Large-Scale Personalized Delivery for Guaranteed Display Advertising with Real-Time Pacing

Zhen Fang[1*], Yang Li[2*], Chuanren Liu[3**], Wenxiang Zhu[1], Yu Zhang[1], Wenjun Zhou[3**]
[1] {fangzhen.pt,wenxiang.zwx,daoji}@alibaba-inc.com, Alibaba Group, Hangzhou, China
[2] yl837@drexel.edu, Drexel University, Philadelphia. USA
[3] {cliu89,wzhou4}@utk.edu, University of Tennessee, Knoxville, USA

***Abstract*—Guaranteed display (GD) has been a successful model for display advertising. Existing solutions usually model GD services as a crowd-level supply allocation problem. This formulation, however, not only ignores user heterogeneity within crowds, but also makes it difficult to incorporate individual-level constraints. In this paper, we present a large-scale system for personalized delivery in GD advertising services. A unique contribution is to model the supply allocation problem at the individual level that accounts for user-ad interactions. Therefore, our system can conveniently incorporate complex constraints, such as the priority of GD contracts, the display frequency of ads, and the effectiveness of ad slots arrangement. Moreover, we develop a real-time pacing strategy to fulfill GD contracts with smooth ad delivery and optimized ad performance, such as cost-per-click (CPC) and cost-per-action (CPA). Our system can be parallelized to efficiently compute the delivery solution with billions of decision variables. Using both offline evaluation and online A/B tests, we demonstrate that our solution outperforms previous methods in terms of both accuracy and efficiency.**

***Index Terms*—Display Advertising, Guaranteed Display, Supply Allocation, User Behavior, Large-Scale System.**

I. Introduction

Large-scale display advertising aims to deliver ads to the target audience under complex constraints. Particularly, guaranteed display (GD), which fulfills advertisement contracts specifying a minimum number of ad impressions, has drawn much attention in research [5, 24, 6, 8, 23]. Existing solutions usually model the GD service as a supply allocation problem, which seeks to meet the demands of multiple contracts with limited supply [9, 6, 3]. Solving the supply allocation problem relies on the partition of potential ad audience (i.e., users) into non-overlapping crowds according to traits (e.g., geographic location, gender, age) of the audience [26]. However, formulating and solving the GD allocation problem at the crowd level suffer from several limitations. First, it is notable that the heterogeneity of individual behaviors in each group of users prevents the crowd-level supply allocation from delivering ads to the right users in an accurate manner, which may, in turn, jeopardize the revenue for the advertisement publishers as well as the return on investment for advertisers. Second, advertisers may have complex requirements that impose user-level constraints on the GD allocation. For example, advertisers can specify the total number of ads to be displayed to any audience and the number of slots used to display the same ad for each arriving audience. As such, GD allocation optimized on coarse-grained user crowds can have undesirable effects or inferior performances in practice.

Developing a personalized ad delivery system to address the limitations of crowd-level GD, however, is non-trivial for large advertisement publishers. Specifically, there are three major challenges. **First**, the scale of user-level GD allocation depends on the Cartesian product of the supply and the demand for advertising services, which is prohibitively large. Consequently, the scale of the allocation problem may reach billions for large publishers serving thousands of contracts with various advertisers to millions of active users. Conventional optimization routines face daunting difficulties in solving the supply allocation problem of such scale. **Second**, users are heterogeneous in their interest and behavior patterns. Therefore, a refined personalized ad allocation plan should accommodate the diversity in user-ad interactions (e.g., click on an ad, add the promoted product into wish-lists, and make a final purchase). **Third**, GD allocation relies on supply forecasts but the forecast errors may have non-ignorable effects on serving ads if we model the GD allocation problem at user level. In comparison with forecasts of crowd-level user impressions, forecasts based on individual users may exhibit more uncertainties due to: (1) the randomness of users' behavior, and (2) the fact that user-level forecast errors will not cancel each other out. As a result, we need strategies with real-time feedback to monitor and control risks (e.g., under delivery) that may be caused by the forecast errors.

In this paper, we develop an efficient and effective system to serve GD contracts with personalized delivery at large scale. Figure 1 illustrates the architecture of our system, which consists of three subsystems: a Customer Relationship Management (CRM) module, an offline system, and an online server. The CRM module, as an interface between the advertisers and the publisher, manages the demand through GD contracts. Specifically, advertisers can customize their contracts by imposing requirements such as the target audiences, the number of advertising impressions to show, the total budget for the contract, and the information type (e.g., content/image/video) to display. Given all contract requirements, the offline system formulates the allocation problem that integrates impression forecasts and user-ad interaction estimates. After that, we compute the optimal allocation plan in parallel. This personalized allocation plan is then pushed

* Equal contribution ** Corresponding author

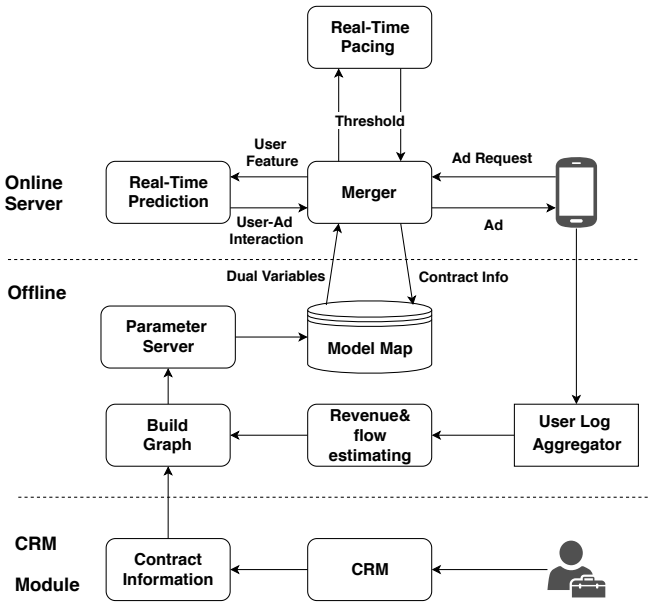


Fig. 1. System for personalized GD allocation.

to the online server that determines which ad to be displayed to an incoming impression in real time. To smoothly deliver the ads over the life cycle of contracts, the online server uses a pacing algorithm to dynamically match high-quality impressions, indicated by large values of estimated user-ad interactions (e.g., conversion rates), to the contracts in the derived allocation plan.

