo. 1);
        Take action a_t through adaptive ε-greedy policy;
        Update λ and bid according to Eq. (1);
        Get r_{t+1} from RewardNet and observe s_{t+1};
        Store (s_t, s_{t+1}, a_t, r_{t+1}) in D_2;
        Sample mini-batch of (s_j, s_{j+1}, a_j, r_{j+1}) from D_2 according to the TD error;
        if episode terminates then
            Set y_j = r_j;
        else
            Set y_j = r_j + γQ(s_{j+1}, argmax_{a'} Q(s_{j+1}, a'; θ); θ^-);
        Perform a gradient descent step on (y_j − Q(s_j, a_j; θ))^2 with respect to θ;
        Every C steps reset Q target = Q;
    Store data for RewardNet (15-18 in Algo. 1);
```

4 Experimental Results

In this section, we evaluate the performance of the proposed DRLB. First, the experimental setup and implementation detail of DRLB are introduced. Then, we quantitatively compare DRLB with FLB and BSLB in Section 4.3. We also investigate the properties of RewardNet and adaptive ϵ-greedy policy respectively in Section 4.4 and 4.5.

4.1 Experimental Setup

Dataset

A real-world dataset from a famous e-commerce advertising platform is used in our experiments. The dataset comprises complete auction logs of 200M impressions with the corresponding user response estimation, i.e. impression value, for 8 continuous days in Jan. 2018. The data of first 7 days is used for training and that of the last day is for testing.

Evaluation Metric

The goal of the bidding agent is to optimize the campaign's return under a budget constraint. The best return can be easily calculated given the complete auction information. Thus, the difference between the algorithm return R and the best return R^* is a simple and effective metric to evaluate the algorithm performance, which is denoted by R/R^*.

Baseline Methods

1) Fixed Linear Bidding: Fixed Linear Bidding (FLB) uses a fixed initial λ_0 to linearly bid according to Eq. (8), which is straightforward and widely used in industry.

$$bid = impression\_value/\lambda_0 \tag{8}$$

2) Budget Smoothed Linear Bidding: [Hegeman *et al.*, 2011] gives a practical way of bidding under budget constraint[1], which is shown in (9). Budget Smoothed Linear Bidding (BSLB) combined the classic bidding Eq. (8) with the current budget consumption information (Δ). When the budget left ratio is lower than the time left ratio, the bid is decreased by adjusting the λ_0 downward, otherwise, the bid is increased to consume more budget.

$$bid = impression\_value/(\lambda_0 * \Delta), \tag{9}$$

where $\Delta = time\_left\_ratio/budget\_left\_ratio$.

4.2 Implementation Details of DRLB

We take a fully connected neural network with 3 hidden layers as our state-action value function Q and the RewardNet, and each layer comprises 100 nodes. The mini-batch size is set to 32 and the replay memory size is set to 100,000. The agent has 11 actions in our implementation, which corresponds to 11 λ adjustments: -50%, -15%, -8%, -3%, -1%, 0%, 1%, 3%, 8%, 15%, 50%. The λ is adjusted every 15 minutes. The annealing speed of ϵ for ϵ-greedy policy is controlled by (10), in which the annealing rate r is a preset hyper parameter. Because the bidding problem is to maximize the sum of immediate reward, so the γ in MDP is set to 1.0. The state consists of 46 features indicating the bidding situation, which is organized into several groups, including time, initial bidding parameters, budget consumption, gained reward, level of auction environment competition, campaign information, and because of limitations of space, the detail features are not listed.

$$\epsilon = \max(0.95 - r * step, 0.05) \tag{10}$$

4.3 Comparison with FLB and BSLB

We conduct experiments to compare the performance of DRLB, FLB and BSLB. In the experiments, λ_0 of each ad campaign is set to its best λ of yesterday. Due to the variance of auction environment and campaign settings, λ_0 may deviate from λ^*. It is obvious that the deviation between λ_0 and λ^* may affect the outcomes of these methods. To investigate the performance with different deviations, quantified as $(\lambda_0 - \lambda^*)/\lambda^*$, we divide all campaigns into 9 groups according to the λ deviation and evaluate the three methods for each

[1]For consistency, the equation is slightly transformed based on the notations in our paper.

