A Request-level Guaranteed Delivery Advertising Planning: Forecasting and Allocation

Hong Zhang
keyzhzhang@tencent.com
Tencent

Lan Zhang*
zhanglan@ustc.edu.cn
University of Science and Technology of China

Lan Xu
lanxu@tencent.com
Tencent

Xiaoyang Ma
xiaoyangma@tencent.com
Tencent

Zhengtao Wu
wzt@mail.ustc.edu.cn
University of Science and Technology of China.

Cong Tang
tangcong@mail.ustc.edu.cn
University of Science and Technology of China.

Wei Xu
davidxu@tencent.com
Tencent

Yiguo Yang
justinyang@tencent.com
Tencent

ABSTRACT

The guaranteed delivery model is widely used in online advertising. The publisher sells impressions in advance by promising to serve each advertiser an agreed-upon number of target impressions that satisfy specific attribute requirements over a fixed time period. Previous efforts usually model the service as a crowd-level or user-level supply allocation problem and focus on searching optimal allocation for online serving, assuming that forecasts of supply are available and contracts are already signed. Existing techniques are not sufficient to meet the needs of today's industry trends: 1) advertisers pursue more precise targeting, which requires not only user-level attributes but also request-level attributes; 2) users prefer more friendly ad serving, which imposes more diverse serving constraints; 3) the bottleneck of the publisher's revenue growth lies in not only the ad serving, but also the forecast accuracy and sales strategy. These issues are non-trivial to address, since the scale of the request-level model is orders of magnitude larger than that of the crowd-level or user-level models. Facing the challenges, we present a holistic design of a request-level guaranteed delivery advertising planning system with careful optimization for all three critical components including impression forecasting, selling and serving. Our system has been deployed in the Tencent online guaranteed delivery advertising system serving billion level users for nearly one year. Evaluations on large-scale real data and the performance of the deployed system both demonstrate that our design can significantly increase the request-level impression forecast accuracy and delivery rate.

CCS CONCEPTS

• **Information systems → Computational advertising**; • **Theory of computation → Models of learning**.

KEYWORDS

Guaranteed Delivery Advertising; Request-level Impression Forecasting; Advertisement Allocation

ACM Reference Format:
Hong Zhang, Lan Zhang, Lan Xu, Xiaoyang Ma, Zhengtao Wu, Cong Tang, Wei Xu, and Yiguo Yang. 2020. A Request-level Guaranteed Delivery Advertising Planning: Forecasting and Allocation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403348

1 INTRODUCTION

A large portion of online display advertising is sold in a guaranteed delivery way. During a typical process of guaranteed delivery advertising, the advertiser places an order for a certain number of ad impressions to be shown to users with certain attributes over a specified period in the future. The publisher needs to accomplish the following three tasks in some (nearly) optimal way: 1) forecasting inventory (the eligible impression opportunities) in the specified period; 2) selling forecasted inventory by determining the maximum amount of impression opportunities that can be guaranteed for the order months/weeks in advance and signing the contract; 3) serving ad in real time when an actual user request arrives by determining which of the thousands of eligible contracts should be displayed for each opportunity in a split-second latency. Even a few percent improvement in the quantity of the sale or delivery rate can increase the publisher revenue by tens of millions of dollars, as well as increase the return on investment for advertisers. Overselling, however, will result in underdelivery and a penalty.

*Lan Zhang is the corresponding author. She is with the School of Computer Science and Technology, School of Data Science, University of Science and Technology of China.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403348

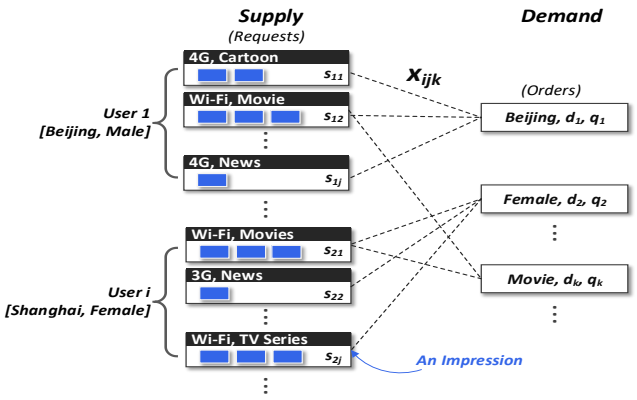

Figure 1: Example of user requests and orders. Each user may have heterogeneous requests with different numbers of impressions. The dash line indicates an impression in the request is eligible to the connected order.

Many efforts have been devoted to optimizing guaranteed delivery advertising [2, 4, 5, 7, 8, 10, 12]. However, industry trends show that existing approaches cannot fulfill the needs of advertisers, users and publishers in the following aspects: **First**, most work formulate and solve the forecasting and allocation problems at the crowd level [2, 4, 7, 8, 10, 12]. Some recent work begin to consider the user-level allocation [5]. In reality, advertisers always pursue more precise targeting. For example, an advertiser may target Shanghai female iPhone users watching a specified popular TV show through a Wi-Fi connection. As the example in Figure 1, each user may have heterogeneous requests and different requests may have different numbers of impression opportunities. Serving such advertisers requires not only crowd-level/user-level attributes but also request-level attributes, such as the name of the content and the network condition. Based on the statistics of Tencent online video advertising, 45% orders require request-level targeting. The scale of attribute combinations at the request level is several orders of magnitude larger than that at the user level, making existing methods inefficient to solve the forecasting and allocation problems. **Second**, previous work in the guaranteed delivery advertising mainly focus on how to efficiently achieve optimal allocation for online serving, assuming that impression forecasts and contracts are already available. Though impression forecast errors and sales strategy have non-ignorable effects on the performance of ad serving [2, 4, 5, 12], few work has delved into the large-scale impression opportunity forecasting and selling problems at the user level[10], let alone at the request level. **Third**, more customized and user-friendly serving constraints are desired by both advertisers and users. Most allocation algorithms are tailored for specific constraints, thus difficult to support customized constraints.

Motivated by the industry trends towards more precise advertising targeting and more user-friendly serving constraints, we design a **R**equest-level guaranteed delivery **A**dvertising **P**lanning system (**RAP**) from a more holistic perspective, which aims to optimize all three critical components including impression forecasting, selling and serving. Figure 2 illustrates the structure of our system. The main components focus on addressing the following two challenging problems:

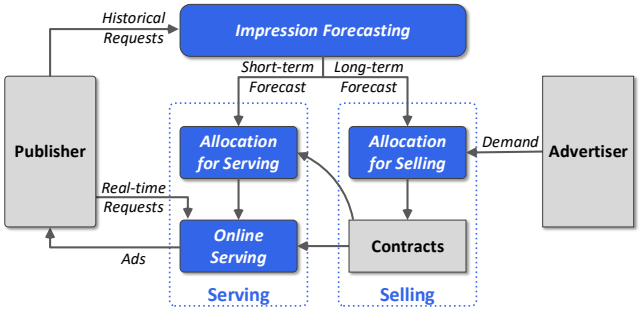

Figure 2: Structure of our request-based guaranteed delivery advertising planning system RAP.

