Inventory Based Recommendation Algorithms

1st Du Chen
Antai College of Economics and Management
Shanghai Jiao Tong University
Shanghai, China
chendu2017@sjtu.edu.cn

2nd Yuming Deng
Alibaba Group
Hangzhou, China
yuming.dym@alibaba-inc.com

3rd Guangrui Ma
Alibaba Group
Hangzhou, China
guangrui.mgr@alibaba-inc.com

4th Hao Ge
Alibaba Group
Hangzhou, China
gehao.haoge@alibaba-inc.com

5th Yunwei Qi
Alibaba Group
Hangzhou, China
yunwei.qyw@alibaba-inc.com

6th Ying Rong
Antai College of Economics and Management
Shanghai Jiao Tong University
Shanghai, China
yrong@sjtu.edu.cn

7th Xun Zhang
Antai College of Economics and Management
Shanghai Jiao Tong University
Shanghai, China
xunzhang@sjtu.edu.cn

8th Huan Zheng
corresponding author
Antai College of Economics and Management
Shanghai Jiao Tong University
Shanghai, China
zhenghuan@sjtu.edu.cn

***Abstract*—We propose two recommendation algorithms for e-commerce with supply limits, a scenario that has not been intensively studied in the literature. One algorithm is a linear programming-based algorithm that uses historical data to approximate customer arrival patterns and generate shadow prices for inventories. The price of inventory can be introduced to existing recommendation algorithms to obtain adjusted rankings for recommendation. The other algorithm balances expected revenue and inventory consumption, and it uses a simple penalty function to reduce the chance of recommending low-inventory-level products. Both algorithms are suitable for online recommendation systems for grocery stores with both online and offline channels, and can incorporate the features of perishable products, which need to be sold within limited time. Both algorithms are tested in a simulation using estimated parameters from Freshippo, a supermarket owned by the Alibaba Group. The numerical results show that both algorithms can generate higher sales volume and higher revenue.**

***Index Terms*—E-commerce, Top-K recommendation, Robust Ranking, Linear Programming**

I. Introduction

In e-commerce systems, the prevailing recommendation algorithms estimate the click through rates (ctr) and conversion rates (cvr) of products for each arrived customer, and rank products in terms of a weighted function of ctr, cvr and product price. The top-K products will be recommended, where K is the number of products to be recommended. These algorithms ignore the supply limit. [1] gives an example showing that under the setting of limited inventories and high customer heterogeneity, a system only recommending the products with highest expected revenue obtained from arrived customers will lose close to 50% of revenue compared with the optimal solution.

A natural improvement is to incorporate inventory limits in the ranking. It is intuitive that a product with high demand and low inventory needs to be adjusted to a lower ranking value. One approach is to use a linear programming(LP) model with inventory constraints to generate the shadow prices of inventories. An adjusted price, i.e., the true price minus the inventory shadow price, can be used to adjust the rankings (e.g., [2], [3], [4]). The LP approach requires satisfactory demand prediction and needs to be resolved when the arrival pattern changes, which may incur computational burdens and delay the response time of online recommendations. Another approach is to generate an inventory-balancing weight [1] for each product, which can be a ratio of the current inventory level to the initial inventory level. The product of the ratio and the original ranking value can be used for ranking. This algorithm is easily implementable and requires little computation, but might be conservative because it does not directly incorporate arrival information.

We evaluate inventory-based recommendation algorithms for grocery stores that have both online and offline channels. Online orders are fulfilled by stocks in the store. These stores offer many perishable products and experience time variant customer arrivals in terms of both customer types and arrival rates. We propose an LP-based algorithm and an inventory-balancing algorithm incorporating the features of grocery stores. The algorithms are tested via simulations based on real data from Freshippo, a supermarket owned by Alibaba Group, including customer arrival, product price, product type, ctr and cvr estimations, inventory levels, etc. Our experimental results suggests that both algorithms can

The authors thank the support from National Natural Science Foundation of China (No. 71871139 and No. 71632007), and the support from Alibaba Innovative Research Program,

achieve higher revenue compared to the prevailing greedy algorithms. Our experimental results also suggest that the LP-based algorithm is sensitive to input data; thus, selecting suitable input data is crucial.