λ Deviation	Campaigns	R/R^*			Improvement of R/R^*	
		FLB	BSLB	DRLB	FLB	BSLB
$[-100\%, -80\%)$	43	0.436	0.525	**0.878**	101.38%	67.24%
$[-80\%, -40\%)$	89	0.434	0.647	**0.884**	103.69%	36.63%
$[-40\%, -20\%)$	66	0.697	0.901	**0.945**	35.58%	4.88%
$[-20\%, 0\%)$	41	0.863	0.936	**0.953**	10.43%	1.82%
$[0\%, 20\%)$	39	0.825	0.925	**0.950**	15.15%	2.70%
$[20\%, 40\%)$	48	0.491	0.947	**0.948**	93.08%	0.11%
$[40\%, 80\%)$	85	0.391	0.904	**0.928**	137.34%	2.65%
$[80\%, 160\%)$	57	0.307	0.813	**0.924**	200.98%	13.65%
$[160\%, \infty)$	32	0.291	0.668	**0.904**	210.65%	35.33%
Average		0.526	0.807	**0.924**	100.92%	18.33%

Table 1: The R/R^* of DRLB, FLB and BSLB, and the improvement of DRLB over other two methods in 9 groups of deviation.

group. For each group we report the average R/R^* results for all the campaigns.

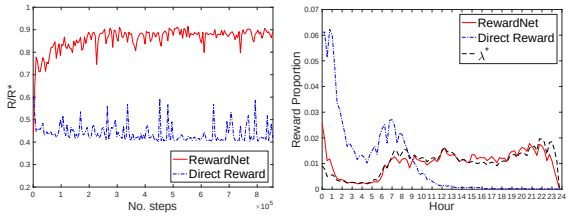
The experimental results are summarized in Table 1, which demonstrate that DRLB outperforms FLB and BSLB in all 9 groups and the overall improvement over FLB and BSLB is 100.96% and 18.36% respectively. Moreover, as the λ deviation increases, the performance of FLB and BSLB degrades enormously while DRLB can still obtain desirable results. In the case that the deviation is small, e.g.[-20%, 0%), all methods deliver satisfying results. In the groups where λ_0 is far away from λ^*, DRLB shows particular advantage over FLB and BSLB, which indicates its superior adaptability even starting with a terrible λ_0. For instance, when the λ deviation lies in [-80%, -40%), the average R/R^* of DRLB is 0.884 while that of FLB and BSLB is 0.434 and 0.647 respectively.

We analyse the experimental results as follows. FLB gives the worst performance since it uses a fixed λ_0 and is unaware of the bidding environment. Both BSLB and DRLB show the ability to cope with the deviated λ_0. However, DRLB is better than BSLB due to the following two reasons. 1) BSLB only takes into consideration $budget\_left\_ratio$ and $time\_left\_ratio$, while DRLB could make full use of bidding information to enable accurate control. 2) According to Eq. (9), BSLB is insensitive to the time elapse at the early stages of the day and thus shows limited adaptability to the environment. As for DRLB, it is always able to make timely reaction since it could adjust λ by a large scale every time step.

4.4 Comparison with Direct Reward

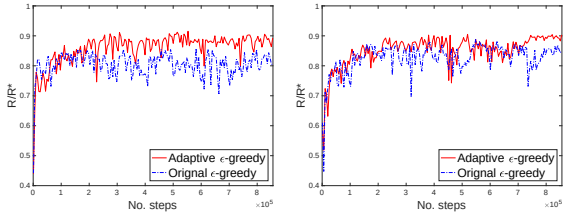
In order to probe the effect of RewardNet, we train two models independently on the same dataset. One model deploys RewardNet, while the other uses direct reward. We dump the models every 960 steps during the training process, and compare them on the test dataset. As illustrated in Figure 2a, the model with RewardNet yields satisfying R/R^* of 0.893 within a small number of steps, while the other model gets stuck with some inferior policy and yields poor R/R^* of 0.418. The experimental results indicate the effectiveness of RewardNet in leading the agent to the optimal policy.