Our contributions can be summarized as follows. First, we improve the service quality for GD contracts by incorporating user-ad interactions, i.e., the probability that target users click on specified ads. Our formulation of the GD allocation problem is flexible to incorporate a variety of constraints. Second, we develop a distributed system to solve the large-scale GD allocation problem. Our system can provide personalized delivery solutions with billions of decision variables necessary for large publishers and platforms. Third, we develop a pacing strategy with real-time feedback to control uncertainty and risks for serving GD contracts online. While generating a smooth delivery plan to supply a wide range of audience impressions, our pacing strategy is designed to optimize ad performance by selecting high-quality impressions characterized by cost-per-click (CPC) and cost-per-action (CPA). Finally, our system has been evaluated with extensive experiments in real-world scenarios. The results show that our system can significantly improve the service quality for large-scale GD allocation in terms of both effectiveness and efficiency.

The rest of this paper is organized as follows. In Section II, we review the related work. In Section III, we present our problem statement. We describe technical details of the large-scale implementation in Section IV. The experimental setup, results, and discussions are presented in Section V. Finally, we conclude our work in Section VI.

II. Related Work

The GD allocation problem, commonly treated as an online matching problem to maximize the amounts of matched supply and demand, has been extensively studied in the literature [17, 20, 2, 21]. In their seminal paper, Karp et al. [17] presented a randomized algorithm to match supply nodes (i.e., user crowds) with unmatched neighbours in the demand side on a bipartite graph. To account for the fact that supply nodes arrive one by one online, Feldman et al. [13], Karande et al. [16] studied the online matching problem assuming users are drawn independently from distributions. Later, Devanur and Hayes [12] introduced the primal-dual framework to solve the allocation problem. Accordingly, some practical algorithms have been developed under this framework. For example, Vee et al. [25] utilized a particular subspace of the dual space to decide a compact plan for near-optimal online allocations. Motivated by this idea, Chen et al. [9] proposed the High Water Mark (HWM) algorithm by allocating an equal fraction from all eligible user crowds for each GD contract. The HWM algorithm, although very fast, sacrifices optimality. To improve the performance of HWM, Bharadwaj et al. [6] designed the SHALE algorithm by converting optimal dual into a good primal solution to the GD allocation problem. To account for contract requirements such as 'reach' and 'frequency', Hojjat et al. [14] proposed to use predetermined patterns generated by a complex heuristic. Recently, Zhang et al. [26] proposed a consumption minimization model, in which they use a greedy algorithm to minimize the user traffic satisfying all contract demands. Those approaches solve the GD allocation problem at crowd level, which limits their capacity of handling more fine-grained user-level ads effects and contract requirements.

Another line of related works consider feedback control aiming to solve the instability problem in real-time display advertising. Many researchers studied the budget pacing strategies that attempt to allocate ads smoothly throughout the life cycle of contracts in real-time bidding. The traditional approach is to use a feedback controller that monitors the difference between actual and desired allocation and use it as feedback to control the system dynamically [7, 22]. Zhang et al. [27] developed a similar mechanism called Water Level (WL) controller for real-time bidding (RTB) systems to improve the robustness of achieving the advertiser's key performance indicators. Different from the feedback controller, other systems have also been developed. Bhalgat et al. [4] introduced the online computation of dual variables to reach a wider range of audience for each contract. Lee et al. [18] presented an online algorithm which selects high-quality impressions and adjusts bid price by distributing the budget across time to improve conversion performance. Agarwal et al. [1] proposed to smooth budget pacing with traffic patterns.

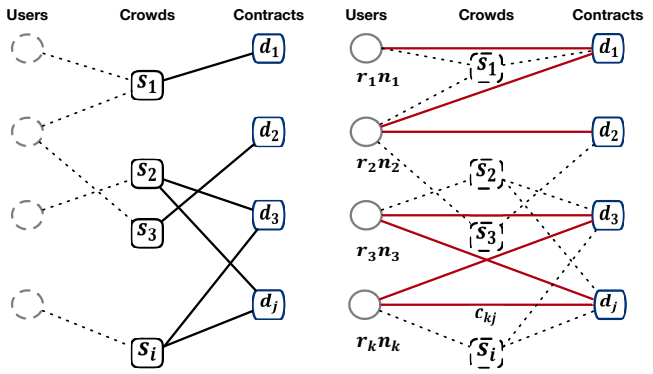
In this study, we solve the GD allocation problem at user level instead of crowd level. In this way, fine-grained contract requirements can be easily incorporated by adding user-level constraints. To reduce the computational complexity incurred by the large number of users, we develop a distributed system

that can efficiently optimize large-scale decision variables. We also develop a real-time feedback process for online GD allocation to eliminate uncertainties and risks incurred by user-level supply forecasts, while at the same time select high-quality impressions intended to improve advertiser experience and publisher revenue. To the best of our knowledge, our pacing strategy is the first to consider user-ad interactions and real-time dynamics in GD allocation problems.

III. Problem Statement

GD advertising is managed through contracts between advertisers and advertising publishers. The contract specifies the demanded number of impressions, d_j, where j is the index of the contract. If the publisher cannot meet the demand, there will be a penalty, p_j, for each under-delivered impression. The publisher who simultaneously serve many contracts will assign a weight of importance, V_j to the j-th contact, to prioritize the allocation of contracts.

The advertiser also specifies the target audiences, i.e., users who will be shown the ads. To this end, the publisher will group its users into different crowds. The supply s_i is defined as the impressions to be served in the i-th crowd.



(a) Crowd-level allocation (b) User-level allocation

Fig. 2. Users, crowds, and contracts for GD advertising.

It is natural to use a crowd-contract bipartite graph to describe the relationship between demand and supply as shown in Figure 2(a). If the j-th contract chooses the i-th crowd as (part of) its audience, there will be an edge between the supply node s_i and the demand node d_j. Existing studies solve the demand-supply allocation problem by determining x_{ij}, the optimal proportion of crowd i allocated to contract j under a few crowd-level constraints [6, 9].

