Learning Adaptive Display Exposure for Real-Time Advertising

Weixun Wang
wxwang@tju.edu.cn
Tianjin University

Junqi Jin
junqi.jjq@alibaba-inc.com
Alibaba Group

Jianye Hao
jianye.hao@tju.edu.cn
Tianjin University

Chunjie Chen, Chuan Yu
{chunjie.ccj,yuchuan.yc}@alibaba-inc.com
Alibaba Group

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

Han Li, Jian Xu, Kun Gai
{lihan.lh,xiyu.xj,jingshi.gk}@alibaba-inc.com
Alibaba Group

Xiaotian Hao
xiaotianhao@tju.edu.cn
Tianjin University

Yixi Wang
yixiwang2017@outlook.com
Tianjin University

ABSTRACT
In E-commerce advertising, where product recommendations and product ads are presented to users simultaneously, the traditional setting is to display ads at fixed positions. However, under such a setting, the advertising system loses the flexibility to control the number and positions of ads, resulting in sub-optimal platform revenue and user experience. Consequently, major e-commerce platforms (e.g., Taobao.com) have begun to consider more flexible ways to display ads. In this paper, we investigate the problem of advertising with adaptive exposure: can we dynamically determine the number and positions of ads for each user visit under certain business constraints so that the platform revenue can be increased? More specifically, we consider two types of constraints: request-level constraint ensures user experience for each user visit, and platform-level constraint controls the overall platform monetization rate. We model this problem as a Constrained Markov Decision Process with per-state constraint (psCMDP) and propose a constrained two-level reinforcement learning approach to decompose the original problem into two relatively independent sub-problems. To accelerate policy learning, we also devise a constrained hindsight experience replay mechanism. Experimental evaluations on industry-scale real-world datasets demonstrate the merits of our approach in both obtaining higher revenue under the constraints and the effectiveness of the constrained hindsight experience replay mechanism.

KEYWORDS
Learning to Advertise, Adaptive Ads Exposure, Real-Time Advertising, Deep Reinforcement Learning, Constrained Two-Level Reinforcement Learning

1 INTRODUCTION
With the advances of deep neural network [14, 21], Deep Reinforcement Learning (DRL) approaches have made significant progress in a number of applications including Atari games [27] and robot locomotion and manipulation [23, 31]. Recently, we also witness successful applications of DRL techniques to optimize the decision-making process in E-commerce from different aspects including online recommendation [11], impression allocation [10, 41], advertising bidding strategies [19, 37, 40] and product ranking [16].

In traditional online advertising, the ad positions are fixed, and we only need to determine which ads to be shown in these positions for each user request [26]. This can be modeled as an ads position bidding problem and DRL techniques have been shown to be effective in learning bidding strategies for advertisers [19, 37, 40]. However, fixing ad positions limit the flexibility of the advertising system. Intuitively, if a user is with high monetization value (e.g., likes to click ads), it is reasonable for the advertising platform to display more ads when this user visits. On the other hand, we are also concerned with displaying too many ads for two reasons. First, it might lead to poor user experience and have a negative impact on user retention. Second, monetization rate is an important business index for a company to moderate. Therefore, in this paper, we consider two levels of constraints: (1) request-level: the maximum number of ads on each request cannot exceed a threshold; and (2) platform-level: the average number of ads over all the requests (within a time window) cannot exceed a threshold. Under the above constraints, we investigate the problem of advertising with adaptive exposure: can we dynamically determine the set of ads and their positions for each user visit so that the platform revenue can be maximized? We call the above problem as advertising with adaptive exposure problem.

Fig.1 illustrates the flexible adaptive exposure mechanism adopted by Taobao's e-commerce company in China. For each user visit, the platform presents a dynamic mixture of product recommendations and product ads. The ad positions are not a fixed prior, and they are determined by the user’s profile and behaviors. The adaptive exposure problem can be formalized as a sequential decision problem. In each step, the recommendation and the advertising systems first select some items based on their scoring systems independently. Then these commodities are sorted altogether by their scores and the top few items are exposed to the request (user).

\footnote{It is worth pointing out that the request here refers to the user’s access to the platform (for example: opening the mobile app, swipe the screen)\footnote{One of the largest}}
We model the above problem as a Constrained Markov Decision Process (CMDP) [3]. Although optimal policies for small-sized CMDPs can be derived using linear programming [3], it is difficult to construct such policies for large-scale and complex real-world e-commerce platforms. Thus, we resort to model-free RL approaches to learn approximately optimal solutions [1, 34]. Existing model-free RL approaches for solving CMDPs are trajectory-based: they update policies by propagating constraint-violation signals over the entire trajectory [1, 29]. Unfortunately, most of them fail to meet the constraints [34]. To address this issue, Tessler et al. [34] propose the Reward Constrained Policy Optimization (RCPO), which decomposes the trajectory constraints into per-state penalties and dynamically adjusts their weights. To ensure that the overall penalty of a trajectory satisfies the given constraint, the constraint-violation signals are also propagated back along the entire trajectory. However, in the advertising with adaptive exposure problem, we need to satisfy both state-level (request-level) and trajectory-level (platform-level) constraints. RCPO only considers trajectory-level constraints and thus cannot be directly applied here.

In this paper, we first model the advertising with adaptive exposure problem as a CMDP with per-state constraint (psCMDP). Then we propose a constrained two-level reinforcement learning framework to learn optimal advertising policies satisfying both state-level and trajectory-level constraints. In our framework, the trajectory-level constraint and the state-level constraint are divided into different levels in the learning process. The higher level policy breaks a trajectory into multiple sub-trajectories and tackles the problem of selecting constraints for each sub-trajectory to maximize total revenue under the trajectory-level constraint. Each sub-trajectory identifies an independent optimization problem with both sub-trajectory constraint and state-level constraint. Here we simplify the sub-trajectory optimization problem at the cost of sacrificing the policy optimality by treating the sub-trajectory constraint as another state-level constraint. In this way, we can easily combine the sub-trajectory constraint with the original state-level constraint and use off-policy methods such as Deep Deterministic Policy Gradient (DDPG) [24] with auxiliary task [18] to train the lower level policy. We also propose Constrained Hindsight Experience Replay (CHER) to accelerate the lower level policy training.