II. Literature Review

Deep learning methods boost the performance of recommendation systems due to their ability to capture complex nonlinear relationships between items and users. Various neural network methods (e.g., [5] [6], [7], [8], etc) have been implemented in many companies. A direct application of the learning methods is how to estimate ctr and cvr, which is definitely an important research problem. For a comprehensive and in-depth survey of this field, see [9] and [10]. Our study, however, focuses on recommendation algorithm development after ctr/cvr estimation, which is a common step of a recommendation system in practice (see [21], [22]).

Most studies in the literature recommend the top-K products to customers without considering inventory limits. One exception is the online resource allocation problem. Various online algorithms have been studied in this setting, including the Adwords problem, online assignment problems and revenue management problems. Our study is closely related to the Adwords problem and online revenue management problems. In both problems, customers see ads from advertisers (or for products) if the advertisers' budget is running out (or the product is out of stock). [11] derive a balancing algorithm from LP in a competitive adversarial model, which shows that the worst-case competitive ratio for Adwords is $1-\frac{1}{e}$. Similar problems are studied by [12] and [13]. [14] propose a training-Based primal-dual algorithm to solve an ad allocation problem, and numerical results confirm that their algorithm performs well on a real data set. [15] consider an online assignment with forecasting, in which the dual space of the optimization problem is used to create an allocation algorithm. [16] present an algorithm that achieves a $1-O(\sqrt{\gamma nlog(mn/\epsilon)})$ competitive ratio. [17] analyze the performance of a greedy algorithm in a randomized input model with queries arriving in a random permutation. [18] can achieve a competitive ratio of $1-O(\sqrt{\gamma nlog(n/\epsilon)})$ if the bid for each assignment follows an unknown i.i.d distribution. [3] study page-based online ad allocation considering constraints with multiple ads per page. The key ingredient of their result is a novel primal-dual analysis, and experiments on real-world data sets show significant improvements.

Online recommendation problems also attract attention from the operations research/management science community. Under a random permutation setting, [19] propose a learning-based algorithm based on LP to obtain a better competitive ratio. [1] introduce an inventory balance algorithm in assortment problems. [20] extend the work of [1] to the settings in which rewards depend on the customer type and not just on the products sold.

Unlike the above studies, our research incorporates the salvage value of unsold products in our algorithm while those works essentially set the salvage value of all products as zero. Moreover, customers' arrivals are typically nonstationary, and we customize our algorithms to handle such time variant customer arrival patterns.

III. Problem Formulation and Algorithms

We consider the following problem. An online retailer sells n products with price $r_1, r_2 \ldots r_n$ for T periods, e.g. a whole day. Without loss of generality, we assume that one customer arrives at each period. Upon arrival at period t, the system will estimate cvr_{it} and ctr_{it} for product i ($i = 1, 2, \ldots n$). The recommendation system then recommends K products to the customer. At the beginning of period 1, the store holds C_i units of product i, which cannot be replenished during the T-period horizon. For simplicity, we assume the goal is to maximize the total expected revenue from the recommendation. The problem is hard because inventory links the decisions across periods, and the traditional myopic recommendation algorithm will not work well. We next describe an LP-based algorithm (LP) and an inventory-balancing algorithm (IB).