1. *Large-scale request-level impression forecasting.* Large-scale high-dimensional data forecasting is a recognized challenging problem. In our case, the dimensionality of the request-level attribute combinations are orders of magnitude larger than that in the state-of-the-art work [10]. In Tencent advertising system, there are billion level impressions per day. The number of valid user attribute combinations is 63,901, and the number of valid request attribute combinations is 4,254,119, which results in an extremely large space for possible targeting. Moreover, the historical impressions are very sparse due to heterogeneous user behaviors. Therefore, forecasting request-level impression requires substantial computing resources and non-trivial features to characterize sparce and diverse requests.

2. *Large-scale request-level impression allocation for selling and serving under customized linear constraints.* Even we have got perfect forecasts, the scale of the allocation problem is the Cartesian product of tens of billions of impressions and thousands of contracts, which makes it very challenging to achieve (nearly) optimal allocation within a short latency. In the selling stage, the latency of allocating days of impressions should be less than a minute. In the serving stage, the latency of allocation should be less than hundreds of milliseconds. Arbitrary customized serving constraints further increase the difficulty of solving the large-scale optimization problem.

Our contribution can be summarized as follows.

(1) We present a holistic design of a large-scale request-level guaranteed delivery advertising planning system, including fine-grain impression forecasting and allocation optimization for selling and serving. Our design allows the granularity of the target attributes be much finer, from user-level to request-level, resulting in more precise advertising. It also supports more complex customized ad serving constraints. This added flexibility without sacrifices of efficiency, we expect, will greatly benefit advertisers, users, as well as the publisher.

(2) We propose a method combining clustering and tensor factorization to achieve accurate request-level impression forecasting and obviously reduce the time complexity. For large-scale allocation, we solve the optimization problem with a set of customized linear constraints and design a parallel algorithm based on parameter sharing and a GPU-accelerated method to significantly speed up the problem solving. We carefully design the objective functions for both impression selling and ad serving. Based on our algorithms, we leverage a lambda architecture to support selling and serving in

a short latency. We also incorporate real-time feedbacks to improve the allocation for online serving.

(3) **RAP** has been deployed in the Tencent online guaranteed delivery advertising system for nearly one year. We extensively evaluate our system on real data of billions of user requests and thousands of contracts. Both the experimental and online results show that our forecasting method outperforms state-of-the-art methods and our allocation solutions achieve a significant improvement on delivery rate and play rate, which have brought a considerable revenue growth.

2 RELATED WORK

Our work is related to two categories of existing work: time series forecasting and guaranteed delivery advertising allocation.

Time series forecasting. Traditional time series forecasting methods have been shown not suitable to deal with sparse high-dimensional data [10]. Recently, Matrix factorization models [1, 3] have been proposed to solve time series forecasting problems. Yu et al. [1] design a temporal regularized matrix factorization (TRMF) framework for high-dimensional noisy data forecasting. Ma et al. [10] propose a spatial temporal tensor factorization model (ST-TF), which achieves the state-of-the-art performance on large-scale high-dimensional data. In [10], it solves the crowd-level user visits forecasting and the scale of the forecasted tensor is ten thousands. In our request-level impression forecasting problem the scale of the tensor is orders of magnitude larger, making those existing methods infeasible.

Guaranteed delivery advertising allocation. The allocation problem is usually considered as a stochastic optimization problem, where there have been many foundational research work [6, 9, 11]. Inspired by these work, many practical algorithms have been proposed to address the allocation problem in guaranteed display advertising. Assuming a perfect forecast of future inventory, [12] can create a compact provably optimal allocation plan. Standard methods are too slow for large-scale commercial advertising applications. High Water Mark (HWM)[4] is a more efficient algorithm based on a greedy heuristic, which sacrifices optimality. To achieve a better tradeoff between running time and optimality, SHALE[2] incorporates optimal dual values to design a two-stage iterative algorithm so as to quickly approximate the optimal solution. Ali Hojjat et al.[7] [8] employ a column generation scheme to solve the problem with reach and frequency constraints. Zhang et al.[14] propose a consumption minimization model whose core is to minimize the user traffic consumed to satisfy all contracts. Recently, Fang et al.[5] present a personalized delivery framework, which models the allocation problem at the user level with individual frequency and slot constraints. All those work model the allocation at the crowd level or user level and many of them depend on the accuracy of inventory forecast. In our problem, we consider the request-level targeting, which significantly increase the scale and complexity of both forecasting and allocation problems. Besides, most allocation methods are tailored for ad serving and specified constraints. They can neither be directly adopted to solve the allocation in the selling stage (the problem formulation and objectives are different), nor easily support various customized serving constraints. Actually those user-level models with specific constraints like [5] are special cases of our request-level model with customized constraints.

3 PROBLEM AND SYSTEM OVERVIEW

A typical advertising planning problem can be represented with a bipartite graph, as shown in Figure 1. On the supply side, there are impression opportunities provided by user requests. The profile of these impression opportunities is composed of attributes of the user (e.g., gender, age, income level and area) and attributes of the request itself (e.g., time, network condition, content type and content name). User requests are heterogeneous, even for one single user. For example, the user i is a female user from Shanghai. Her requests of type j is for TV series through a Wi-Fi connection. In this type of request, she creates three impression opportunities. s_{ij} denotes the total amount of impressions provided by the requests of type j of user i. Note that a type of requests is a set of requests with the same attributes. On the demand side, each order targets a specific type of impressions. In Figure 1, the order k targets impressions in user requests for movies. It requests a certain amount d_k of impressions. q_k is the set of serving constraints (e.g., reach and frequency). An edge from a supply node to a demand node is added if and only if the impression represented by this supply node is eligible for the order represented by the demand node. In guaranteed delivery advertising, advertisers order the impression opportunities months/weeks in advance and specify the ad serving durations. So the publisher needs to forecast the inventory and allocate impression opportunities to each order in the selling stage as well as allocate actual impressions to each signed contract in the serving stage. x_{ijk} denotes the proportion of impressions in the requests of type j of the user i allocated to the order/contract k. The ultimate goal of the publisher is to optimize his revenue as well as the experience of both advertisers and users by selling and serving as many eligible impressions as possible under serving constraints.

To achieve this goal, we provide a comprehensive system design for request-level guaranteed display advertising planning (in Figure 2), named **RAP**, containing the following core components:

(1) Impression forecasting: given the historical impressions, before signing a contract, the publisher conducts a long-term forecasting to estimate the available inventory s_{ij} in the ad serving duration of orders; before serving ads, the publisher also needs a short-term forecast of s_{ij} to optimize the ad allocation.