Note that our framework can be naturally extended to more levels by further decomposing each sub-trajectory into a number of sub-trajectories. Thus it is expected that the quality of the learned policy would be improved when we increase the number of levels, which means the length of each sub-trajectory at the lower levels is reduced. Thus our framework is flexible enough to make a compromise between training efficiency and policy optimality. In this paper, we set our framework to be two levels. One additional benefit of our two-level framework is that we can easily reuse the lower level policy to train the higher level constraint selection policy in case the trajectory-level constraint is adjusted. We evaluate our approach using real-world datasets from Taobao platform both offline and online. Our approach can improve the advertising revenue and the advertisers’ income while satisfying the constraints at both levels. In the lower level, we verify that CHER mechanism can significantly improve the training speed and reduce the deviation of the per-state constraint. Moreover, in the higher level, our method can make good use of the lower level policy set to learn higher level policies with respect to different platform-level constraints.

2 PRELIMINARY: CONSTRAINED REINFORCEMENT LEARNING

Reinforcement learning (RL) allows agents to interact with the environment by sequentially taking actions and observing rewards to maximize the cumulative reward [32]. RL can be modeled as a Markov Decision Process (MDP), which is defined as a tuple \((S, A, R, P)\). \(S\) is the state space and \(A\) is the action space. The immediate reward function is \(R: S \times A \times S \rightarrow \mathbb{R}\). \(P\) is the state transition probability, \(S \times A \times S \rightarrow [0, 1]\). There exists a policy \(\pi\) on \(A\), which defines an agent’s behavior. The agent uses its policy to interact with the environment and generates a trajectory \(T = \{s_0, a_0, r_0, s_1, \ldots, s_T, a_T, r_T, s_{T+1}, \ldots \}\). Its goal is to learn an optimal policy \(\pi^*\) which maximizes the expected return given the initial state:

\[
\pi^* = \arg \max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{T} y^t r_t | \pi \right]
\]

(1)

Here \(y \in [0, 1]\) is the discount factor, \(T\) is the length of the trajectory \(T\). The Constrained Markov Decision Process (CMDP) [3] is generally used to deal with the situation, by which the feasible policies are restricted. Specifically, CMDP is augmented with auxiliary cost functions \(C_T: S \times A \times S \rightarrow \mathbb{R}\) and a upperbound constraint \(\alpha_T\). Let \(J_{C_T}(\pi)\) be the cumulative discounted cost of policy \(\pi\). The expected discounted return is defined as follows:

\[
J_{C_T}(\pi) = \mathbb{E}_{T \sim \pi} \left[ \sum_{t=0}^{T} y^t C_T(s_t, a_t, s_{t+1}) | \pi \right]
\]

(2)


Table 1: List of notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>The sequence of incoming requests. $Q = {q_1, q_2, \ldots, q_M}$. $</td>
</tr>
<tr>
<td>$q_i$</td>
<td>The $i$-th request in the day.</td>
</tr>
<tr>
<td>$N^d$</td>
<td>The number of candidate ads.</td>
</tr>
<tr>
<td>$N^r$</td>
<td>The number of recommended products.</td>
</tr>
<tr>
<td>$N_i$</td>
<td>The number of commodities shown to $q_i$. Usually, $N_i = N^r_i \forall i \leq i' \leq M$. And $N_i &lt; N^d + N^r$.</td>
</tr>
<tr>
<td>$N^d_i$</td>
<td>The number of ads exposed for $q_i$.</td>
</tr>
<tr>
<td>$PVR$</td>
<td>The total percentage of ads exposed in one day. $PVR = \frac{\sum_{i=1}^{M} N^d_i}{\sum_{i=1}^{M} N^r_i}$.</td>
</tr>
<tr>
<td>$PVR_i$</td>
<td>The percentage of ads exposed for $q_i$. $PVR_i = \frac{N^d_i}{N_i}$.</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>The maximum percentage of the total ads exposed in one day.</td>
</tr>
<tr>
<td>$\beta$</td>
<td>The maximum percentage of the ads exposed for each request.</td>
</tr>
<tr>
<td>$D_i$</td>
<td>The candidate ads set for $q_i$. $</td>
</tr>
</tbody>
</table>

The set of feasible stationary policies for a CMDP is then:

$$\Pi_C = \{\pi \in \Pi : J_{C}(\pi) \leq \nu_T\}$$  \hspace{1cm} (3)

And the policy is optimized by limiting the policy $\pi \in \Pi_C$ in the Equation 1. For DRL methods, Achiam et al. [1] propose a new approach which replaces the optimization objective and constraints with surrogate functions, and uses Trust Region Policy Optimization [30] to learn the policy, achieving near-constraint satisfaction in each iteration. Tessler et al. [34] use a method similar to WeiMDP [13]. WeiMDP [13] introduces a weight parameter $\xi \in [0, 1]$ and a derived weighted reward function $r^T_\xi$, which is defined as:

$$r^T_\xi = \xi \cdot r_T + (1 - \xi) \cdot C_T(s_t, a_t, s_{t+1})$$  \hspace{1cm} (4)

where $r_T$ and $C_T(s_t, a_t, s_{t+1})$ are the reward and the auxiliary cost under the transition $(s_t, a_t, s_{t+1})$ respectively. For a fixed $\xi$, this new unconstrained MDP can be solved with standard methods, e.g. Q-Learning [32]. Tessler et al. [34] use the weight $\xi$ as the input to the value function and dynamically adjust the weight by backpropagation.

3 ADVERTISING WITH ADAPTIVE EXPOSURE

3.1 Adaptive Exposure Mechanism

In an E-commerce platform, user requests come in order, $Q = \{q_1, q_2, \ldots, q_M\}$. When a user sends a shopping request $q_i$, $N_i$ commodities are exposed to the request based on the user’s shopping history and personal preferences. The $N_i$ commodities are composed of advertising and recommendation products. Exposing more ads may increase the advertising revenue. However, the exposed ads for users are not necessarily their favorite or needed products. Therefore, we should limit the number of exposed ads for each user’s request.

For each request $q_i$, traditional E-commerce systems use fixed-positions to expose ads: $N^d_i = K$, $\forall i \leq i' \leq M$, where $K$ is the number of fixed positions. However, it is obvious that this advertising mechanism is not optimal. Different consumers have different shopping habits and preferences to different products and ads. Therefore, we can expose more advertising products to those consumers who are more likely to click and purchase them (thus increasing the advertising revenue) and vice versa.