A. Personalized Allocation in GD Advertising

In reality, crowd-level allocation in GD advertising is insufficient for several reasons. First, we observe that, even within the same crowd group, users often exhibit different interests and behavior patterns. Such fine-grained heterogeneity cannot be fully exploited in the crowd-level GD allocation framework, resulting in a less effective allocation plan. Second, it is very difficult to incorporate user-level constraints (e.g., the total amounts of ads displayed to any audience and the number of slots used to show the same ad to an arriving impression) for crowd-level allocation in a principled manner. The user-level constraints are necessary to accommodate the sophisticated requirements imposed by advertisers.

Therefore, we develop a system that delivers personalized allocation plan in GD advertising by optimizing the problem at the user level and incorporating user-ad interactions. As shown in Figure 2(b), an individual user is linked (in dashed lines) to a contracted ad through a user-crowd bipartite graph that maps the user to the crowd he belongs to. The edge between user k and contract j not only indicates that contract j chooses user k as its target audience but also means that there will be a score (denoted by c_{kj}) representing the likelihood that user k will click on the ad of contract j.

The user-level bipartite graph can characterize the actual process that individual users visit the advertising platform and view the delivered ad. In particular, note that an individual user can visit the advertising platform multiple times with each time being exposed to a number of ads. In other words, for each specified user k, there will be an r_k denoting the total number of times the user k visits our platform and a n_k representing the number of ads shown to the user each time. As a result, the total amount of impressions generated by the k-th user is $r_k n_k$ and the corresponding supply for contract j can be calculated as $S_j = \sum_{k \in Z(\Gamma(j))} r_k n_k$.

These detailed aspects allow us to consider user-level constraints in GD contracts. For example, to prevent the ad deliveries being perceived as intrusive, many advertisers require that the total number of ads to be displayed to a user cannot exceed certain times (frequency constraints) [19]. To improve the effectiveness of display advertising, many advertisers may also require that the same ad can only be displayed in one of the available slots for an arriving impression.

To determine the personalized deliveries (in solid lines) directly between the user and the ads, we need to answer the following questions: 1) how to model the user-ad interactions? 2) how to integrate the user-ad interactions in the formulation of GD allocation problem? and 3) how to compute the optimal solutions efficiently? We address (1) in Section III-B, (2) in Section III-C, and (3) in Section IV.

B. Modeling User-Ad Interaction

To effectively match potential users to ads, we use a predictive model to calculate the probability that a user clicks on a given ad. Three types of features have been considered: (1) user behavioral features based on records of their interactions with the advertising platform, such as visits, clicks, and purchase activities summarized within various time windows; (2) users' demographic features, such as age group, gender, and purchase levels; and (3) brand awareness features by aggregating brand-specific visits, clicks, and purchases.

Due to its successful applications in prior works [10, 11], we adopt a deep neural network (DNN) model to predict user-ad interactions. Once trained with historical behavior records, our model predicts whether or not a given user clicks after being exposed to the ad. Although we use users' click in our

study, our framework is flexible to incorporate other behaviors (e.g., put items into cart, or make a purchase).

To effectively mitigate the errors in traffic forecasts, we update the offline optimization with the most recent data and re-run the model in every hour. Accordingly, we set the life cycle of a contract as one hour with a pre-determined budget. To accommodate the update frequency of the offline optimization, we predict user-ad interactions hourly with the most recent customer behavior data. We use the area under the curve (AUC) to evaluate the prediction accuracy of the DNN model on user-ad interaction for seven consecutive days. As illustrated in Figure 3, the average AUC of the hourly-updated model in the test day consistently outperforms that of the daily-updated model by around 10%. This suggests that the hourly-updated predictions of user-ad interactions are sufficient to generate effective personalized allocation plans.

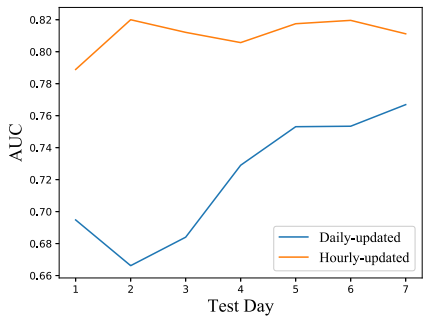


Fig. 3. The performance of hourly/daily-updated model.

In this study, we define user-ad interactions as some specific actions of a user in response to an ad delivery. For example, users may click on the ad, put promoted items into their wish lists, or make a final purchase. Among various types of user-ad interactions, we maximize users' clicks in the optimization for the following reasons: (1) Users' click is the contracted performance goal for many advertisers with the aim of raising brand awareness. For example, some companies would like to attain a certain amount of attention through GD contracts when they launch a new product; (2) Some ad publishers also serve as e-commerce platforms such as Amazon.com or eBay.com where a large proportion of demand for advertising come from the merchants (advertisers) on those platforms. These ad publishers with a dual role may suffer from less competitive merchants by optimizing alternative users' actions such as final purchase in the sense that it reduces merchants' incentive to convert clicks into sales [15].

C. Integrated Objective Function

We optimize an allocation plan by balancing three goals: maximizing representativeness, minimizing penalty, and maximizing potential customers. We denote the neighborhood relationship in the user-crowd bipartite graph by $Z(\cdot)$. That is, crowd i uses $Z(i)$ to specify the set of desired target users while user k can find the crowd group he belongs to using $Z(k)$. With the group of audiences defined, contract j uses $\Gamma(j)$ to specify the set of desired target crowds. The total eligible supply for the contract j is $S_j = \sum_{i\in\Gamma(j)} s_i$. Accordingly, we use $\Gamma(i)$ to represent the set of ads to be displayed to the i-th crowd. Therefore, we use $\Gamma(Z(k))$ to represent the set of contracts to be served to the k-th user. Likewise, all users targeted to contract j is $Z(\Gamma(j))$.