(2) Impression allocation for selling: when a new order k arrives, given on the long-term forecast of s_{ij}, the publisher should find the optimal faction x_{ijk} to best fulfill the new demand d_k under the constraint q_k as well as minimize the negative impact on existing contracts.

(3)Impression allocation for serving: to achieve optimal ad serving, the publisher needs to generate a compact allocation plan x_{ijk} based on the short-term forecast of s_{ij}. When a request arrives, given a set of eligible contracts, the allocation plan and the real-time feedbacks, the publisher must decide the optimal contracts to serve the impressions in the request.

4 IMPRESSION FORECASTING

Both optimal selling and serving of guaranteed delivery advertising are based on the impression forecasting. In this section, we present our efficient request-level impression forecasting method.

In general, guaranteed delivery advertising is sold on a daily basis, so in our forecasting problem we take the day as the smallest unit of time. Let s_{ij}^t denote the total impressions of the requests of

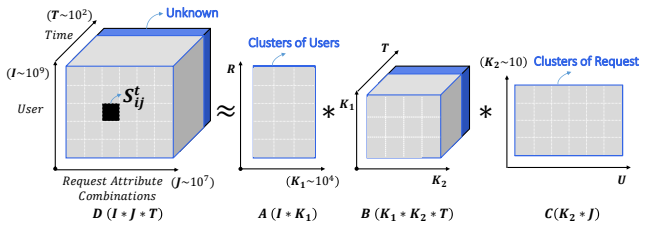

Figure 3: Request-level impression opportunity forecasting based on clustering and tensor factorization.

type j of the user i on the t-th day. The total inventory on the t-day is a set $D^t = (s_{ij}^t | i \in I, j \in J)$. I and J are sets of indices of users and request types respectively. Given the historical N-day inventory $D^1, D^2, \cdots, D^N$, our goal is to predict the future M-day inventory $D'^{N+1}, D'^{N+2}, \cdot, D'^{N+M}$, which should minimize the loss function $L(D'^{N+1}, \cdot, D'^{N+M})$.

Intuitively, the loss function can be defined as the L_1 difference between the real impressions and the forecasted impressions:

$$L(D'^{N+1}, \cdots, D'^{N+M}) = \frac{1}{M}\frac{1}{|I|}\frac{1}{|J|}\sum_{t=1}^{M}\sum_{I}\sum_{J}|s'^{N+t}_{ij} - s^{N+t}_{ij}|,$$

where s'_{ij} is the number of forecasted impressions.

Theoretically, we can solve the forecasting problem by minimizing the loss function following the idea of tensor factorization. As depicted in Figure 3, we can represent impressions $D'^1, D'^2, \cdots, D'^{N+M}$ by a 3-rd order tensor $\boldsymbol{D} \in \mathbb{R}^{I*J*T}$ with three modes, namely user, request attribute combination (request type) and date. Here I and J respectively denote the numbers of users ($\sim 10^9$) and request attribute combinations ($\sim 10^7$). T means there are T days data in total. Then s_{ij}^t is an element of $\boldsymbol{D}$. We can predict the unknown inventory $\{s_{ij}^t | t > N\}$ (the blue part in tensor $\boldsymbol{D}$ Figure 3) by exploring the optimal tensor factorization that minimizes the loss function. However, the loss function is impractical and existing tensor factorization models (e.g., the state-of-the-art work ST-TF[10]) cannot be directly adopted. The reason is that the dimension of the request-level tensor $\boldsymbol{D}$ in our problem is several orders of magnitude larger than that in ST-TF[10] and much sparser, making existing models infeasible.

To deal with the high dimensionality and sparsity, we incorporate clustering into tensor factorization, and improve the loss function according to the targeting of existing contracts.

Tensor Factorization with clustering. As shown in Figure 3, we decompose the original tensor $\boldsymbol{D}$ into two time-independent factor matrices $\boldsymbol{A} \in \mathbb{R}^{I*K_1}$ and $\boldsymbol{C} \in \mathbb{R}^{K_2*J}$, and a smaller tensor $\boldsymbol{B} \in \mathbb{R}^{K_1*K_2*T}$. We treat the newly added dimension K_1 as the clustering of users and $\boldsymbol{A}$ as the conditional probability distribution of each user in the user clusters. Similarly, we treat the newly added dimension K_2 as the clustering of user requests and $\boldsymbol{C}$ as the conditional probability distribution of each request attribute combination in the request clusters. Then we can consider the tensor $\boldsymbol{B}$ as the joint probability distribution of the clusters of users and clusters of requests changing with time. In this way, we decompose the original forecasting problem into two clustering problems and one lower-dimensional forecasting problems

Reweighting forecast errors for target attribute combinations. Considering that advertisers are interested in the impressions of target attribute combinations, the accuracy of every s'_{ij} shouldn't weight equally. Let γ be a set of target user and request attribute combinations determined by contracts. $S^t(\gamma) = \sum_{(i,j)\in\gamma} s_{ij}^t$ defines the actual total impressions of the target attribution combinations in γ on the t-th day. $S'^t(\gamma)$ is the estimated value corresponding to $S^t(\gamma)$. $\pi(\gamma)$ is the probability distribution of target attribute combinations. Then we can have the reweighted loss function to measure the forecast error for target attribute combinations:

$$L_R = \frac{1}{M}\sum_{t=1}^{M}\sum_{\gamma\in\Gamma}\pi(\gamma)|\sum_{(i,j)\in\gamma} s'^{N+t}_{ij} - \sum_{(i,j)\in\gamma} s^{N+t}_{ij}| \quad (1)$$

This loss indicates the possible underdelivery and is our major minimization objective.

User clustering. To obtain matrix $\boldsymbol{A}$, we put all individual users into groups ($\sim 10^4$) determined by popular target user attribute combinations in existing contracts. For example, we put female iPhone users from Shanghai into the same user group. Let $\mathbb{G}$ be set of all user groups. $G(i) \in \mathbb{G}$ represents the group of user i. The total impressions of user group g on the t-th day is $\sum_{\{i:G(i)=g\}} s_{ij}^t$, which contains denser impressions and is easier to forecast. Based on the user clustering, we can define the following loss measuring the structural error of forecasted impressions of user groups:

$$L_U = \frac{1}{M}\frac{1}{|\mathbb{G}|}\frac{1}{|J|}\sum_{t=1}^{M}\sum_{J}\sum_{g\in\mathbb{G}}|\sum_{\{i:G(i)=g\}} s'^{N+t}_{ij} - \sum_{\{i:G(i)=g\}} s^{N+t}_{ij}|. \quad (2)$$

This loss is used to prevent the model from over-fitting on the target attribute combinations of historical contracts.