A. LP-based Algorithm

We use an LP model to generate shadow prices for inventories. Shadow price is a widely adopted concept in Linear Programming. Each constraint is associated with a shadow price that measures the scarcity of the resource in that constraint. A higher scarcity level will result in a higher shadow price of the associated resource. In many cases, shadow price is the optimal value of the Lagrangian multiplier of the associated constraint. As customers' arrival is generally not stationary, we need to divide the whole horizon into S segments. In each segment, we assume that customers' arrival is stationary. Before the start of segment s, we use the following LP model to generate the shadow prices of inventories to be used in that segment.

$$\begin{aligned}
\text{[P1]}\quad \max \quad & \sum_{i=1}^{n}\sum_{t=1}^{\hat{T}_s} \hat{cvr}^s_{it}\hat{ctr}^s_{it} r_i x_{it} + \sum_{i=1}^{n} w_i I_i \\
\text{s.t.} \quad & \sum_{t=1}^{\hat{T}_s} \hat{cvr}^s_{it}\hat{ctr}^s_{it} x_{it} + I_i = \hat{b}^s_i,\ \forall i = 1,2\ldots n \quad (1)\\
& \sum_{i=1}^{n} x_{it} \le K,\ \forall t = 1,2,\ldots \hat{T}_s \\
& I_i \ge 0,\ \forall i = 1,2,\ldots,n \\
& 0 \le x_{it} \le 1, \forall i,t
\end{aligned}$$

We randomly draw $\hat{T}_s$ of historical customers' click through rate ($\hat{ctr}^s_{it}$) and conversion rate ($\hat{ctr}^s_{it}$), which represents the customers' arrival pattern segment s. $\hat{b}^s_i$ stands for the inventory scaled for the sample. w_i stands for the salvage value of product i, w_i is usually less than half of r_i for a perishable product, and w_i is very close to r_i for a nonperishable product. x_{it} is a decision variable that equals 1 if product i is recommended to customer t, or zero otherwise. I_i is the leftover inventory. The model maximizes the total expected

Algorithm 1 LP-based Algorithm

```
1: for s=1,2,...,S do
2:     Prepare parameters ($\hat{cvr}^s_{it}\hat{ctr}^s_{it}, \hat{b}^s_i$)        ▷ Segment-specified parameters
3:     Solve [P1] with above parameters
4:     Get the shadow price $\alpha^s_i$ of constraint (1)
5:     for Customer $t$ in Period $T_s$ do
6:         Remove products with zero inventory value
7:         Get the estimation of $cvr^s_{it}ctr^s_{it}$
8:         $score_{it} \leftarrow (r_i - \alpha^s_i)cvr^s_{it}ctr^s_{it}$
9:         Sort $\{score_{it}\}^n_{i=1}$ in descending order, obtain sorted rank $[i]$
10:        $S^t = \{i : [i] \leq K\}$
11:    end for
12: end for
```

Algorithm 2 Inventory Balancing Algorithm

```
1: for period t=1,2,...,T do
2:     $score_{it} \leftarrow \Psi(I^{t-1}_i/C_i)(r_i - w_i)cvr_{it}ctr_{it}$
3:     Sort $\{score_{it}\}^n_{i=1}$ in descending order, obtain sorted rank $[i]$
4:     $x_{it} = 1$ if $[i] \leq K$, $x_{it} = 0$ otherwise
5: end for
```

revenue from the recommendation plus the salvage value of the leftover inventory. Please note that we can simply set $w_i = 0$ for all products if the salvage value does not need to be considered. The first constraint is the inventory-balancing constraint in which the expected number sold plus the leftover equals the inventory at the beginning of segment s. The second constraint is the cardinality constraint requiring that no more than K products are recommended to each customer. The last constraint relaxes the integer requirement of x_{it}. We can solve the above LP model and to obtain the dual inventory price for each product as the shadow price of (1).

It is clear that for each segment, the parameters $(\hat{cvr}^s_{it}\hat{ctr}^s_{it}, \hat{b}^s_i)$ approximate the features of customers arriving in segment s. Under the assumption that customers in different segments are independently identically distributed, $\hat{cvr}^s_{it}\hat{ctr}^s_{it}$ can be drawn from customers who arrived in the last segment. If we believe the distribution of customers is not iid across the segments in the same day, but is identical in the same segment in previous days, we could draw $\hat{cvr}^s_{it}\hat{ctr}^s_{it}$ from customers arriving in the same segment in former days. Similarly, under the iid assumption, we can set $\hat{b}^s_i = \frac{c^s_i}{(S-s+1)}$, where c^s_i is the inventory of product i at the beginning of segment s. Otherwise, $\hat{b}^s_i = \frac{c^s_i \hat{N}^s_i}{(\hat{N}^s_i + \dots + \hat{N}^S_i)}$, where $\hat{N}^s_i$ is the predicted demand for product i in segment s. The steps of the LP-based algorithm are described in Algorithm 1.