To this end, recently Taobao (one of the largest Chinese E-commerce platform) has begun to adopt more flexible and mixed exposing mechanism for exposing advertising and recommended products (Fig. 1). Specifically, for each request $q_i$, the recommendation and the advertising systems first select the top $N^r$ and $N^d$ items based on their scoring systems independently (Fig. 1, Step (2) and Step (3)). Then these commodities are sorted altogether according to scores in descending order(Fig. 1, Step (5)) and the top $N_i$ items are exposed to this request (Fig. 1, Step (6)).

In the meantime, to ensure users’ shopping experience, we need to impose the following constraints:

- **platform-level constraint** $C_{\text{platform}}$, the total percentage of ads exposed in one day should not exceed a certain threshold $\alpha$:

  $$PVR \leq \alpha$$  \hspace{1cm} (5)

- **request-level constraint** $C_{\text{request}}$, the percentage of ads exposed for each request should not exceed a certain threshold $\beta$:

  $$PVR_i \leq \beta, \forall i \leq i' \leq M$$  \hspace{1cm} (6)

where $\alpha \leq \beta$. This means that we can exploit this inequality requirement to expose different numbers of ads to different requests according to users’ profiles, e.g. expose more ads to users who are more interested in the ads and can increase the average advertising revenue, and fewer ads for the others. On one hand, for each request, the size of $N^d_i$ can be automatically adjusted according to the quality of the candidate products and ads. On the other hand, the positions of $N_i$ items are determined by the quality of the products and ads, which can further optimize the user experience. In this way, we can increase the total advertising revenue while satisfying both the request-level and platform-level constraints. The scoring systems for both recommendation and advertising sides can be viewed as black boxes. However, from the advertising perspective, we can adaptively adjust the score of each ad to change their relative rankings and the number of ads to be exposed eventually (Fig. 1, Step (4)).

The adaptive exposure mechanism can potentially improve the advertising revenue, however, it faces a number of challenges. First, the ads score adjusting strategy is highly sensitive to the dynamics of recommendation system (e.g., system upgrading) and other components of advertising system (e.g., the candidate ads selection mechanism may upgraded). Our advertising system needs to be adjusted to meet the constraints. Second, actual business needs to change from time to time (adjust to platform-level and request-level...
3.2 Formulation

3.2.1 Problem Description. From the advertising perspective, the above advertising exposure problem can be seen as a bidding problem: The product displayed in each user request is determined by the score (bid price) of the advertising item and the recommended item (Rank by bid price, the higher the price, the more possible to be displayed), and the advertising system adjusts the score of the original advertisement (the auction bid) to satisfy the constraint and increase revenue (auction results). We follow the settings of the bidding problem in Display Advertising [9, 39] and extend it to advertising with adaptive exposure problem. Formally, for the j-th ad $d_j \in D_i$ in request $q_i$, its score is adjusted as follows:

$$\text{score}'_{i,j} = \text{score}_{i,j} \times \eta_{i,j}$$

(7)

where $\eta_{i,j} = b(q_i, d_j; \theta)$, $b$ is a bidding function and $\theta$ is the parameters of $b$, $\text{score}_{i,j}$ is the original score given by the advertising system for $d_j$ in request $q_i$. Within the advertising system only, we cannot directly figure out whether the ad (which score has been adjusted) will be finally exposed to the request. We can only get the final displayed results from the Shuffled System (Fig. 1, Step (5)). So we define $w(b(q_i, d_j; \theta), q_i, d_j) = E_{\phi}(l(b(q_i, d_j; \theta), \phi))$ as the probability of winning the bid request ($q_i, d_j$) with bid adjustment ratio $b(q_i, d_j; \theta)$, where $\phi$ is the parameter of the recommendation system and $l(b(q_i, d_j; \theta), \phi)$ indicates whether the advertisement $d_j$ is finally displayed in request $q_i$ given the recommendation system’s parameters $\phi$. We use $v(b(q_i, d_j; \theta), q_i, d_j)$to denote the expected revenue value of the advertising product $d_j$ under request $q_i$. Then under the premise of satisfying the constraints $C_{\text{request}}$ and $C_{\text{platform}}$, the optimization goal of the advertising system can be written as follows:

$$\max_\theta \sum_{q_i \in Q} \sum_{d_j \in D_i} v(b(q_i, d_j; \theta), q_i, d_j) \times w(b(q_i, d_j; \theta), q_i, d_j)$$

s.t. \begin{align}
\sum_{d_j \in D_i} v(b(q_i, d_j; \theta), q_i, d_j) & \leq \beta, \forall q_i \in Q \\
\sum_{q_i \in Q} \sum_{d_j \in D_i} v(b(q_i, d_j; \theta), q_i, d_j) & \leq \alpha
\end{align}

(8)

Requests arrive in chronological order. To satisfy the constraint $C_{\text{platform}}$ (e.g., the maximum proportion of displaying ads during a day of the platform), if the system exposes too many ads during early period, it should expose fewer ads later. Hence the above problem is naturally a sequential decision-making problem.

3.2.2 Problem Formulation. To solve such a sequential decision-making problem, one typical method is to model it as MDP [9] or CMDP [37], and then use reinforcement learning techniques to solve the problem. In practice, we cannot acquire and make accurate predictions of the environmental information like $v(b(q_i, d_j; \theta), q_i, d_j)$ and $w(b(q_i, d_j; \theta), q_i, d_j)$ aforesaid, thus we resort to model-free reinforcement learning techniques. However, since there exist both platform-level and request-level constraints, the traditional CMDP [3] cannot be directly applied here. We propose a special CMDP which we term as CMDP with per-state constraint (psCMDP). Formally a psCMDP can be defined as a tuple $(S, A, R, P, C_T, C_S)$. Comparing to the original CMDP [3], we see that the difference here is that for each trajectory $\tau$, psCMDP needs to satisfy not only the trajectory-level constraint $C_T$:

$$J_{C_T}(\pi) = \sum_{t=0}^{T} \gamma^t C_T(s_t, \pi(s_t), s_{t+1}) \leq u_T$$

(9)

but also the per-state constraint $C_S$ over each request:

$$J_{C_S}(\pi) = C_S(s_t, \pi(s_t), s_{t+1}) \leq u_S, \forall (s_t, a_t, s_{t+1}) \in \tau$$

(10)

where $C_S : S \times A \times S \rightarrow \mathbb{R}$ and $u_S$ is the upper bound of $C_S$. So the set of feasible stationary policies for a psCMDP is:

$$\Pi_{ps} = \{ \pi \in \Pi : J_{C_T}(\pi) \leq u_T \} \cap \{ \pi \in \Pi : J_{C_S}(\pi) \leq u_S \}$$

(11)

The components of a psCMDP are described in details as follows:

- $S$: The state should reflect both the environment and the constraints in principle. In our settings, we consider the following statistics for $s_t$: 1) information related to the current request $q_t$, e.g., features of the candidate ads; 2) system context information, e.g., the number of ads exposed up to time $t$.