Users are heterogeneous in terms of user-ad interactions, and a high-quality user (i.e., a user who is more likely to click on a given ad as indicated by higher c_{kj}) is desired by all matched advertisers. Allocating more high-quality users to a contract than another would jeopardise overall advertiser satisfaction. We thus attempt to achieve a representative allocation which consists of the same proportion of impressions from each user allocated to a contract. Each contract j is associated with a penalty p_j that punishes the under delivery u_j, i.e. the number of impressions delivered less than the contracted demand d_j. We minimize the total penalty $\sum_j p_j u_j$ and maximize the the number of potential user-ad interactions (e.g., clicks). For advertisers, the click count measures the number of potential customers for an ad campaign. For the ad publisher, a large number of user-ad interactions indicate the relevance and effectiveness of ad serving to users, which in turn, can improve the satisfaction of users and advertisers. Taken together, we consider the following optimization problem:

$$\begin{aligned}\min \quad & \frac{1}{2}\sum_{j,k\in Z(\Gamma(j))} r_k n_k \frac{V_j}{\theta_j}(x_{kj}-\theta_j)^2 \\ & + \sum_j p_j u_j - \lambda \sum_{j,k\in Z(\Gamma(j))} r_k n_k x_{kj} c_{kj} \qquad (1)\end{aligned}$$

$$s.t. \sum_{k\in Z(\Gamma(j))} r_k n_k x_{kj} + u_j \geq d_j,\ \forall j \quad \text{(demands)}; \quad (2a)$$

$$\sum_{j\in\Gamma(k)} x_{kj} \leq 1,\ \forall k \quad \text{(supplies)}; \quad (2b)$$

$$x_{kj}, u_j \geq 0,\ \forall k,j \quad \text{(non-negativity constraints)}; \quad (2c)$$

$$r_k x_{kj} \leq f_j,\ \forall k,j \quad \text{(frequency constraints)}; \quad (2d)$$

$$n_k x_{kj} \leq 1,\ \forall k,j \quad \text{(slot constraints)}. \quad (2e)$$

The decision variable x_{kj} is the proportion of impressions generated by the k-th user of crowd $Z(k)$ that is allocated to contract j, which can also be interpreted as the probability that the impressions from the user is allocated to the contract. Here $\theta_j = \frac{d_j}{S_j}$, the ratio between the demand d_j and the total desired supply S_j of the j-th contract, serves as the representative allocation proportion for users matched to contract j. We minimize the deviation of the proposed allocation x_{kj} from the θ_j using a L_2-norm. A weighting scheme is also considered to account for the relative priority V_j (i.e., a larger value of V_j means the contract j is more important), the eligible supply S_j, and the value of θ_j of each contract. We calculate the total number of predicted user clicks on contract j as $\sum_{j,k\in Z(\Gamma(j))} r_k n_k x_{kj} c_{kj}$.

Constraints (2a) are demand constraints that define the under delivery as the gap between the contracted demand

and the proposed delivery. Constraints (2b) are supply constraints. They guarantee that the total allocated proportion of impressions should less than one. Constraints (2c) are non-negative constraints. To leverage the user-level optimization on a finer scale, we use new constraints that regulate the serving pattern of each ad to individual users. In specific, we consider frequency constraints (2d) where the same ad cannot be displayed to the same user more than f_j times; and slot constraints (2e) where only one slot can be used to display the same ad in each user visit.

The scale of personalized GD allocation can be extremely large due to the enormous size of the user-contract bipartite graph, making it challenging to solve the optimization problem and allocate ads in real time. In the next section, we develop a distributed algorithm to accelerate the offline optimization and a novel feedback control strategy for the online GD services.

IV. Large-Scale Implementation

In this section, we first illustrate our offline algorithm, which is designed to solve billion-scale allocation problems regularly. Then, we show our online serving strategy that selects high-quality impressions by adaptively updating a dynamic threshold for a smooth delivery.

A. Offline Optimization Phase

In the offline phase, we develop a distributed algorithm that takes advantage of some special properties of the optimal duals and data structure to solve the billion-scale allocation problem. Our offline algorithm consists of two stages. In stage one, optimal duals for personalized ads allocation are derived using a parameter server with multiple workers. In stage two, the optimal duals from stage one are converted to good primal solutions. We optimize the job scheduling for parallel computation by considering the topological properties of user-ad pairs in the allocation process. The optimized job scheduling accelerates the computation and makes the regular updates of the offline model possible for billion scale problems. We describe the two stages as follows.

1) Stage One: To derive an optimal allocation plan, we use duality theory to compute optimal duals for demand constraints and supply constraints, denoted as α_j^* and β_k^*, respectively, and then convert these optimal duals into good primal solutions similar to [6]. The challenge is that we have to solve the optimal duals in an efficient manner that accommodates distributed computing for our integrated objective function with user-ad interaction.

Specifically, using the Karush-Kuhn-Tucker (KKT) conditions, the values of α_j^* and β_k^* are derived by solving the following equations:

$$\sum_{k \in Z(\Gamma(j))} r_k n_k x_{kj}^* = d_j \tag{3}$$

and

$$\sum_{j \in \Gamma(Z(k))} x_{kj}^* = 1 \tag{4}$$

with

$$x_{kj}^* = \min\left(\frac{f_j}{r_k}, \frac{1}{n_k}, \max\left(0, \theta_j\left(1 + \frac{\alpha_j^* + \lambda c_{kj} - \beta_k^*}{V_j}\right)\right)\right).$$

For our personalized delivery optimization, the computational costs of solving α_j are relatively small compared to that of solving β_k because the number of signed contracts is usually less than the number of users. Therefore, we resort to distributed computation using a parameter server with multiple workers. The α_j are updated by the server and the tasks of calculating β_k are assigned to worker nodes.

Our system updates α_j using an approximate method instead of solving Equation 3 directly. Specifically, note that α_j are monotonically increasing in the iteration process with the upper bound $\alpha_j^{t+1} \leq \alpha_j^t + V_j\left(1 - \frac{d_j(\alpha^t)}{d_j}\right)$, where α^t refers to the value of α in the t-th iteration and $d_j(\alpha^t) = \sum_{k \in Z(\Gamma(j))} r_k n_k x_{kj}$. This allows us to update the value of α_j^{t+1} using the result of α_j^t and $d_j(\alpha^t)$ from the previous iteration. Therefore, we compute:

$$\alpha_j^{t+1} = \alpha_j^t + l V_j\left(1 - \frac{d_j(\alpha^t)}{d_j}\right), \tag{5}$$

where $l \in [0, 1]$ is the learning rate.