B. Inventory-Balancing Algorithm

Inventory-balancing [1] tries to balance the expected adjusted revenue obtained with the inventory consumption in each period. An increasing penalty function $\Psi : [0, 1] \rightarrow [0, 1]$ with $\Psi(0) = 0, \Psi(1) = 1$ is defined to penalize lower inventory levels. At period t, it solves the following optimization problem:

$$
\begin{aligned}
\text{[P2]}\ \max \quad & \sum_{i=1}^{n} \Psi(I^{t-1}_i/C_i)(r_i - w_i)ctr_{it}cvr_{it}x_{it} \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{it} \leq K \\
& x_{it} \in \{0, 1\}
\end{aligned}
$$

where I^{t-1}_i denotes the remaining inventory of product i at the end of period $t-1$, C_i represents the initial inventory level of product i. The value of penalty function $\Psi(I^{t-1}_i/C_i)$ decreases with the ratio of the current inventory to the initial inventory, which means a product with a low inventory ratio is less likely to be offered.

[P2] can be easily solved by sorting products with $\Psi(I^{t-1}_i/C_i)(r_i - w_i)ctr_{it}cvr_{it}$ in a decreasing order. The top K products are then recommended. Theoretically, if the purchase probability is estimated based on a multinomial logit model (see [23]), the IB method can obtain a competitive ratio of no less than $1 - \frac{1}{e}$ if $\Psi(x) = \frac{e}{e-1}(1 - e^{-x})$ and $w_i = 0$ (see [1]). The steps of the IB algorithm are described in Algorithm 2.

Algorithm 1 and Algorithm 2 are different from two perspectives. First, Algorithm 1 needs to solve an LP model for each segment. Shadow prices will be updated periodically to incorporate inventory information. Shadow prices remain unchanged within the current segment. To some extent, the accuracy of shadow prices highly depends on the updating frequency. Algorithm 2 updates inventory levels at each period, and has little computational burden. Second, the LP model used in Algorithm 1 requires certain prediction precision and assumes that arrivals are independent to obtain accurate shadow price estimation. In other words, if the prediction is more accurate, Algorithm 1 performs better. Algorithm 2 is

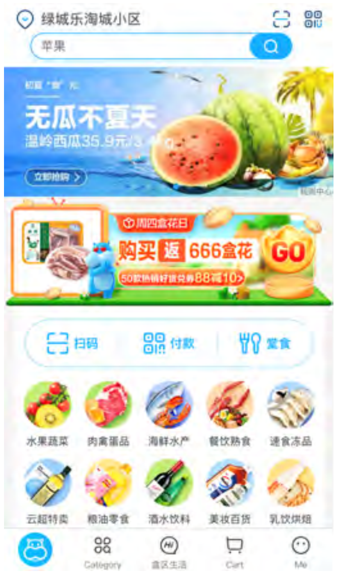

Fig. 1. Freshippo Home Page

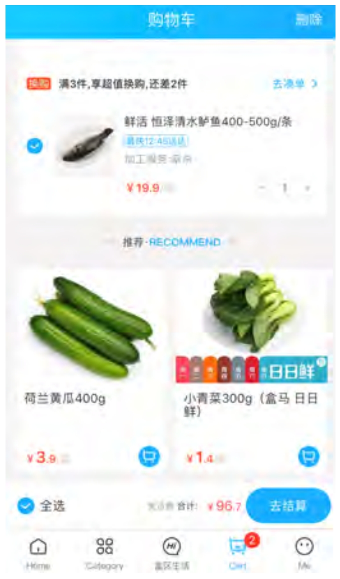

Fig. 2. Freshippo Cart Page

robust and does not need prediction and the independent arrival assumption, but might obtain a conservative total revenue.