- $A$: Considering the system picks out products for each request by the score of all the products, we adjust all the ad candidates’ score of a request at once. Accordingly, we denote $a_t = (\eta^1_t, \eta^2_t, ..., \eta^N_t)$, where $\eta^j_t$ is the coefficient of the j-th ad $d_j$ for request $q_t$, where $1 \leq j \leq N^d$.

- $R$: $R(s_t, a_t) = \sum_{d \in D^m_t} v(s_t, d)$, where $a_t$ is the score adjustment action in state $s_t$, $D^m_t$ is the set of ads finally exposed in $s_t$ and $v(s_t, d)$ is the revenue value of displaying ad $d$ in $s_t$. We set $v(s_t, d)$ as the Generalized Second-Price after the actual sorting of the advertising items and recommended items.

- $P$: State transition models the dynamics of requests visiting and system information changes. The effect of $a_t$ on state transitions is reflected in: Different $a_t$ would lead to different ads in $s_t$, which would also affect the total number of ads that have been shown (which is a component of $s_{t+1}$). Similar ways of modeling state transitions have also been adopted previously in Cai et al. [9], Jin et al. [19] and Wu et al. [37].

Specifically, for the constraints:

- $C_T$: It is defined as the platform level constraint $C_{\text{platform}}$ - the advertising exposure constraint over a day (trajectory), and set discount factor $\gamma$ to 1, $C_T : \sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1}) \leq \alpha$.

- $C_S$: It is defined as the request level constraint $C_{\text{request}}$ - the advertising exposure constraint over each request (state), and set discount factor $\gamma$ to 1, $C_S : \sum_{d \in D^m_t} \frac{N^d_t}{N_t} \leq \beta, \forall i \in \mathbb{N}$.

With all the definitions above, an optimal policy $\pi^*$ is defined as follows:
We propose a constrained two-level reinforcement learning framework to address the constrained advertising optimization problem. The overall structure is shown in Fig. 2 and the framework is described in Algorithm 1. We split the entire trajectory into a number of sub-trajectories. The optimization task of the higher level is to learn the optimal policy \( \pi^* \) for selecting constraints for different sub-trajectories to maximize long-term revenue while ensuring that the constraint over the whole trajectory \( C_T \) is satisfied (Algorithm 1, Line 4). Given a sub-trajectory constraint \( C_{ST} \) from the higher level, the lower level is responsible for learning the optimal policy \( \pi_s(s_t; C_{ST}) \) over its sub-trajectory while ensuring that both the sub-trajectory constraint \( C_{ST} \) and the per-state constraints \( C_S \) are satisfied (Algorithm 1, Line 2 - 3). In this way, the original psCMDP optimization problem is simplified by decoupling it into the two independent optimization sub-problems.

By decoupling the adaptive exposure learning problem in such a two-level manner, we have higher adaptability and can quickly make response to dynamically changing e-commerce environments. This property is critical in online e-commerce environments since slower response would result in significant amount of monetary loss to the company. First, the platform-level constraint may vary frequently due to the company’s business strategic change. In this case, the lower level policies we have learned can be reused and only the higher level policy needs to be retrained. Second, the recommendation system or other components of the advertising system may change frequently, and our adjustment policy needs to be updated accordingly. In this case, we only need to retrain the lower level policies while the higher level part can be retained.

### 3.3 Solution: Constrained Two-level Reinforcement Learning

We propose a constrained two-level reinforcement learning framework to address the constrained advertising optimization problem. The overall structure is shown in Fig. 2 and the framework is described in Algorithm 1. We split the entire trajectory into a number of sub-trajectories. The optimization task of the higher level is to learn the optimal policy \( \pi^* \) for selecting constraints for different sub-trajectories to maximize long-term revenue while ensuring that the constraint over the whole trajectory \( C_T \) is satisfied (Algorithm 1, Line 4). Given a sub-trajectory constraint \( C_{ST} \) from the higher level, the lower level is responsible for learning the optimal policy \( \pi_s(s_t; C_{ST}) \) over its sub-trajectory while ensuring that both the sub-trajectory constraint \( C_{ST} \) and the per-state constraints \( C_S \) are satisfied (Algorithm 1, Line 2 - 3). In this way, the original psCMDP optimization problem is simplified by decoupling it into the two independent optimization sub-problems.

By decoupling the adaptive exposure learning problem in such a two-level manner, we have higher adaptability and can quickly make response to dynamically changing e-commerce environments. This property is critical in online e-commerce environments since slower response would result in significant amount of monetary loss to the company. First, the platform-level constraint may vary frequently due to the company’s business strategic change. In this case, the lower level policies we have learned can be reused and only the higher level policy needs to be retrained. Second, the recommendation system or other components of the advertising system may change frequently, and our adjustment policy needs to be updated accordingly. In this case, we only need to retrain the lower level policies while the higher level part can be retained.

#### 3.3.1 Lower Level Control

In the lower level, we have to address the sub-problem of learning an optimal advertising policy under a particular sub-trajectory constraint \( C_{ST} \) provided by the higher level part. In our approach, we convert the constraint of each \( C_{ST} \) straightforward into state level constraints to do more precise manipulating. We simplify the sub-trajectory optimization...
Weixun Wang, Junqi Jin, Jianye Hao, Chunjie Chen, Chuan Yu, Weinan Zhang, Jun Wang, Xiaotian Hao, Yixi Wang, and Han Li, CIKM’19, November 2019, BEIJING, CHINA
Jian Xu, Kun Gai

problem at the cost of sacrificing the policy optimality by treating the sub-trajectory constraint $C_{ST}$ as a state-level constraint. It is obvious that the original state-level constraint $C_S$ is more strict than $C_{ST}$. Thus, we can easily combine $C_{ST}$ and $C_S$ into a single state-level constraint $C_T$. Obviously once the state level constraint is satisfied, the higher level constraint would be satisfied at the same time. Thus, given a sub-trajectory constraint $C_{ST}$ by the higher level policy, we can directly optimize the lower level policy at the state level. One natural approach in CMDP is to guide the agent’s policy update by adding an auxiliary value related to the per-state constraint to each immediate reward (Equation 4). And, during policy update, both the current and the future penalty values are considered. However, in our lower level, since each transition satisfies the constraint $C_S$ independently, each action selection does not need to consider its future per-state constraints $C_S$. 