Algorithm 1 Stage One Algorithm

```
function UpdateBeta                                   ▷ Worker
    Pull all α from server
    for k ← 0 to len(Supply) do
        Update β_k with Equation 3 and 4
        for j ← 0 to len(Γ(Z(k))) do
            Update x_kj with new β_k
        end for
    end for
    Push all n_k r_k x_{k,j} to server
end function
function UpdateAlpha                                  ▷ Server
    Gether all n_k r_k x_{k,j} from worker
    for j ← 0 to len(Demand) do
        for t ← 0 to len(Iterations) do
            Update α_j^{t+1} with Equation 5
        end for
    end for
    Update all α_j to worker
end function
```

The corresponding pseudo-code is shown in Algorithm 1 using the aforementioned parameter server. The parameter server sets initial values for α_j and sends the supply data to the worker nodes, which calculate β_k and then push x_{kj} back to the server. On the worker side, we derive the value of β_k by solving Equation 4. It can be shown the value of $\sum_{j \in \Gamma(Z(k))} x_{kj}$ will increase as β_k decreases given α_j, λ and

c_{kj}. Thus a *binary search algorithm*[1] can be used to find the solution of Equation 4 with:

$$\beta_k \in \left[0, max_{j \in \Gamma(Z(k))}(\alpha_j + \lambda c_{k,j})\right].$$

2) Stage Two: The dual variables calculated in stage one are nearly optimal for the ad allocation problem. In stage two, we generate good primal solutions for delivery probability x_{kj}. To allocate ads to an impression, we first order the contracts by the eligible supply S_j from smallest to largest. Notice that in practice, a latter allocated contract may not be able to reach the fraction of impressions specified by Equation 3 if the eligible supply is depleted. Therefore, we calculate x_{kj} while maintaining a fraction of left-over inventory $\widetilde{s_k}$ that cannot be exceeded. To this end, we introduce a new variable ς_j, defined as the fraction of ad opportunities assigned to the jth contract in the allocation order, in place of α_j in the allocation:

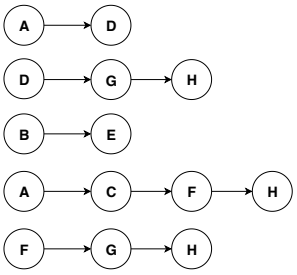
$$\sum_{k \in Z(\Gamma(j))} min(\widetilde{s_k}, r_k n_k \overline{x_{kj}}) = d_j \tag{6}$$

with

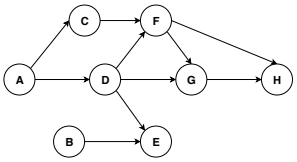
$$\overline{x_{kj}} = \min\left(\frac{f_j}{r_k}, \frac{1}{n_k}, \max(0, \theta_j(1 + \frac{\varsigma_j + \lambda c_{kj} - \beta_k^*}{V_j}))\right),$$

where β_k^* is computed in stage one.

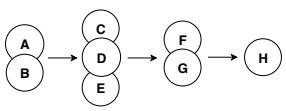
We follow the allocation order to calculate ς_j in a sequential manner, which would incur huge computational costs to fulfill the daily requirements from thousands of contracts for large ad publishers. Nevertheless, we observe that, for contracts without overlapping audiences, we can compute their allocations in parallel. For example, two contracts targeting users in Beijing and Shanghai respectively can be parallelized since they do not compete for the audience. Following this idea, we schedule the computing jobs as follows. First, we derive the personalized allocation order of contracts for each user by removing irrelevant contracts. Then, we build a directed acyclic graph (DAG) to capture the contract order for all users. Finally, we schedule the parallel computation based on the topological order in the DAG.



(a) Allocation order for five users.



(b) The DAG of contracts.



(c) The optimal schedule.

Fig. 4. Job scheduling for parallel computing.

[1]https://en.wikipedia.org/wiki/Binary_search_algorithm

Figure 4 gives an example for parallel job scheduling. The original allocation order for the five users is [A] → [H] with eight batches shown in Figure 4(a). Then we build the DAG graph in Figure 4(b) and get the final topological order [A, B] → [C, D, E] → [F, G] → [H] in Figure 4(c) with only four batches, which is just half of the original eight batches.

Therefore, we first compute the optimal schedule of parallel allocation with the final topological order of the DAG that includes all contracts. Then, we solve x_{kj} in parallel on the worker nodes. On the server, we set the initial value of ς_j as α_j^* from stage one and update ς_j by replacing α_j with ς_j in Equation 5. Given values of c_{kj}, dual variables ς_j and β_k^*, vector θ, priority vector V, and hyper-parameters λ, we calculate the delivery probability:

$$x_{kj} = min(\widetilde{s_k}, \overline{x_{kj}}),$$

where $\widetilde{s_k} = \widetilde{s_k} - x_{kj}$ with $\widetilde{s_k} = 1$ as the initial value of the available weight for user k. The obtained x_{kj} will be used for online allocation subject to a dynamic threshold.

B. Online Serving Phase

In previous studies, researchers use a roulette algorithm or greedy algorithm for GD problems for the online serving. Specifically, the allocation plan is determined at the beginning of the campaign using the computed delivery probabilities x_{kj} together with the impression forecasts. This makes the performance of both algorithms depend on the accuracy of impression forecasts, and the corresponding forecast errors may result in a higher risk of under delivery. To reduce this risk, we develop a real-time pacing method with dynamic thresholds to mitigate the uncertainty introduced in the offline phase and smooth the delivery of impressions during the life cycle of contracts. In this way, we can take into account the subtle temporal patterns of the traffic in allocation plan. We find that the minute-level traffic distribution is quite stable for a given ad. Therefore, we allocate impressions to ads at minute level during the life cycle of contracts to reduce uncertainty. The basic idea of our real-time pacing algorithm is that the portion of budget spent in every time slot should commensurate with the traffic distribution. In other words, we allocate ads to only a fraction of arriving impressions proportional to the real-time traffic. To achieve this, we adjust the delivery probability $x_{kj} = 0$ if the click probability c_{kj} is below a dynamic threshold δ_{tj}.