Request clustering. To efficiently obtain the matrix $\boldsymbol{C}$, we cluster actual daily impression vectors of each user into K_2 clusters. Here, daily impression vectors of a user are daily histograms of his/her actual impressions in the popular target attribute combinations (that are predefined based on the statistics of historical contracts). Let the set of all users' daily impression vectors be V, which will be divided into K_2 clusters $V_1, V_2, \cdots, V_{K_2}$. Let μ_c be the center vector of cluster c. Σ_c^{-1} represents the covariance matrix of cluster c. Then the objective function of clustering can be defined by Euclidean distance as

$$L_C = \sum_{c=1}^{K}\sum_{x_i\in V_c}(x_i - \mu_c)^2. \quad (3)$$

If we consider a hybrid soft clustering, the objective function of the clustering task can also be defined by the Mahalanobis distance as $L_c = \sum_{c=1}^{K}\sum_{x_i\in V_c}(x_i - \mu_c)^T\Sigma_c^{-1}(x_i - \mu_c)$. We conduct clustering to minimize L_C and obtain the matrix $\boldsymbol{C}$. In our implementation, we determine the number of clusters using the algorithm in [13] and get 15 clusters by k-means. We use two months of real historical data (from October to November in 2018) to evaluate the consistency of the clustering results. The weekly clustering results show that at least 12/15 cluster centers overlap between any two weeks. It conforms to our design that the clustering is relatively stable and time-independent. Therefore, the matrix $\boldsymbol{C}$ doesn't need to be updated frequently.

Multi-objective loss function and forecasting. Combing Eq.(1), Eq.(2) and Eq.(3), we obtain the complete objective:

$$L_F = L_R + \lambda_u L_u + \lambda_c L_c. \quad (4)$$

Here, λ_u and λ_c are parameters balancing multiple objectives. Since the scale of tensor $\boldsymbol{B}$ is much smaller than the original tensor $\boldsymbol{D}$, we can use a deep spatial-temporal tensor factorization model (ST-TF) in [10] to forecast unknown values in $\boldsymbol{B}$ (the blue part in tensor $\boldsymbol{B}$ in Figure 3) to minimize the objective function L_F. The forecasted values indicate the joint probability distribution of user group and request cluster in the future. Now we have obtained matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ and tensor $\boldsymbol{B}$, thus we can calculate the blue part in the original tensor, which are the forecasted M-day inventory.

Implementation. In our implementation, we do not need to build the original tensor based on all historical users. We only focus on the recently active users, which result in a smaller I. For the deep learning model training, we sample one from every thousand users within a time window as the training samples to reduce the size of training data. Our design significantly reduces the time complexity for large tensor factorization and mitigates the difficulty of high-dimensional sparse data forecasting. The forecast accuracy of our deployed system is 88% $\sim$ 92%, which significantly benefits the request-level impression selling and serving.

5 IMPRESSION ALLOCATION

Now we have obtained the forecasted impressions s_{ij}. In this section, we will introduce how to optimally allocate these impressions to advertisers' contracts in the selling and serving stages. First, we will present the formulation of the basic allocation optimization problem. Then we design a novel method to efficiently solve a large-scale optimal allocation. In the end, we define the allocation problems for ad selling and serving, and present the system design.

5.1 Basic Allocation Problem Formulation

Recall that the ad allocation problem is usually modeled as a variant of the bipartite matching problem with some additional constraints, as the example in Figure 1. In guaranteed delivery advertising, a demand contract k usually has a penalty coefficient p_k for under-delivery u_k (i.e., the number of impressions delivered less than d_k). We need to determine the optimal allocation x_{ijk}, that is the fraction of impressions s_{ij} is allocated to contract k. An allocation is feasible if it satisfies the basic demand and supply constraints [2] as well as some additional constraints like frequency. The optimal allocation is a feasible allocation that minimizes some objective function. A typical objective of guaranteed delivery advertising is a tradeoff between maximizing the representativeness and minimizing the penalty. We aim to support arbitrary customized linear constraints and achieve efficient problem solving for large-scale problems, especially impression selling and ad serving. Here, taking frequency constraint as an example, we define a basic optimal allocation problem as follows:

$$\min \quad f(x_{ijk}) = \frac{1}{2}\sum_k \sum_{i,j\in\Gamma(k)} \frac{V_k s_{ij}}{\theta_k}(x_{ijk}-\theta_k)^2 + \sum_k p_k u_k \quad (5)$$

$$\begin{aligned} s.t. \quad & \forall k \quad \sum_{i,j\in\Gamma(k)} x_{ijk}s_{ij} + u_k \geq d_k && \text{demand constraint} \\ & \forall i,j \quad \sum_{k\in\Gamma(i,j)} x_{ijk} \leq 1 && \text{supply constraint} \\ & \forall i,k \quad \sum_{j\in\Gamma(k)} x_{ijk}s_{ij} \leq q_k && \text{frequency constraint} \\ & \forall i,j,k \quad x_{ijk} \geq 0, u_k \geq 0 && \text{non-negativity constraint} \end{aligned}$$

Here, $\Gamma(k)$ is the neighborhood of demand node k, likewise, $\Gamma(i,j)$ is the neighborhood of supply node ij. In the objective function, the first term is the non-representativeness that measures the L_2 distance from the allocation x_{ijk} to a target θ_k. Similar to existing work, we set $\theta_k = \frac{d_k}{\sum_{(i,j)\in\Gamma(k)} s_{ij}}$. V_k is the relative priority of the contract k. The second term in objective function is the total penalty for underdelivery.

Constraints. The frequency constraint controls the maximum/minimum number of times each user should see the ads from the same contract. In our design, the contract frequency constraint can be easily replaced by other linear constraints, such as the advertiser frequency constraint (the maximum/minimum number of times each user should see the ads from the same advertiser), slot constraint (the maximum number of ads from the same contract/advertiser in each user request), reach (the minimum number of unique individuals who should see an advertiser's ads) and daily demand constraint of contracts.

5.2 Optimization Algorithm

The allocation problem in Eq.(5) is a convex optimization problem with linear constraints. There exist a set of solutions $x_{ijk} = 0$ that hold the constraints. Therefore, we can find the optimal solution for the primal problem by solving its dual problem based on the KKT conditions. For problem in Eq.(5), the Lagrangian function is

$$\begin{aligned} & L(x,u,\alpha,\beta,\gamma,\eta,\psi) \\ &= \frac{1}{2}\sum_k\Big(\sum_{i,j\in\Gamma(k)} \frac{V_k s_{ij}}{\theta_k}(x_{ijk}-\theta_k)^2 + \sum_k p_k u_k\Big) \\ &- \sum_k \alpha_k\Big(\sum_{i,j\in\Gamma(k)} x_{ijk}s_{ij} + u_k - d_k\Big) + \sum_i\sum_j \beta_{ij}\Big(\sum_{k\in\Gamma(i,j)} x_{ijk}s_{ij} - s_{ij}\Big) \\ &+ \sum_i\sum_k \gamma_{ik}\Big(\sum_{j\in\Gamma_j(k)} x_{ijk}s_{ij} - q_k\Big) - \sum_i\sum_j\sum_k \eta_{ijk}x_{ijk} - \sum_k \psi_k u_k. \end{aligned}$$