IV. Numerical Study

In this section, we compare the algorithms based on Freshippo's real dataset. Freshippo, a business-to-customer supermarket owned by the Alibaba Group, offers an efficient and flexible shopping experience by seamlessly blending the online and offline shopping modes. It provides a vast selection of products in its brick-and-mortar stores, including perishable products such as live seafood, milk and vegetables, and nonperishable products such as bottled drinks, sauces. A large portion of the revenue is from its online channel, i.e., the Freshippo APP as shown in Figure 1, in which an order placed will be delivered to the customer's address in approximately half an hour. We test our algorithms in the recommendation slots of Freshippo shopping-cart page, as shown in Figure 2. In Figure 2, two vegetables are recommended. More products will be recommended if a customer slides his or her fingers across the screen.

A. Dataset

We test our algorithm on a shop-day level dataset. Specifically, we use two-day page view data from 5:55 AM to 22:55 PM in a store, which includes approximately 27 millions records for each day. Each record contains SKUID (stock keeping unit identity), predicted *CTR* (click-through rate), predicted *CVR* (conversion rate), *pvtimestamp* (page view timestamp) and *pvid* (page view identity). Customer arrival is drawn from the empirical data proportionally.

We obtain data including product prices, product category (nonperishable or perishable), salvage value and initial inventory level at the beginning of the day. The descriptive statistics of price, salvage value (i.e., the salavage value if a product cannot be sold within that day; obviously, a perishable product's salvage value is low), and the initial inventory are shown in Table I. It is obvious that perishable products have a slightly lower price and higher initial inventory level, and the salvage value is zero. Please note that, for the sake of confidentiality, the numbers are deliberately tuned.

B. Experiment Setting

Four algorithms are tested in our simulation: the LP-based algorithm (Algorithm 1), the inventory-balancing algorithm (Algorithm 2), a greedy algorithm (**GREEDY**), in which products are ranked in a decreasing order of the expected adjusted revenue $ctr_{it}cvr_{it}r_i$ and the top K products are recommended, and a business-rule-based heuristic algorithm run by Alibaba (**REAL**). In the **REAL** algorithm, products are decreasingly ranked by a score considering both purchase and traffic objectives, and the top K products are recommended. We follow the company's business rule and set $K = 20$. Our LP model is solved with OR-tools, an open-source optimization solver.

We test the algorithms in different settings mainly from two perspectives: The first perspective depends on whether salvage cost is considered. We set w_i according to the value of Table I if salvage is considered; and $w_i = 0$ for all products if salvage is not considered. Intuitively, perishable products are more likely to be recommended if salvage cost is considered, which we believe is suitable for grocery stores and ignored by most existing algorithms. Second, we test how the input affects the performance of the LP-based algorithm. We used two types of inputs: historical data from the same segment in the previous day (-1day), the other is the historical data from the previous segment (-1seg).

C. Results

We compare the algorithms based upon both revenue-related metrics and inventory-related metrics as follows:

TABLE I
DESCRIPTIVE STATISTICS ABOUT SKUS

		Mean	Std	Min	25%	50%	75%	Max
Non-perishable Product	Price	2.54	2.58	0.48	1.75	2.11	2.57	4.79
	Salvage	2.49	2.53	0.43	1.70	2.06	2.52	4.74
	Initial Inventory	3.55	3.99	1.00	2.40	3.00	3.64	6.79
Perishable Product	Price	2.46	2.80	0.01	1.77	2.11	2.55	5.80
	Salvage	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Initial Inventory	4.76	5.52	1.00	2.89	3.69	4.61	6.91