Furthermore, we depict a typical case of the reward distribution over time steps to investigate the reward design trap, which is shown in Figure 2b. The ideal reward distribution derived from a fixed λ^* is also illustrated as a reference. The results demonstrate that the model trained with direct reward is prone to obtain more reward in the early steps, which exhausts the budget and results in poor performance from a long-term view. In the case of RewardNet, the reward distribution is similar to that of λ^*, which shows that RewardNet helps the agent to avoid greedy behaviour and better utilize the budget for long-term benefits.



(a) (b)

Figure 2: Comparison between RewardNet and direct reward. (a) The R/R^* of two models over steps. (b) Reward distribution of two models along with the ideal one.



(a) (b)

Figure 3: Performance of adaptive ϵ-greedy and original ϵ-greedy. (a) Annealling rate: 2e-5. (b) Annealling rate: 1e-5.

4.5 Adaptive ϵ-greedy Policy for Exploration

Experiments are performed to compare our adaptive ϵ-greedy policy with the original ϵ-greedy policy. We evaluate these policies in settings with two different annealing rates in Eq. (10), i.e. r=2e-5 and r=1e-5 respectively. The results shown in Figure 3 demonstrate that our adaptive exploration policy helps the agent explore effectively and achieve better performance in both settings. Specially, the adaptive ϵ-greedy exploration enables fast convergence and significantly outperforms the original ϵ-greedy in the setting with the higher annealing rate. This indicates the superior efficiency of our exploration strategy in circumstance where training time is limited, which is common in reinforcement learning problems.

5 Conclusion

In this paper, we propose a model-free deep reinforcement learning method to solve the budget constrained bidding problem in RTB display advertising. The bidding problem is innovatively formulated as a λ control problem based on linear bidding equation. To solve the reward design trap, which makes the agent hard to converge to the optimum, we design RewardNet to generate reward instead of using the direct reward. Furthermore, the problem of insufficient exploration is also alleviated by dynamically changing the random probability of the original ϵ-greedy policy. The experiments upon the real-world dataset show that our model converges quickly and

significantly outperforms the widely used bidding methods. Last but not least, the idea of RewardNet is general, which can be applied to other deterministic MDP problems, especially for those aiming to maximize long-term result when the reward function is hard to design.

References

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

[Edelman *et al.*, 2007] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review*, 97(1):242–259, 2007.

[Hasselt, 2010] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[Hegeman *et al.*, 2011] John Hegeman, Rong Yan, and Gregory Joseph Badros. Budget-based advertisment bidding, November 10 2011. US Patent 20130124308A1.

[Lee *et al.*, 2012] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 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*, pages 768–776. ACM, 2012.

[Lin *et al.*, 2016] Chi-Chun Lin, Kun-Ta Chuang, Wush Chi-Hsuan Wu, and Ming-Syan Chen. Combining powers of two predictors in optimizing real-time bidding strategy under constrained budget. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2143–2148. ACM, 2016.

[McMahan *et al.*, 2013] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[Mnih *et al.*, 2015] 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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Qu *et al.*, 2016] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1149–1154. IEEE, 2016.

[Strehl *et al.*, 2006] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. Pac model-free reinforcement learning. In *ICML*, 2006.

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

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

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

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

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

[Zhang *et al.*, 2016] Weinan Zhang, Kan Ren, and Jun Wang. Optimal real-time bidding frameworks discussion. *arXiv preprint arXiv:1602.01007*, 2016.

[Zhou *et al.*, 2008] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*, pages 566–576. Springer, 2008.