**Enforce Constraints with Auxiliary Tasks.** Considering the above reasons, we propose a method similar to auxiliary tasks [18] by adding an auxiliary loss function based on per-state constraints. We use $L_{RL}$ and $L_{CS}$ to denote the RL loss function and the per-state constraint $C_S$ loss function respectively. During training, the policy is updated towards the direction of minimizing the weighted sum of the above two:

$$\min L'(\theta) = w_1 L_{RL}(\theta) + w_2 L_{CS}(\theta)$$

(13)

where $w_1$ and $w_2$ are the weights, $\theta$ are the parameters of the value network. For example, for critic part in DDPG [24], the original critic loss function is:

$$L_{RL}(\theta) = [r + \gamma Q(s_{t+1}, \pi(s_{t+1}|\theta^*); \theta^*) - Q(s_t, a_t; \theta)]^2$$

(14)

and the additional loss function for per-state constraints can be defined as follows:

$$L_{CS}(\theta) = [Q(s_t, a_t; \theta^*) + \delta(s_t, a_t, s_{t+1}|C_S) - Q(s_t, a_t; \theta)]^2$$

(15)

where $\theta^*$ and $\theta^\prime$ are the online critic network parameters, actor network parameters and the target critic network parameters respectively and $\delta(s_t, a_t, s_{t+1}|C_S)$ is the function of $C_S$. The value of $\delta(s_t, a_t, s_{t+1}|C_S)$ is used to control the degree that Q-function is updated when the constraint is violated. For example, when we want to limit the pvr of each request close to 0.4, we can set $\delta(s_t, a_t, s_{t+1}|C_S) = -(pvr_t - 0.4)^2$, where $pvr_t$ is the pvr value of request $q_t$. Intuitively, the more $(s_t, a_t, s_{t+1})$ deviates from the target pvr, the more its corresponding Q-value will be decreased. Similar techniques have also been used to ensure the optimality of the expert demonstrations by using a margin classification loss [15].

**Constrained Hindsight Experience Replay.** To increase the sample efficiency, we propose leveraging the idea of hindsight experience replay (HER) [5] to accelerate the training of optimal policies for different sub-trajectory constraints. HER relieves the problem of sample inefficiency in DRL training by-reusing transitions, which can be obtained by using different goals to modify reward. We extend this idea to propose the constrained hindsight experience replay (CHER). Different from HER, HER does not directly revise the reward. Instead, it uses different constraints to define the extra loss $L_{CS}$ during training. The overall algorithm for training lower level policies under CHER is given in Algorithm 2. When we learn a policy to satisfy constraint $c_4$ (specific constraints on each state), it obtains the transition: $(s_t, a_t, r_t, s_{t+1}, c_2)$ (Algorithm 2, Line5 - 8). We can replace $c_4$ with another constraint $c_4'$ and then reuse those samples $(s_t, a_t, r_t, s_{t+1}, c_4')$ to train a policy satisfying constraint $c_4'$ (Algorithm 2, Line12).

3.3.2 Higher Level Control. The higher level task is to determine trajectory-level constraints for each sub-trajectory to maximize the expected long-term advertising revenue while satisfying the original trajectory constraint $C_T$. At each decision point, the higher level policy (we term as constraint choice policy, CCP) selects a constraint for the next sub-trajectory, and the corresponding lower level policy takes over and determines the ads adjustment score for each request within that sub-trajectory. After the lower level policy execution is finished, the accumulated revenue over that sub-trajectory is returned to the higher level policy as its abstraction immediate reward $r$. The above steps repeat until we reach the end of the trajectory, and then we obtain the actual percentage of ads displayed over the whole trajectory, which can be compared with the trajectory constraint as an additional reward $r^{ab}$ with weighted $w_2$:

$$r^{ab} = r^a + w_2 \times r^e$$

(16)

Similar to WebMDP [13], we use DQN [27] for the higher level policy optimization. More advanced RL techniques (such as CPO [1], SPSA [29]) can be applied as well. Note that our higher level control is similar to the temporal abstraction of hierarchical reinforcement learning (HRL) [7, 20]. However, in contrast to learning how to switch option [7] and alleviate the sparse reward problem [20] in HRL, our work leverages the idea of hierarchy to decompose different constraints into different levels.

---

4By satisfying the state-level constraint in the lower level, we reduce the optimization problem of the higher level into an optimization problem that only needs to consider the trajectory-level constraint.
4 EXPERIMENTS

4.1 Experimental Setup

Our experiments are conducted on a real dataset of the Chinese largest e-commerce platform Taobao, and the data collection scenario is consistent with the problem description in Section 3. In Fig. 1, we have demonstrated that the score adjustment of a product produced by the advertising system does not affect the selection and scoring of the candidate products produced by the recommendation system. It only influences the relative ranking of the ads compared with the recommendation products and affects the final mixed sorting results. Therefore, the online data collected from platform can be reused to evaluate the effects of score adjusting through resorting the mixture of the original recommended products and the regraded ad products. Similar settings can be found in related work [9, 19, 28, 39]. Specifically, we replay the users’ access logs in chronological order to simulate the users’ requests. The state of our psCMDP is represented by integrating the features of all candidate ads and the system contextual information. The action is defined as the score adjusting ratios for candidate ads. Finally, the reward and the satisfaction condition of each constraint are calculated accordingly following the definition in Section 3.2. All details can be found in the appendix.

4.2 Does CHER improve performance?

To verify the effectiveness of using CHER, we compare the impact of using CHER on the learning speed and stability with a baseline DDPG [24] under the same network structure and parameters.