Specifically, the life cycle of a contract is discretized into T equal-length time slots and $spend_{jt}$, for $1 \leq t \leq T$, denotes the cumulative of actual spend of contract j from the beginning of the life cycle till the start of time slot t. The budget we would like to spend for a contract at time slot t is set to be proportional to the forecasted volume of traffic during that time slot. Hence, the allocation of cumulative budget from the beginning of the life cycle till the start of time slot t is

$$b_{jt} = \frac{\sum_{\tau=1}^{\tau=t} f_\tau}{\sum_{\tau=1}^{\tau=T} f_\tau} B_j \tag{7}$$

where f_τ is the forecasted total volume of traffic during time slot τ and B_j is the total amount of budget for contract j. Our algorithm finds a threshold δ_{tj} for click probability c_{kj} at time slot t in a way that contracts with low click probability are dropped from the allocation order. To do so, we construct an empirical distribution of click probability $q_{tj}(x)$ based on the historical data for each contract. Then, we adaptively serve contract j to a fraction a_{jt} of incoming impressions that are likely to interact with the ad as indicated by higher c_{kj}. The value of a_{jt} is updated at the start of each time slot by

$$a_{jt} = \begin{cases} a_{jt-1}(1+\eta) \wedge 1 & \text{if } spend_{jt-1} \leq b_{jt} \\ a_{jt-1}(1-\eta) \vee 0 & \text{if } spend_{jt-1} > b_{jt} \end{cases} \tag{8}$$

Here $0 \leq \eta \leq 1$ is an adjustment rate which is set at a constant 5% in our study. Once the a_{jt} is updated for time slot t, the corresponding threshold δ_{tj} can be calculated as

$$\delta_{tj} = argmin_x | \int_x^1 q_{tj}(x) - a_{jt} | \tag{9}$$

and contract j will be removed from allocation order if $c_{kj} < \delta_{tj}$. We illustrate the relationship of δ_{tj} and a_{tj} in Figure 5.

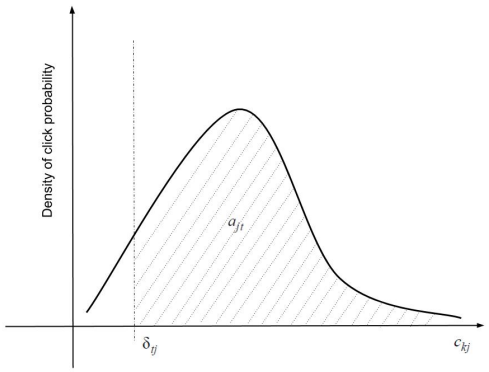


Fig. 5. Threshold selection for online pacing based on user-ad interactions.

We would expect that, on average, the larger the threshold is, the less budget will be spent in the unit of time for ad delivering, and vice versa. In an extreme case, a contract will not be delivered to any user regardless of the delivery probability if the threshold is one, and thus no budget will be spent. By doing this, we can adaptively serve ads to high-quality impressions.

Previous algorithms such as HWM and SHALE allocate impressions to contracts by ranking contracts based on delivery probabilities that are derived from traffic forecasts for the entire life cycle of contracts. Our proposed method is different from these algorithms in a way that we set a threshold for each delivery probability in some time interval and optimize these thresholds dynamically with real-time budget information. Moreover, we consider user-ad interactions in the pacing algorithm that allocates ads to high-quality impressions.

V. EXPERIMENTS

We conduct experiments with real-world data sets from Taobao, the largest e-commerce website in China, to demonstrate the effectiveness of our system. When customers use the Taobao App, contracted ads will be delivered to target audiences via webpage banners or recommendation spots called "Guess You Like". In the following, we first demonstrate the effectiveness and efficiency of our algorithm in solving α and β in the offline phase. Then, we present the results from online A/B testing with our system for large-scale personalized ad delivery with real-time pacing.

A. Offline Phase Evaluation

1) Setup: The offline performance was compared with two state-of-the-art benchmark algorithms: HWM [9] and SHALE [6] on an array of supply-demand graphs with different number of contracts and impressions. The performance metrics include delivery rate, penalty cost, L2 distance, and average cost per click (CPC), which are defined as follows.

- The **delivery rate** is calculated as

$$D = \frac{\sum_j (d_j - u_j)}{\sum_j d_j}, \tag{10}$$

  which represents the delivered impressions as a proportion of the total impressions.
- The **penalty cost** is the total penalty incurred to the publisher for being unable to deliver the guaranteed number of impressions. It is calculated as

$$P = \sum_j p_j u_j. \tag{11}$$

- The **L2 distance** measures the weighted deviation of a generated allocation from a representative allocation as defined in our objective function

$$L = \sum_{j,k \in Z(\Gamma(j))} r_k n_k \frac{V_j}{\theta_j} (x_{kj} - \theta_j)^2. \tag{12}$$

- The **average CPC** measures the average cost for advertisers to harvest a click through GD contracts

$$CPC = \frac{\sum_j B_j}{\sum_j \sum_{k \in Z(\Gamma(j))} r_k n_k x_{kj} c_{kj}}. \tag{13}$$

  where B_j is the total budget for the j-th contract.

Therefore, lower penalty cost, L2 distance, CPC, and higher delivery rate are desired. In addition, we also measure the running time for solving α and β of each job.

2) Performance: As shown in Table I and Figure 6, our system attains a near **50%** improvement of CPC compared to other methods in various scenarios. By reducing CPC in half, our system is particularly suitable for GD services with potentially improved revenue performance for both publishers and advertisers. Meanwhile, our system achieves similar (or slightly worse) performance on delivery rate and the corresponding penalty cost. Given the benefit from improved CPC and potential revenue, such loss of delivery might be

TABLE I
THE PERFORMANCE SUMMARY OF DIFFERENT ALGORITHMS WITH DIFFERENT DATA SCALES.

(Demand, Supply)	Algorithm	Performance Metrics				Computing Time (in secs)		
		Delivery Rate	Penalty Cost	L2 Distance	CPC	Solving α	Solving β	Total
(50, 40000)	HWM	0.528	1.66×10^7	1.18×10^6	2.99	0.096	0.378	0.474
	SHALE	0.530	1.66×10^7	1.18×10^6	2.97	0.312	1.216	1.528
	Our Method	0.525	1.65×10^7	1.18×10^6	**1.59**	0.001	0.848	0.849
(35, 100000)	HWM	0.441	7.28×10^7	4.50×10^7	2.78	0.452	1.140	1.593
	SHALE	0.457	7.13×10^7	4.48×10^7	2.67	2.964	7.941	10.905
	Our Method	0.446	7.24×10^7	4.54×10^7	**1.58**	0.001	2.604	2.605
(20, 300000)	HWM	0.579	1.89×10^7	3.20×10^7	3.36	1.593	3.569	5.162
	SHALE	0.699	1.40×10^7	2.56×10^7	3.89	1.607	3.557	5.165
	Our Method	0.662	1.41×10^7	2.38×10^7	**1.96**	0.001	1.454	1.455

negligible in practice. The ability to deliver representative allocations does not differ significantly between these methods as indicated by similar L2 distances.