According to the stationary conditions $\frac{\partial L}{\partial x_{ijk}} = 0$ and $\frac{\partial L}{\partial u_k} = 0$, we have $x_{ijk} = \theta_k(1 + \frac{\alpha_k - \beta_{ij} - \gamma_{ik} - \eta_{ijk}}{V_k}), 0 \leq \alpha_j \leq p_j$. Because $\eta_{ijk} = 0$ or $x_{ijk} = 0$, the allocation is

$$x_{ijk} = \max\{0, \theta_k(1 + \frac{\alpha_k - \beta_{ij} - \gamma_{ik}}{V_k})\}. \quad (6)$$

To solve the dual variables $\alpha_k, \beta_{ij}, \gamma_{ik}$, we can obtain the gradients of these dual variables: $\frac{\partial L}{\partial \alpha_k} = d_k - \sum_{(i,j)\in\Gamma(k)} x_{ijk}$, $\frac{\partial L}{\partial \beta_{ij}} = \sum_{k\in\Gamma(i,j)} x_{ijk} - 1$, $\frac{\partial L}{\partial \gamma_{ik}} = \sum_{j\in\Gamma_j(k)} x_{ijk}s_{ij} - q_k$. Using a coordinate descent or gradient descent method, we can solve the problem iteratively until the objective function converges. In each iteration we first calculate α, β, γ and then get x_{ijk}. Since every linear constraint has a corresponding dual variable, we can solve optimization problems with other linear constraints in the same manner.

5.3 Efficient Algorithm Implementation

Theoretically, the gradient descent algorithm is able to find the optimal allocation. When it comes to the real applications, however, the extremely large scale of the problem poses severe challenges to the implementation of the algorithm.

Observations. In the case that user i is not eligible for contract k, we have $\forall j, x_{ijk} = 0$. Because we have $\gamma_{ik} = 0$ or $\sum_{j\in\Gamma_j(k)} x_{ijk}s_{ij} = q_k$, then $r_{ik} = 0$ in this case. In the case request j is not eligible for contract k, we have $\forall i, x_{ijk} = 0$. Based on the statistics of our deployed system, $\frac{|\{x_{ijk}|x_{ijk}\neq 0\}|}{|I||J||k|} < 2.7\%$, and $\frac{|\{\gamma_{ik}|\gamma_{ik}\neq 0\}|}{|I||k|} < 3\%$. Therefore, the dual variable γ_{ik} are very sparse, so as the edges in the bipartite graph. Moreover, the scale of α_k is relatively small. Only the variable β_{ij} is dense due to the reason that almost every user i has a number of requests so that $s_{ij} \neq 0$. These observations motivate us to implement our algorithm by leveraging the power of parallel computing and GPU-accelerated computing, which significantly speed up the problem solving.

Parallel algorithm. We employ a group of workers to efficiently solve this problem in a distributed manner that each worker computes a subset of β_{ij} locally and shares α_k and non-zero γ_{ik} with each other through a parameter server. Since the scale of α_k is small and γ_{ik} is very sparse, the cost for parameter sharing is small. Specifically, all users are divided into w groups and each group is assigned to a worker. Then the computation is conducted parallelly as the following loop: 1) each worker downloads the newest α_k and non-zero γ_{ik} from the parameter server; 2) each worker computes x_{ijk} for its own group of users according to Eq.(6) and updates and local β_{ij}; 3) each worker computes and upload $\Delta\alpha_k$ and non-zero $\Delta\gamma_{ik}$ to the parameter server; 4) the parameter server aggregates $\Delta\alpha_k$ and $\Delta\gamma_{ik}$ from all workers to generate new α_k and γ_{ik}. This loop continues until the objective function converges.

GPU-Accelerated Computing. For each worker, computing the value of x_{ijk} (Eq.(6)) one by one is the most time-consuming step. To further speed up the computation, we treat x_{ijk} as a tensor and packed all parameters α_k, β_{ij} and γ_{ik} into three equal size tensors. In this way, each worker can efficiently compute the tensor of x_{ijk} by leveraging the power of GPU. Considering that x_{ijk} is very sparse, based on the edges in the bipartite graph, we can compress the tensor to only retain those non-zero dimensions. In this way, the computation cost is significantly reduced.

5.4 Allocation for Selling

In the selling stage, orders of advertisers arrive one at a time. The publisher allocates forecasted inventory to each new order and sign the contract. The major objective is to best fulfill the demand of existing contracts. The secondary objective is to maximize the saleable impressions for a new order. To deal with months of impressions and thousands of contracts in a short latency, we leverage a lambda architecture consisting of three layers: offline allocation, online allocation and a serving layer for responding to orders.

Offline allocation for selling. The offline allocation has sufficient time to search the global optimal solution for signed contracts. When selling months of impressions, a better temporal smoothness is an important guarantee for a better pacing for ad serving. Therefore, we add a daily demand constraint and a penalty for daily underdelivery to the basic allocation problem in Eq.(5). The objective function of offline allocation for selling is

$$\begin{aligned} \min \quad & f(x_{ijk}) - \sum_t \sum_{k\in\Gamma(t)} p_k^t w_k^t \\ s.t. \quad & \forall k, t \quad \sum_{i,j\in\Gamma(k,t)} x_{ijk}s_{ij} + w_k^t \geq d_k^t. \end{aligned} \tag{7}$$

Here, d_k^t is the guaranteed delivery amount of impressions for contract k on the t-th day. It can be specified by the advertiser or publisher. w_k^t is the underdelivery of contract k on the t-th day. p_k^t is the penalty coefficient of contract k on the t-th day. p_k^t is usually a decreasing function of t. We solve this problem using the GPU cluster based parallel algorithm in Section 5.3.

Online allocation for selling. The online allocation processes a newly arrived order in a sub-minute latency. The publisher aims to best fulfill the demand of the new order with minimal negative impact on the allocation of existing contracts. We add objective and constraints for the new order to the basic problem:

$$\begin{aligned} \min \quad & f(x_{ijk}) - \sum_I \sum_J y_{ij} \\ s.t. \quad & \forall i, j \sum_{k\in\Gamma(i,j)} x_{ijk} + y_{ij} \leq 1, \sum_{j\in\Gamma(y)} y_{ij}s_{ij} \leq q, \quad y_{ij} \geq 0 \end{aligned} \tag{8}$$

Here, y_{ij} is the proportion of the supply s_{ij} assigned to the new order. q is the frequency constraint of the new order. Based on the offline global optimal solution, this problem can be solved with a small number of iterations, so as to achieve a short response latency.