TABLE II
NUMERICAL STUDY RESULTS

	Without Salvage					With Salvage		
	GREEDY	LP-based		IB	REAL	LP-based	IB	REAL
Revenue-Related		-1day	-1seg			-1seg		
Sales Volume	-	-2.84%	+2.33%	+4.29%	+4.56%	-1.00%	-2.71%	+4.56%
Revenue	-	+0.12%	+0.70%	+1.97%	-11.11%	-4.50%	-3.98%	-11.11%
Peri.Ratio	-	-14.18%	-7.22%	-3.78%	-0.64%	+13.95%	+18.54%	-0.64%
Revenue+Salvage	-	-0.84%	-0.06%	+0.16%	-1.53%	+0.85%	+1.24%	-1.53%
RevenuePerPv	-	+0.11%	+0.67%	+2.01%	-11.06%	-4.46%	-4.02%	-11.06%
RevenuePerOrder	-	+2.89%	-1.31%	-2.54%	-14.11%	-3.85%	-1.40%	-14.11%
Inventory-Related								
Sold-out Rate	-	-3.06%	+0.33%	+3.29%	+2.02%	+0.22%	-5.32%	+2.02%
Leftover Rate	-	+0.00%	-2.62%	-6.63%	-2.61%	-3.11%	+0.16%	-2.61%

Revenue

- Sales Volume: the total number of sold products.
- Total Revenue: the aggregated prices of sold products.
- Peri.Ratio: the ratio of the number of sold perishable products to that of total sold products.
- Revenue+Salvage, the sum of the revenue and the salvage value of unsold products.
- Revenue Per PV, the expected revenue obtained for each page view.
- Revenue Per Order, the expected revenue obtained from each order.

Inventory

- Sold-out Rate: the ratio of the number of sold-out SKUs to total SKUs.
- Leftover Rate: the ratio of the quantity of inventory left to initial inventory.

We summarize the results in two parts: *without salvage*, i.e., all w_i are assumed to be 0, and *with salvage*, i.e., w_i is the actual value. The results are displayed in Table II, in which only the increasing or decreasing percentages compared to **GREEDY** without salvage are shown.

When the salvage value is not considered, the **IB** algorithm obtains the highest sales volume, the highest sold-out rate, and the lowest leftover rate. For the LP-based algorithm, *-1seg* performs significantly better than *-1day* in our instance, which implies that customers' arrival at may differ for the same time of day, and utilizing previous-segment customers' information is preferred.

When the salvage value is considered, as shown in the right part of Table II, **IB** obtains the highest revenue, obtains the highest revenue plus leftover salvage value, and sells more perishable products. It is easy to see that *Peri.Ratio* of **LP-based** (**IB**) increases from -7.22% (-3.78%) to 13.95% (18.54%) when salvage value is considered, which implies that considering salvage value can boost perishable product sales.

The **REAL** algorithm generates the highest sales volume possibly because it focuses on sales volume and click through maximization. If the salvage value is not considered, **IB** can achieve high sales volume close to that of **REAL**, but much higher revenue. The result suggests that **REAL** tends to recommend low price items, and our inventory-based algorithms tend to recommend products with higher prices as they try to balance the sales volume and revenue.

The simulations are run on a PC with an Intel 7 processor and 16GB memory. As the customers' arrival intensity can be large during peak hours, our LP model involves approximately 100 thousand decision variables and approximately 5 thousand constraints for some segments; it takes approximately 70 seconds to obtain a solution and may not be suitable for a real-time recommendation system. The computational time of *IB* is negligible because it only involves multiplication and addition operations.

V. CONCLUSION

In this paper, we evaluated several recommendation algorithms on Freshippo where the daily inventory capacity is usually finite and limited. Simulation experiments on real data reveal that both inventory balancing algorithm and the LP-based algorithm can achieve higher revenue compared to prevailing greedy algorithms. The LP-based algorithm is sensitive to customer heterogeneity and nonstationary arrival patterns. We believe that properly selecting input data is essential, and endowing the LP-based algorithm with dynamics by incorporating the inventory-balancing penalty coefficient could further improve revenue performance. We leave that exploration for future large-scale online experiments that may evaluate both the short-term and long-term impacts of inventory-based algorithms. Moreover, revenue is only one dimension of a recommendation algorithm. A future study may include more dimensions such as product quality, engagement, etc.