For the benchmark algorithms, the HWM is the fastest since it runs only one iteration. However, the performance of HWM is relatively poor compared to those of the SHALE and our method. The SHALE slightly outperforms our method on delivery rate, since we consider a complex object function with more optimization goals. However, as shown in Figure 6, the SHALE has a significantly poor performance on CPC compared to our method.

As aforementioned, the computational bottleneck of the ad allocation problem is the time of solving α and β. We accelerate the computation of α by parallel job scheduling based on the topological order in the DAG, and β by using a distributed computing system with multiple worker nodes. Therefore, our method greatly reduces the running time to generate an optimal allocation plan, especially for large datasets.

3) Batch Reduction: We further show the effectiveness of our parallel algorithm in reducing the number of computation batches in an array of daily supply-demand scenarios during one week. First, we generate the original allocation order that contains all contracts to be served from the supply side and use $OriginBatch$ to denote the number of batches before task scheduling. Then, we use our algorithm to customize eligible contracts for each user and derive the reduced number of batches $ReduceBatch$ for parallel computing. We use the metric $Speed\_upRatio = \frac{OriginBatch}{ReduceBatch}$ to measure the speed-up ratio of our methods in Table II. Compared to the serial method used in SHALE, the task reduction using DAG can effectively accelerate the computation by 14 times.

4) Learning Rate: We investigate the convergence behavior of our method at different learning rates. Figure 8 shows that larger learning rates can accelerate the training process. However, there is a trade-off between optimality and speed. When the learning rate l is above 1.0, the convergence will be fast at the beginning but the ultimate delivery rate will be inferior as compared to the optimal level. In an extreme situation, if we set the learning rate to a large value, such as $l = 2.0$, the performance will deviate far away from the optimal solution. Our experiments suggest that the optimal learning rate will land between 0.5 and 0.7.

TABLE II
TASK REDUCTION OF BATCHES IN STAGE TWO.

Supply-demand Scale	Origin Batch	Reduce Batch	Speed-up Ratio
6.51×10^9	6009	458	**13.12**
5.11×10^9	5646	491	**11.50**
4.22×10^9	5467	463	**11.81**
4.21×10^9	5443	442	**12.31**
4.23×10^9	5262	392	**13.42**
5.36×10^9	7089	510	**13.90**
4.61×10^9	5606	357	**15.70**

5) Parameter Tuning: We test the performance of our method with different values of λ. Figure 9 shows the effect of λ on delivery rate and average CPC. We can see that the average CPC monotonically decreases as λ increases, suggesting that a larger λ is desired if we want to obtain a better CPC. It is also notable that the improvement of CPC by tuning λ is significant when λ is small, but becomes marginal as λ increases. The delivery rate is weakened as λ increases. Therefore, we make a trade-off between the average CPC and the delivery rate such that the delivery rate decreases by 1% to maintain enough average CPC.

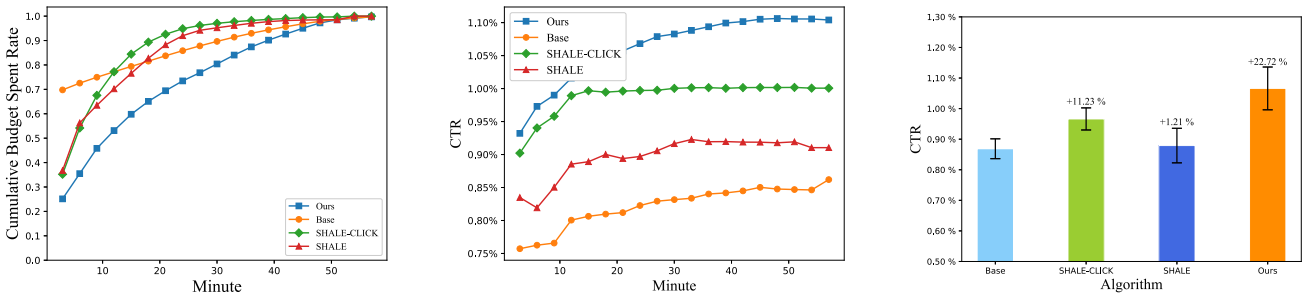
B. Online A/B Testing

We conduct online A/B testing on the Taobao website that fulfills GD contracts through a mobile app. During seven consecutive days, we use actual Click-Through-Rate (CTR) to evaluate the performances of various methods examined in the A/B testing. To make the treatment groups comparable to each other, we randomly divide the traffic into equally-sized streams. We demonstrate the effectiveness of our model by comparing it with SHALE, the state-of-the-art algorithm for GD allocation problems. We also compare our model with an enhanced SHALE model that takes click probability into consideration in the objective function of the allocation problem. Besides, a baseline model, which allocates ads to target audiences by simply ranking those audiences based on their purchase history, is also included in the comparison.

We first examine the pacing performance during the life cycle of contracts. Figure 7(a) shows that our proposed method outperforms all the compared algorithms as indicated by the smooth accumulation of actual budget spent. The baseline

(a) Data scale = (50, 40000) (b) Data scale = (35, 100000) (c) Data scale = (20, 300000)

Fig. 6. The comparison on delivery rate and average CPC.



(a) Pacing (b) CTR Curve (c) CTR

Fig. 7. Comparisons of different methods.

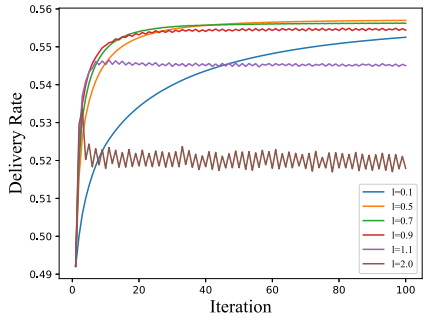


Fig. 8. The learning rate curve.