Suppose \( C_S \) is the number of exposure ads for each request, and cannot exceed 5, so we set \( C_S \) to be consisting of 5 constraints. Each goal in \( PVR_i = 0.3, 0.35, 0.4, 0.45, 0.5 \) represents the expected average number of ads exposed per request. Intuitively, we can use the constraint as part of the input and use a network to satisfy all constraints [5]. However, considering the learning stability, we use 5 different networks to satisfy different constraints. Since we use DDPG, we add \( L_C \) to \( L_{\text{Critic}} \) during Critic training,

\[
L^C_{\text{Critic}} = L_{\text{Critic}} + w \times L_C
\]

\[
L_C = E_i(\left(-[PVR_i - PVR] + Q(s, a; \theta') - Q(s, a; \theta)\right)^2)
\]

where \( w \) is set to 10, and \( PVR_i \) is the percentage of ads exposed for \( i \)-th request \( q_i \). We set up 4 different random seeds, and the experiment results are shown in Fig. 3. We only show the results of \( PVR_i = 0.35, 0.4, 0.45 \) due to the space limit. The criterion for an algorithm will be better if its result is closer to the target constraint. We can find that, under different constraints, DDPG with CHER is better than DDPG in terms of training speed, achieving and stabilizing around constraints. In order to understand the rationality of the policy after training, we randomly sampled some user visits in the dataset. By recording the actions of adjusting scores of the advertisements in these user visits (Fig. 4), our approach learning can be intuitively understood as follows: if the advertising product has higher value (ecpm, price), then its score is adjusted higher.

4.3 Verify the Effectiveness of Constrained Two-level Reinforcement Learning

In order to verify the two-level structure can bring about an increase in revenue, we compare the performance of different methods under different platform-level constraints \( PVR=0.35, 0.41, 0.46 \) (the upper bound of the advertising rate of each day is 0.35, 0.41, 0.46 respectively, \( C_T=0.35, 0.41, 0.46 \) with the state level constraint fixed \( PVR_i = 0.5 \), the upper bound of the advertising rate of each request is 0.5, \( C_S = 0.5 \)). Since we are considering a new adaptive exposure mechanism, there are no existing approaches suitable for comparison. In our paper, we consider the following two approaches as baselines: 1). manual: the score of an advertisement is manually adjusted by human experts. 2). CHER + DDPG: a previous trained model of Section 4.2. It corresponds to a policy of using a fixed request-level constraint for the whole trajectory without adaptive adjustment. Since the performance of DDPG varies a lot, we add a complementary CHER to DDPG and use this optimized approach (CHER+DDPG) to attain a more stable \( PVR_i \).
Weixun Wang, Junqi Jin, Jianye Hao, Chunjie Chen, Chuan Yu, Weinan Zhang, Jun Wang, Xiaotian Hao, Yixi Wang, and Han Li, CIKM’19, November 2019, BEIJIN, CHINA

### Table 2: The Performance of Our Approach in One Day.

<table>
<thead>
<tr>
<th>Policy</th>
<th>hour</th>
<th>PVR</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>ours</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manual</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target</td>
<td></td>
<td>0.35</td>
<td>14321</td>
</tr>
<tr>
<td>manual</td>
<td></td>
<td>0.356</td>
<td>143121</td>
</tr>
<tr>
<td>CCP</td>
<td></td>
<td>0.356</td>
<td>143121</td>
</tr>
</tbody>
</table>

### Table 3: The Performance of Our Approach for each Hour.

<table>
<thead>
<tr>
<th>Hour</th>
<th>Revenue</th>
<th>CCP</th>
<th>CCP - π₁</th>
<th>PVR</th>
<th>CCP</th>
<th>CCP - π₁</th>
<th>Revenue / PVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>11556</td>
<td>15845</td>
<td>4289</td>
<td>0.01280837</td>
<td>0.01590928</td>
<td>230.8%</td>
<td>864562</td>
</tr>
<tr>
<td>9</td>
<td>15595</td>
<td>23422</td>
<td>7827</td>
<td>0.0162089</td>
<td>0.02192751</td>
<td>100%</td>
<td>1025727</td>
</tr>
<tr>
<td>10</td>
<td>20157</td>
<td>28979</td>
<td>8822</td>
<td>0.0184266</td>
<td>0.02321300</td>
<td>238.6%</td>
<td>962125</td>
</tr>
<tr>
<td>11</td>
<td>18221</td>
<td>24739</td>
<td>6518</td>
<td>0.01880709</td>
<td>0.02246692</td>
<td>238.6%</td>
<td>1093907</td>
</tr>
<tr>
<td>12</td>
<td>16777</td>
<td>18646</td>
<td>1869</td>
<td>0.01794808</td>
<td>0.01895375</td>
<td>238.6%</td>
<td>934751</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>18129</td>
<td>16023</td>
<td>2106</td>
<td>0.02096899</td>
<td>0.01851524</td>
<td>238.6%</td>
<td>846562</td>
</tr>
<tr>
<td>16</td>
<td>22913</td>
<td>20450</td>
<td>2463</td>
<td>0.02233828</td>
<td>0.01964052</td>
<td>238.6%</td>
<td>1052727</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>12919</td>
<td>11432</td>
<td>-1347</td>
<td>0.01914268</td>
<td>0.01718366</td>
<td>238.6%</td>
<td>764789</td>
</tr>
<tr>
<td>18</td>
<td>11424</td>
<td>9786</td>
<td>-1638</td>
<td>0.01633943</td>
<td>0.01428198</td>
<td>238.6%</td>
<td>699167</td>
</tr>
<tr>
<td>19</td>
<td>11586</td>
<td>10081</td>
<td>-1505</td>
<td>0.01570854</td>
<td>0.01391865</td>
<td>238.6%</td>
<td>737560</td>
</tr>
<tr>
<td>22</td>
<td>18362</td>
<td>15391</td>
<td>-2971</td>
<td>0.02780465</td>
<td>0.02398777</td>
<td>238.6%</td>
<td>660393</td>
</tr>
<tr>
<td>23</td>
<td>12720</td>
<td>10584</td>
<td>-2136</td>
<td>0.02291417</td>
<td>0.01988751</td>
<td>238.6%</td>
<td>555115</td>
</tr>
</tbody>
</table>

### Figure 5: Learning Curves Compared with Policy π₁.

### Figure 6: Learning Curves Compared with Policy π₂.

### Figure 7: Learning Curves Compared with Policy π₃.

**Does higher level control improve performance?** To distinguish different policies in the behaviour policy set, we use π₀, π₁, π₂, π₃, π₄ to refer to the different lower level policies (DDPG+CHER) previously trained in Section 4.2 under different platform-level constraints. PVR = 0.3, 0.35, 0.4, 0.45, 0.5. The temporal abstraction value is set to 1 hour, which means the higher level CCP makes a decision per hour. After selecting the sub-trajectory constraint, the behavior policy of the state level is activated for adjusting ads’ scores in the flowing hour with the sub-trajectory constraint fixed. In our experiments, we combine the double DQN architecture with the dueling structure to train CCP. The state of CCP consists of hourly information, such as the timestamp, hourly eCPM, PVR from 00:00 to current time. The objectives of the higher level policy are: (1) achieving approximately the same number of exposure ads with target PVR; (2) improving revenue as much as possible. Detailed results are shown in Table 2 and Fig. 5 - 7, in which we see the CCP can increase the revenue of each day compared to the manual and DDPG+CHER policies under the same constraint Cₚ. Therefore, we demonstrate that our approach learns to expose different numbers of ads in different time periods, which means that more ads are exposed when the value of the ads in a request is higher and fewer ads are shown in other time slots.