5.5 Allocation for Serving

During serving, user requests arrive one at a time and each request provides several impressions. The publisher allocates each impression to a contract in real time to optimize the objectives. In our design, we add a secondary goal to the basic allocation problem (Eq.(5)) to improve the total click through rate (CTR). The objective function is

$$\min \quad f(x_{ijk}) - \lambda \sum_k \sum_{i,j\in\Gamma(k)} \omega_k x_{ijk} ctr_{ijk}. \tag{9}$$

Here ctr_{ijk} is the CTR for contract k on the impressions s_{ij}. ctr_{ijk} is provided by the CTR prediction component of an advertising system. ω_k be the weight of CTR for contract k. λ is the hyper parameter balancing different objectives.

Based on the forecasted impressions one day in advance, we generate an allocation plan using the method in Section 5.3 to guide the ad serving. To better cope with the impression forecast errors and real-time demand changes caused by newly arrived orders, we need frequent corrections of the allocation plan according to the real-time feedbacks. According to Eq.(6), let $\rho_{ijk} = 1 + \frac{\alpha_k - \beta_{ij} - \gamma_{ik}}{V_k}$, we have $x_{ijk} = \max\{0, \theta_k\rho_{ijk}\}$. Here ρ_{ijk} is determined by the bipartite graph. It needs to be updated every minute. $\theta_k = \frac{d_k}{\sum_{(i,j)\in\Gamma(k)} s_{ij}}$ needs to be updated according to the real-time d_k and s_{ij}. Towards minimizing the objective function, online serving takes the following steps. 1) We generate a bipartite graph based on the forecasted impressions and signed contracts. It is sufficient to update the bipartite graph every hour considering that the basic unit of time for both impression forecast and demand is the day. 2) According to the real-time (millisecond level) statistics of impressions and

contracts, we update d_k, s_{ij} and θ_k. 3) Given the bipartite graph and real-time d_k and s_{ij}, the online allocation component solves the optimization problem (Eq.(9)) and outputs tensor x_{ijk} and tensor ρ_{ijk}. This computation is conducted every minute. 4) The online serving component takes as input ρ_{ijk} and θ_k to compute the final x_{ijk} and serve ads accordingly.

6 SYSTEM EVALUATION

We have implemented RAP and deployed it in the Tencent online guaranteed delivery advertising system for nearly one year. This system serves billion level users with a variety of ads, including video ads, in-feed ads, splash ads, etc. The deployment of RAP has brought a significant revenue growth.

6.1 Datasets

In this section, we extensively evaluate RAP with real impression and contract data of Tencent online video site.

Dataset for forecasting. There are a billion of impressions per day. Each impression has four types of user attributes and 17 types of request attributes. User attributes include platform (with 7 options such as iPhone and PC), area (with 396 options such as Shanghai), age and gender. Request attributes include content type (with more than 300 options such as movie and TV), content name (with more than 3,400,000 options such as "Spider-Man" and "Friends"), network (5 options such as Wi-Fi and 4G), target time (with 48 options, every half an hour is an option), etc. The training data is from June 1, 2018 to November 30, 2019 and the testing data is from December 1, 2019 to December 31, 2019.

Datasets for allocation. To evaluate the allocation for selling, we use 650,223 sampled impressions and 5,753 contracts on January 1, 2020. To evaluate the allocation for serving, we use two datasets on December 31, 2019. A large dataset contains 726,390 sampled impressions and 5,782 contracts. A small dataset contains 9,087 sampled impressions and 104 contracts from Shanghai.

Unless otherwise specified, evaluations were conducted on a server with a Tesla P40 GPU and a Intel Xeon CPU E5-2680 v4.

6.2 Impression Forecasting

Models for comparison. The evaluation results in [10] show that traditional time series forecasting methods are not competent to deal with sparse high-dimensional data. Hence, we implement the following state-of-the-art high-dimensional time series forecasting models for comparison:**ST-TF**[10], **TRMF**[1], a **CNN** model and **LSTM**. The structures and parameters of those models are all consistent with that in [10].

If we consider all valid attributes of impressions, the number of attribute combinations will be too large for previous models to handle. For fair comparison, we first consider only the three most popular attributes, namely platform, area and content type (*PAC*). Then we consider the five most popular attributes, namely platform, area and network, gender and content type (*PANGC*). In this end we evaluate our method on all target attribute combinations of real contracts.

Forecasting impressions N days in advance on *PAC* attribute combinations. Let $N = \{1, 2, \cdots 28\}$, Figure 4 illustrates forecast accuracy of different methods changing against N. Table 1 presents their 28-day average forecast accuracy. The results show that our design achieves significantly better accuracy than TRMF, CNN and LSTM. Overall, the performance of RAP is better than ST-TF. The 28-day average accuracy of RAP is 87.76% while that of ST-TF is 86.53%. When N is small, ST-TF has slightly better accuracy. As N increases, the accuracy of RAP gradually exceeds that of ST-TF.

Method	Accuracy	# of Parameters	Training Time (s)
RAP	**0.8776**	137,462	1,393
ST-TF	0.8653	61,850	942
TRMF	0.8139	1,800,879	1,127
CNN	0.8028	174,748	815
LSTM	0.6858	8,716	1,213

Table 1: 28-day average forecasting accuracy for different models on PAC attribute combinations.

Performance on *PANGC* attribute combinations. The scale is 15 times larger that in the *PAC* case. Facing the high-dimensional tensor, TRMF, CNN and LSTM fail to converge within an acceptable training time. Figure 5 illustrates the training process of ST-TF in the *PAC* and *PANGC* cases. In the *PAC* case, the loss of ST-TF steadily declines and converges after 700 steps. In the *PANGC* case, after a decline the loss oscillates around 0.2 and has difficulty to converge. Adding two attributes (network and gender) decreases the accuracy of ST-TF by about 6%. The more attributes are considered, the worse ST-TF performs. The results show that those state-of-the-art methods cannot handle high-dimensional sparse data, thus are not capable of solving our request-level impression forecasting task.

Performance on all target attribute combinations. In this case, we consider all attribute combinations that have been targeted by real contracts. Except RAP, all other models fail to generate forecast results. Table.2 presents the forecast accuracy of RAP. When N equals 1, 7 and 14, the accuracy is 90.13%, 91.76% and 89.87% respectively, which are even slightly better than the accuracy in the *PAC* case. **In our deployed system, the forecast accuracy on all target attribute combinations is** $88\% \sim 92\%$**. The results verify that our design can achieve efficient and accurate large-scale request-level impression forecast.**

Comparison with sampling methods. As aforementioned, existing deep learning based methods do not work when considering all attributes combinations. In industrial applications, it is common to adopt sampling results on historical data as forecasts. Thus, we compare our method with the following widely used sampling methods: 1)Traditional sampling (**TS**) samples a certain proportion of daily historical data; 2)Smooth sampling (**SS**) uses a 30-day sliding window to sample the average value of multi-day historical data. Table.2 presents the forecast accuracy of different sampling methods on the *PAC* attribute combinations and all target attribute combinations. Our method achieves a clear improvement compared with two sampling methods and performs the best in all cases.