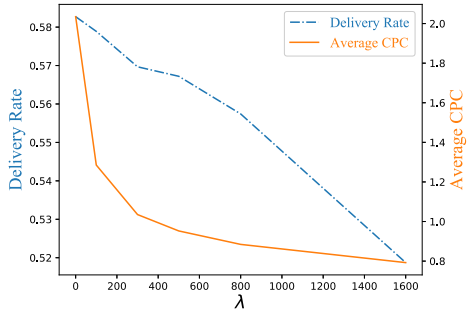


Fig. 9. Sensitivity of parameter λ.

model has the worst performance as it spends more than half of the entire budget in the first few minutes. The SHALE and the SHALE-CLICK perform better than the baseline model because they use an offline greedy algorithm for budget pacing. Our method makes further improvement in budget pacing with a real-time feedback control strategy.

Figure 7(b) shows the time series of average cumulative CTR of display ads using different algorithms. The SHALE-CLICK achieves a higher CTR compared to the baseline model and the SHALE by considering user-ad interactions in the offline optimization. Our method outperforms the SHALE by considering user-ad interactions in both offline optimization and online serving. The high CTR obtained by our method attenuates the cost for advertisers to harvest profitable clicks on their advertisements.

The bar plot in Figure 7(c) shows that the overall user conversion performance of our method has a relative increase of 22.7% against the baseline model, 21.2% against the SHALE and 10.6% against the SHALE-CLICK. We place error bars to represent 95% confidence intervals. T-tests indicate that these improvements are significant.

It is worthy of note that the proposed online pacing strategy needs a warm-up stage for threshold adjustments to match the traffic distribution in the first few minutes. This may lead to a marginal improvement of CTR at the beginning of the life cycle of the contract. As our method learns more about the contracts, the thresholds are updated to more accurate values and the performance of our model boosts accordingly.

VI. Conclusion

We present a large-scale system for personalized delivery of display advertising. We incorporate user-ad interactions in the allocation problem with complex constraints and then develop distributed algorithms to solve the optimization problem. Our real-time serving strategy supports budget pacing by controlling the risk of under delivery of the demanded ad impressions. Extensive experiments demonstrate the effectiveness and efficiency of our system. To the best of our knowledge, our advertisement serving system is the first to support the user-level impression allocation and the budget pacing in real time.

References

[1] Deepak Agarwal, Souvik Ghosh, Kai Wei, and Siyu You. Budget pacing for targeted online advertisements at linkedin. In *KDD'14*, pages 1613–1619, 2014.

[2] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014.

[3] Saeed Alaei, Esteban Arcaute, Samir Khuller, Wenjing Ma, Azarakhsh Malekian, and John Tomlin. Online allocation of display advertisements subject to advanced sales contracts. In *ADKDD'09*, pages 69–77, 2009.

[4] Anand Bhalgat, Jon Feldman, and Vahab Mirrokni. Online allocation of display ads with smooth delivery. In *KDD'12*, pages 1213–1221, 2012.

[5] Vijay Bharadwaj, Wenjing Ma, Michael Schwarz, Jayavel Shanmugasundaram, Erik Vee, Jack Xie, and Jian Yang. Pricing guaranteed contracts in online display advertising. In *CIKM'10*, pages 399–408, 2010.

[6] Vijay Bharadwaj, Peiji Chen, Wenjing Ma, Chandrashekhar Nagarajan, John Tomlin, Sergei Vassilvitskii, Erik Vee, and Jian Yang. Shale: An efficient algorithm for allocation of guaranteed display advertising. In *KDD'12*, pages 1195–1203, 2012.

[7] Shankar P Bhattacharyya and Lee H Keel. Robust control: the parametric approach. In *Advances in Control Education 1994*, pages 49–52. Elsevier, 1995.

[8] Bowei Chen, Shuai Yuan, and Jun Wang. A dynamic pricing model for unifying programmatic guarantee and real-time bidding in display advertising. In *ADKDD'14*, pages 1:1–1:9, 2014.

[9] Peiji Chen, Wenjing Ma, Srinath Mandalapu, Chandrashekhar Nagarjan, Jayavel Shanmugasundaram, Sergei Vassilvitskii, Erik Vee, Manfai Yu, and Jason Zien. Ad serving using a compact allocation plan. In *EC'12*, pages 319–336, 2012.

[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *DLRS'16*, pages 7–10, 2016.

[11] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys'16*, pages 191–198, 2016.

[12] Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *EC'09*, pages 71–78, 2009.

[13] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating $1 - 1/e$. In *FOCS'09*, pages 117–126, 2009.

[14] S Ali Hojjat, John Turner, Suleyman Cetintas, and Jian Yang. Delivering guaranteed display ads under reach and frequency requirements. In *AAAI'14*, pages 2278–2284, 2014.

[15] Yu Hu, Jiwoong Shin, and Zhulei Tang. Incentive problems in performance-based online advertising pricing: Cost per click vs. cost per action. *Management Science*, 62(7):2022–2038, 2015.

[16] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC'11*, pages 587–596, 2011.

[17] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC'90*, pages 352–358, 1990.

[18] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. Real time bid optimization with smooth budget delivery in online advertising. In *ADKDD'13*, 2013.

[19] Hairong Li, Steven M Edwards, and Joo-Hyun Lee. Measuring the intrusiveness of advertisements: Scale development and validation. *Journal of Advertising*, 31 (2):37–47, 2002.

[20] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.

[21] Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *SODA'12*, pages 1690–1701, 2012.

[22] Daniel E Rivera, Manfred Morari, and Sigurd Skogestad. Internal model control: Pid controller design. *Industrial & Engineering Chemistry Process Design and Development*, 25(1):252–265, 1986.

[23] H. Shen, Y. Li, and Y. Chen. Robust ad delivery plan for guaranteed display advertising. In *ICIA'14*, pages 1125–1130, 2014.

[24] John Turner. The planning of guaranteed targeted display advertising. *Operations Research*, 60(1):18–33, 2012.

[25] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *EC'10*, pages 109–118, 2010.

[26] Jia Zhang, Zheng Wang, Qian Li, Jialin Zhang, Yanyan Lan, Qiang Li, and Xiaoming Sun. Efficient delivery policy to minimize user traffic consumption in guaranteed advertising. In *AAAI'17*, pages 252–258, 2017.

[27] Weinan Zhang, Yifei Rong, Jun Wang, Tianchi Zhu, and Xiaofan Wang. Feedback control of real-time display advertising. In *WSDM'16*, pages 407–416, 2016.