**Why higher level control can improve performance?** We analyze the finally exposed ads and all the corresponding candidate ads of each request within a day. First, we set the advertising rate of each

---

*In fact, a more fine-grained decomposition can lead to a better performance. However, we simply set the minimum temporal unit to 1 hour here to make the analysis of the improvement of CCP easier.*
We should show more ads
We should show fewer ads

with the daily PVR constraint satisfied. We conduct the experiments on the setup). However, we found the manual method could not ensure a relatively stable stability of the pvr constraint, so we omit the results for fair comparisons.

Figure 8: The changing curves of per-hour advertising rates with one day. Fix: The proportion of ads exposed on each request is fixed and set to 0.35; Oracle: After one day, we could figure out the best dynamical advertising rate available for each hour under conditions that satisfy the daily constraint: $PVR = 0.35$ through data analysis.

request to a fixed value 0.35. Then we calculate out the proportion of the finally exposed ads within each hour to the total number of ads in that day, which is represented as the Fix policy in Fig. 8. Keeping the total number of ads displayed in a day exactly the same, Oracle is calculated by resorting all the candidates of all requests together according to the scores and picks out the top 35% ads to display. Note that this Oracle policy shown in Fig. 8 is the best strategy available for displaying ads in one day. We can clearly find out that during the time period of hour 8 - hour 12, the advertising rate of the Oracle policy is more than 35%, which means that we should display more ads within this period to enlarge revenue. Accordingly, during hour 17 - hour 20 and hour 22 - hour 23, the advertising rate of the Oracle policy is less than 35%, which means that we should reduce the number of the unpromising ads and leave this opportunity to the more valuable ones. The detailed advertising performance of each hour is shown in Table. 3. We can clearly see that the revenue gap between the baseline policy and our approaches mainly appear on hour 8 - hour 12; Besides, our method can obtain more cost-effective advertising exposure within hour 8 - hour 12 and hour 15 - hour 16. Our method can dynamically adjust the advertising number corresponding to different time periods with the daily PVR constraint satisfied.

### 4.4 Online Results

Lastly, we also report the production A/B test experiments, which compared the performance of our approach to a currently deployed baseline (which displays a fixed number of ads to every user) in Taobao’s online platform. We conduct the experiments on the section of “guess what your like”, where a mixture of recommendations and advertisements are displayed to the users. Our method does not fix the numbers and positions of ads. We apply our designed mechanism to adaptively adjust the scores of each ad to different users so as to display different numbers of ads to different users in different positions. More details are illustrated in section 3.1. For a fair comparison, we keep the platform-level constraint the same for all approaches. As a result, we find that our approach does present different numbers of ads to different users in different positions satisfying the preset constraint overall. Besides, we observe 9%, 3%, and 2% improvements in RPM (Revenue Per Mille), CTR (Click-through rate) and GMV (Gross Merchandise Value) respectively, which indicates that our adaptive exposure mechanism not only significantly increase the revenue of the platform (RPM) but also the revenues of advertisers (GMV). We also introduced the detailed online implementation process in the appendix.

5 CONCLUSION

We first investigate the flaws in traditional E-commerce systems using fixed-positions to expose ads, and further propose more flexible advertising methods (Adaptive Exposure Mechanism) to alleviate the defects of fixed-positions. Further, we emphasize that there are a series of challenges when applying adaptive Exposure Mechanism in actual scenarios. We first model it as a psCMDP problem with different level constraints, and propose a constrained two-level reinforcement learning framework to solve this problem. Our framework offers high adaptability and quick response to the dynamic changing e-commerce environments. We also propose a novel replay buffer mechanism, CHER, to accelerate the policy training of the lower level. We have demonstrated that our designed adaptive Exposure Mechanism can provide more flexible advertising displaying methods while satisfying a series of constraints through offline simulation experiments and online verification. At the same time, we also verified that the constrained two-level reinforcement learning framework can effectively utilize the adaptive Exposure Mechanism to improve the platform revenue and user experience while satisfying the constraints.

REFERENCES

q-learning from demonstrations. In Proceedings of AAAI.


A APPENDIX

A.1 Discussion: Adaptive Exposure Current research on dynamic ad exposure focuses on sponsored search[6], stream advertising in news feeds[17], etc. Their dynamic ad exposure mainly refers to how to select the appropriate location and quantity to display the ad in the fixed optional ad positions[6], or how to dynamically insert the ad in the feeds through the user’s previous browsing process[17]. Compared with sponsored search[6], our mechanism does not limit the position of advertisements. Instead, it chooses the number and location of ads by means of score sorting, which means that our approach can bring more flexibility. Compared with stream advertising in news feeds[17] we consider the mixed-display scenario where both recommended products and advertised products are displayed altogether to the customers and their display orders are determined by their relative rankings.

A.2 Related Work

A.2.1 Bidding Optimization in Real-Time Bidding. Under the Real-Time Bidding (RTB) settings in E-commerce advertising, amounts of work have been proposed to estimate the impression values, e.g. click-through rate (CTR) [25] and conversion rate (CVR) [22], which help to improve the bidding effectiveness via predicting more precise impression values. Besides, the user impression analysis, bidding optimization is one of another most concerned problems in RTB, whose goal is to dynamically set a more appropriate price for each auction aiming at maximizing some key performance indicators (KPIs) (e.g. CTR) [36]. However, constraints are inevitable while solving optimization problems in real world bidding situations. So, smarter bidding strategies are needed for attaining higher KPI values (e.g. the cumulative impression value), which can be achieved through reinforcement learning techniques [9, 28, 39]. In these approaches, they optimize the bidding strategy under the fixed budget constraint and the budget will be reset at the beginning of each episode. Perlich et al. [28] and Zhang et al. [39] propose static bid optimization frameworks based on the distribution analysis of the previously collected log data. However, their approach can’t apply well to the setting in which the data distribution is unstable and will change from day to day in extreme circumstances. For this reason, Cai et al. [9] model the bidding problem as a MDP and consider the budgets allocation as a sequential decision problem. Experimental results show the robustness of their reinforcement learning approach under the non-stationary auction environments.