6.3 Impression Allocation

1) Allocation for Selling

Recall that RAP leverages a lambda architecture and solves the offline allocation problem (in Eq.(7)) and online allocation problem (in Eq.(7)) to optimize the impression allocation for orders. We compare our selling strategy with the following methods:

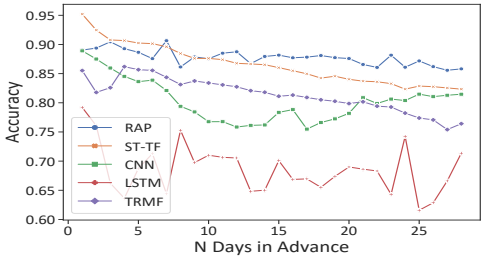

Figure 4: Impression forecast accuracy N days in advance on *PAC* attribute combinations.

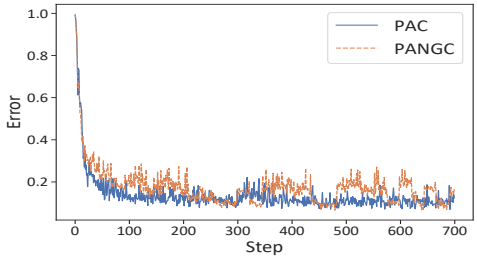

Figure 5: The training process of ST-TF in the PAC and PANGC cases.

Method	Contract Targets			PAC		
	1	7	14	1	7	14
RAP	90.13	91.76	89.87	89.04	90.65	87.95
SS	74.03	73.91	74.27	74.03	73.91	74.27
TS	87.84	86.10	87.61	86.98	84.39	77.62

Table 2: Forecast accuracy N-day in advance on different target attribute combinations.

	Time (s)	# of Saleable Impressions
FIFO	0.1	182
Heuristic-HWM	1	3688
RAP	30	4906

Table 3: Average number of saleable impressions allocated for new orders by different methods.

•**FIFO**: a newly arrived order has a lower priority than all singed contracts. Therefore, keeping the allocation to signed contracts unchanged, only remaining eligible impressions can be assigned to the new order.

•**Heuristic-HWM**: We adopt HWM[4] to allocate impressions to a new order. Specifically, we first determine the upper and lower bounds of the impressions available for the new order. The upper bound is the total impression count, and the lower bound is the result of FIFO. Then we give the new order a reservation amount between the upper and lower bounds by binary search to turn the selling problem into a series of typical delivery problems. We solve each delivery problem by HWM until we find the maximum available inventory that does not increase the underdelivery rate of signed contracts.

Given the real 650,223 impressions and 5,753 contracts on January 1, 2020, we randomly select 100 orders from the 5,753 contracts as new orders and let other contracts remain signed. Table. 3 presents the average number of saleable impressions allocated to each new order by different methods. FIFO performs the worst. **Compared with Heuristic-HWM, RAP increase the number of saleable impressions for new orders by 33%, which implies 33% revenue increase.** The allocation time cost of RAP for each new order is 30 seconds, that of Heuristic-HWM is 1 second. In our deployed system, sub-minute latency is acceptable for most orders, therefore RAP has brought a significant revenue improvement. For some latency-sensitive orders, we use Heuristic-HWM to achieve an immediate response.

2) Allocation for Severing

We evaluate the following metrics for different serving methods:

•Underdelivery rate: represents the under-delivered impressions as a proportion of the total guaranteed demand. It is the main minimizing objective and defined as $U = \frac{\sum_k u_k}{\sum_k d_k}$.

•Play rate: represents the delivered impressions as a proportion of the total impressions. A higher play rate represents a better utilization of ad opportunities, thus higher profit of the publisher. The play rate is defined as $P = \frac{\sum_k \sum_{i,j\in\Gamma(k)} s_{ij} x_{ijk}}{\sum_i \sum_j s_{ij}}$.

•L_2 distance: measures how much the generated allocation deviates from a representative allocation as defined in the objective function $L_2 = \frac{1}{2} \sum_k \sum_{i,j\in\Gamma(k)} \frac{V_k s_{ij}}{\theta_k} (x_{ijk} - \theta_k)^2$.

•Click through rate (CTR): is the total clicks divided by the total number of delivered impressions.

•Time cost: includes the initialization time cost and average time cost of each iteration.

Methods for comparison. We implement the following state-of-the-art methods and compare them with our method: **HWM[4]**, **SHALE[2]** and ALI[5]. Specially, we consider three variants of ALI[5]: 1) the original **ALI** uses a gradient descent method and a coordinate descent method to compute the dual variables α and β respectively; 2) **ALI_CD** uses only a coordinate descent method; 3) **ALI_GD** uses only a gradient descent method. We also consider four variants of RAP: 1) **RAP_CD**: uses a coordinate descent method to solve dual variables α, β and γ of the basic problem in Eq.(5); 2) **RAP_GD**: uses a gradient descent method to solve the basic problem ; 3)**RAP_CD_ CTR**: uses a coordinate descent method to solve the refined problem in Eq.(9) with the secondary objective to maximize the CTR. **RAP_GD_CTR**: users a gradient descent method to solve the refined problem.

Figure 6 and Figure 7 illustrate the underdelivery rate and play rate changing with iterations on the large dataset (726,390 impressions and 5,782 contracts). The results on the small dataset are quite similar. Table. 4 and Table. 5 present all metrics of different methods after 100 iterations. Since HWM has only one iteration, it costs the least time but results in the worst underdelivery rate. The three variants ALI and SHALE achieve similar underdelivery rate, play rate and L_2 distance. But the variants of ALI cost much less time than SHALE. As shown by the obvious gaps in Figure 6 and Figure 7, **the variants of RAP achieve significant improvements compared with existing methods and have the lowest underdelivery rate and highest play rate. For our deployed system, the underdelivery rate is** 3% ∼ 5%**.** For example, the underdelivery rate of RAP_GD_CTR is 1.34% on the large dataset, which is 4.34% lower than that of ALI_GD. The play rate of RAP_GD_CTR is

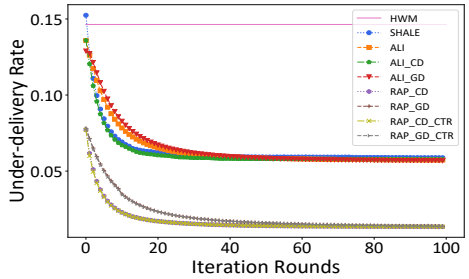

Figure 6: The underdelivery rate changes with iterations on the large dataset.

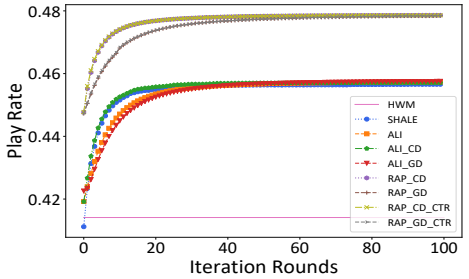

Figure 7: The play rate changes with iterations on the large dataset.