By contrast, we are the first to propose a more general deep reinforcement learning framework which takes more realistic business constraints into consideration. In our settings, we concentrate on the practical advertising with adaptive exposure problem. Not only do we consider the trajectory level constraint $C_T$, but also the state level constraint $C_S$. This is the main reason why the previous approaches are not applicable to our settings.

A.2.2 Constrained Reinforcement Learning. We are focusing on a constrained optimization problem and a number of researches have been done. One typical solution is the constrained reinforcement learning. Uchibe and Doya [35] propose a policy gradient...
algorithm which uses gradient projection to enforce the active constraints. However, their approach is unable to prevent the policy from becoming insecure at the beginning of the training. Later, Ammar et al. [4] propose a theoretically-motivated policy gradient method for lifelong learning under safety constraints. Unfortunately, they involve an expensive inner loop which contains an optimization of a semi-definite program, making it unsuitable for the DRL settings. Similarly, Chow et al. [12] propose a primal-dual sub-gradient method for risk-constrained reinforcement learning, which takes policy gradient steps trading off return for lower risk while simultaneously learning the trade-off coefficients (dual variables).

More recently, a number of DRL-based approaches have been proposed to address the constrained optimization problem. Achiam et al. [1] use the conjugate gradient method to optimize the policy. However, the computational cost will significantly arise as the constraint number increases, resulting in such approaches inapplicable. Tessler et al. [34] propose the Reward Constrained Policy Optimization (RCPO), which converts the trajectory constraints into per-state penalties and dynamically adjusts the weight of each per-state penalty during the learning procedure via propagating the constraint violation signal over the entire trajectory.

Our work tackle the multi-constraint problem from a different point of view and take the relationship between the different constraints into account. We decouple the original multi-constraints optimization problem into relatively independent single constraint optimization problems and propose a constrained two-level reinforcement learning framework. More importantly, our two-level framework is quite general and any state-of-the-art RL algorithms can be flexibly applied to learning procedures of both levels.

### A.3 Network structure and training parameters

#### A.3.1 CHER
Both the actor network and the critic network are four-layer fully connected neural networks, where each of the two hidden layers consists of 20 neurons and a ReLU activation function is applied on the outputs of the hidden layers. A tanh function is applied to the output layer of the actor network to bound the size of the adjusted scores. The input of the actor network and critic network is a tensor of shape 46 representative feature vectors of the request’s candidate ad items and the number of currently exposed items. The output of the actor network and the critic network are respectively 15 actions and corresponding Q-values. The learning rate of the actor is 0.001, the learning rate of the critic is 0.0001, and the size of the replay buffer is 50000. The exploration rate starts from 1 and decays linearly to 0.001 after 50000 steps.

#### A.3.2 Higher Level Control
DQN network has three-layer neural networks. The hidden layer consists of 20 neurons and a ReLU activation function is applied on the outputs of the hidden layer. Then we connect the hidden layer output to: 1) the nodes with the same number of actions, which is used to simulate the action advantage value $A$, 2) only one node, which is used to simulate the state value $V$. Finally we obtain $Q(s, a) = V(S) + (A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a'))$. The size of the replay buffer is 5000. We use the prioritized replay to sample the replay buffer. The learning rate is 0.0006. The exploration rate starts from 1 and linearly decays to 0.001 after 1000 steps. Also we set the discount factor $\gamma = 1$.

### A.4 Experimental Setup
Based on the log data, for each request, we collect 15 recommended products and their scores marked by the recommendation system as candidates for each request, and the information of 15 advertising products, such as: eCPM (effective cost per mille), price, predicted Click-Through-Rate (pCTR), and initial score. Since the actual amount of data is significantly large, we sample a part of the data for empirical evaluation, and verify that the sampled data has representativeness to the real data set. In both training and evaluation stages, we split the previously collected data by day and replay the requests in chronological order for simulation. We consider the data flow from 00:00 AM to the next day as a trajectory. At the beginning of each day, the number of ads has been displayed and number of requests have been reset to 0. At the end of each day, we count the daily number of ads displayed and make a judgement whether the trajectory-level constraint $C_T$ has been satisfied.

A state consists of 46 dimensions including the characteristics of the 15 candidate ads: eCPM, price, pCTR, and the number of exposed ads. Action is the coefficient to adjust scores for 15 ads, action = $\{\eta_1, \eta_2, \ldots, \eta_{15}\}$. After adjusting scores using actions, 15 candidate advertising commodities and 15 candidate recommended commodities are sorted based on the new scores. The reward is calculated according to the ads in the first 10 exposure items. We can replay the data to train and test the effect of our algorithm offline in two ways: 1) after ads adjusting scores, whether the quantity of ads in the 10 exposure items meets $C_T$ and $C_S$, and 2) the rewards of the exposed ads. Actually, the positions of ads have impact on user behaviors. E.g., the ads in front are more possible to be clicked, and so on. Hence the reward is defined as:

\[
r = \sum_d f_p(d) \times eCPM(d)
\]

where $eCPM(d)$ is the eCPM value of the ad $d$, and $f_p(d)$ corrects the eCPM by considering the influence of different positions. $f_p(d)$ is fitted using the real data. (Fig. 9)
A.5 Online Experiment

Due to the architecture of the online engine in Taobao, we replaced the gradient based DDPG with the widely used gradient-free Cross Entropy Method (CEM, an evolution based genetic algorithm) within the platform. When deploying our algorithm to the online environment, we consider two separate processes: (1) online serving and data collection; (2) offline training. For (1), we use Blink (an open source stream processing framework which is specially designed and optimized for e-commerce scenarios) to record the constantly updated online data. Besides, to fully explore the parameter space of CEM, we split the online traffic into a number of buckets and deploy different sets of parameter configurations at the same time. Different buckets are controlled by different parameters. After processing each user’s request, the newly produced data is recorded to corresponding data tables by Blink. For (2), a centralized learner will periodically update its parameters based on the latest recorded data, generate different sets of parameters for different buckets and synchronize them to the parameter server. At the same time, the online search engine deployed at each bucket also regularly request the latest parameters from the parameter server.