Algorithm	U	P	CTR	L_2	Init Time	Avg Time
HWM	0.1463	0.4141	0.0312	0.0292×10^4	114.8751	-
SHALE	0.0588	0.4566	0.0312	6.0951×10^4	114.8361	0.9116
ALI	0.0573	0.4573	0.0327	6.2012×10^4	148.4866	0.2411
ALI_CD	0.0578	0.4571	0.0327	6.2486×10^4	144.7784	0.9112
ALI_GD	0.0568	0.4575	0.0327	6.0871×10^4	146.9235	0.0494
RAP_CD	0.0134	0.4786	0.0312	8.5062×10^4	251.9034	1.1356
RAP_GD	0.0137	0.4784	0.0312	7.8719×10^4	258.9217	0.0689
RAP_CD_CTR	**0.0134**	**0.4786**	**0.0327**	8.5250×10^4	248.8460	1.1285
RAP_GD_CTR	0.0137	0.4785	0.0327	7.9012×10^4	261.8263	0.0709

Table 4: Performance of different methods on the large dataset after 100 iterations (U is underdelivery rate, P is play rate, Avg Time is the average time for each iteration).

Algorithm	U	P	CTR	L_2	Init Time	Avg Time
HWM	0.1847	0.8336	0.0309	0.564×10^3	0.5602	-
SHALE	0.0862	0.9343	0.0308	2.789×10^3	2.3205	0.0504
ALI	0.0902	0.9302	0.0324	2.205×10^3	2.3772	0.0156
ALI_CD	0.0861	0.9345	0.0324	2.495×10^3	2.3343	0.0526
ALI_GD	0.0894	0.9311	0.0324	2.082×10^3	2.3300	0.0034
RAP_CD	0.0453	**0.9762**	0.0309	3.449×10^3	4.1851	0.0695
RAP_GD	0.0484	0.9731	0.0309	3.048×10^4	4.1926	0.0047
RAP_CD_CTR	**0.0453**	0.9761	**0.0324**	3.435×10^4	4.2138	0.0698
RAP_GD_CTR	0.0484	0.9730	0.0324	3.040×10^4	4.3696	0.0026

Table 5: Performance of different methods on the small dataset after 100 iterations (U is underdelivery rate, P is play rate, Avg Time is the average time for each iteration).

47.86% on the large dataset, which is 2.11% higher than that of ALI_GD. The play rate of RAP_GD_CTR on the small dataset achieves 97.61%. The L_2 distance of RAP is slightly larger than SHALE and ALI, which means a little loss of representativeness. Among the variants of RAP, we can see that using a gradient method can reduce the time cost by a order of magnitude with negligible loss of underdelivery rate and play rate. As an example, RAP_GD_CTR costs only 6.3% time of RAP_CD_CTR. For 100 iterations, RAP_GD_CTR costs 7.1s on the large dataset and 0.26s on the small dataset. Results in Table. 4 and Table. 5 also show that adding the secondary CTR maximization objective in Eq.(9) slightly improves CTR by 0.15% on both datasets.

Time cost. For the basic CPU solution (in Section 5.2) we use 56 cores of Intel Xeon CPU E5-2680 v4, it costs 43.68 seconds per iteration. For the GPU-accelerated parallel solution in Section 5.3 we use 4 Tesla P40 GPU, it costs 0.809 second per iteration. **The results show that our design can save 98.1% time cost.**

7 CONCLUSION

We present a holistic design of a large-scale guaranteed delivery advertising planning system, which supports efficient request-level impression forecasting and allocation. Our system empowers advertisers to achieve sufficiently precise targeting while keeping high delivery rate and play rate, which can significantly increase the publisher revenue and return on investment of advertisers. It can also provide more user-friendly ad serving with the ability to handle a set of customized linear serving constraints.

ACKNOWLEDGMENTS

This research is supported by the National Key R&D Program of China 2017YFB1003003, NSF China under Grants No. 61822209, 61932016, 61751211, China National Funds for Distinguished Young Scientists with No.61625205, the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] Mohammad Taha Bahadori, Qi Rose Yu, and Yan Liu. 2014. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *NIPS*. 3491–3499.
[2] Vijay Bharadwaj, Peiji Chen, Wenjing Ma, Chandrashekhar Nagarajan, John Tomlin, Sergei Vassilvitskii, Erik Vee, and Jian Yang. 2012. Shale: an efficient algorithm for allocation of guaranteed display advertising. In *ACM SIGKDD*. 1195–1203.
[3] Preeti Bhargava, Thomas Phan, Jiayu Zhou, and Juhan Lee. 2015. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *WWW*. ACM.
[4] Peiji Chen, Wenjing Ma, Srinath Mandalapu, Chandrashekhar Nagarjan, Jayavel Shanmugasundaram, Sergei Vassilvitskii, Erik Vee, Manfai Yu, and Jason Zien. 2012. Ad serving using a compact allocation plan. In *13th ACM Conference on Electronic Commerce*. 319–336.
[5] Zhen Fang, Yang Li, Chuanren Liu, Wenxiang Zhu, Yu Zheng, and Wenjun Zhou. 2019. Large-Scale Personalized Delivery for Guaranteed Display Advertising with Real-Time Pacing. In *ICDM*. IEEE, 190–199.
[6] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. In *50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 117–126.
[7] Ali Hojjat, John Turner, Suleyman Cetintas, and Jian Yang. 2014. Delivering guaranteed display ads under reach and frequency requirements. In *AAAI*.
[8] Ali Hojjat, John Turner, Suleyman Cetintas, and Jian Yang. 2017. A unified framework for the scheduling of guaranteed targeted display advertising under reach and frequency requirements. *Operations Research* 65, 2 (2017), 289–313.
[9] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. 2011. Online bipartite matching with unknown distributions. In *43rd annual ACM symposium on Theory of computing*. 587–596.
[10] Xiaoyang Ma, Lan Zhang, Lan Xu, Zhicheng Liu, Ge Chen, Zhili Xiao, Yang Wang, and Zhengtao Wu. 2019. Large-scale User Visits Understanding and Forecasting with Deep Spatial-Temporal Tensor Factorization Framework. In *ACM SIGKDD*. 2403–2411.
[11] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. 2012. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *23rd annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1690–1701.
[12] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. 2010. Optimal online assignment with forecasts. In *11th ACM conference on Electronic commerce*. 109–118.
[13] Ulrike Von Luxburg et al. 2010. Clustering stability: an overview. *Foundations and Trends® in Machine Learning* 2, 3 (2010), 235–274.
[14] Jia Zhang, Zheng Wang, Qian Li, Jialin Zhang, Yanyan Lan, Qiang Li, and Xiaoming Sun. 2017. Efficient delivery policy to minimize user traffic consumption in guaranteed advertising. In *AAAI*.