References

[1] Golrezaei, Negin, Hamid Nazerzadeh, and Paat Rusmevichientong. ”Real-time optimization of personalized assortments.” Management Science 60.6 (2014): 1532-1551.

[2] Chen, Ye, et al. ”Real-time bidding algorithms for performance-based display ad allocation.” Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 2011.

[3] Devanur, Nikhil R., et al. ”Whole-page optimization and submodular welfare maximization with online bidders.” ACM Transactions on Economics and Computation (TEAC) 4.3 (2016): 1-20.

[4] Dvořák, Wolfgang, and Monika Henzinger. ”Online ad assignment with an ad exchange.” International Workshop on Approximation and Online Algorithms. Springer, Cham, 2014.

[5] Covington, Paul, Jay Adams, and Emre Sargin. ”Deep neural networks for youtube recommendations.” Proceedings of the 10th ACM conference on recommender systems. 2016.

[6] Cheng, Heng-Tze, et al. ”Wide & deep learning for recommender systems.” Proceedings of the 1st workshop on deep learning for recommender systems. 2016.

[7] Okura, Shumpei, et al. ”Embedding-based news recommendation for millions of users.” Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017.

[8] Zhou, Guorui, et al. ”Deep interest network for click-through rate prediction.” Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018.

[9] Zhang, Shuai, et al. ”Deep learning based recommender system: A survey and new perspectives.” ACM Computing Surveys (CSUR) 52.1 (2019): 1-38.

[10] Batmaz, Zeynep, et al. ”A review on deep learning for recommender systems: challenges and remedies.” Artificial Intelligence Review 52.1 (2019): 1-37.

[11] Mehta, Aranyak, et al. ”Adwords and generalized online matching.” Journal of the ACM (JACM) 54.5 (2007): 22-es.

[12] Feldman, Jon, et al. ”Online ad assignment with free disposal.” International workshop on internet and network economics. Springer, Berlin, Heidelberg, 2009.

[13] Buchbinder, Niv, Kamal Jain, and Joseph Seffi Naor. ”Online primal-dual algorithms for maximizing ad-auctions revenue.” European Symposium on Algorithms. Springer, Berlin, Heidelberg, 2007.

[14] Feldman, Jon, et al. Online stochastic ad allocation: Efficiency and fairness. No. arXiv: 1001.5076. 2010.

[15] Vee, Erik, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. ”Optimal online assignment with forecasts.” Proceedings of the 11th ACM conference on Electronic commerce. 2010.

[16] Buchbinder, Niv, et al. ”Online submodular maximization: Beating 1/2 made simple.” Mathematical Programming (2020): 1-21.

[17] Gairing, Martin, and Max Klimm. ”Greedy metric minimum online matchings with random arrivals.” Operations Research Letters 47.2 (2019): 88-91.

[18] Devanur, Nikhil R., et al. ”Near optimal online algorithms and fast approximation algorithms for resource allocation problems.” Proceedings of the 12th ACM conference on Electronic commerce. 2011.

[19] Agrawal, Shipra, Zizhuo Wang, and Yinyu Ye. ”A dynamic near-optimal algorithm for online linear programming.” Operations Research 62.4 (2014): 876-890.

[20] Gallego, Guillermo, et al. Online personalized resource allocation with customer choice. Working Paper. http://arxiv. org/abs/1511.01837 v1, 2016.

[21] Vee E, Vassilvitskii S, Shanmugasundaram J. Optimal online assignment with forecasts[C]//Proceedings of the 11th ACM conference on Electronic commerce. 2010: 109-118.

[22] Bharadwaj V, Chen P, Ma W, et al. Shale: an efficient algorithm for allocation of guaranteed display advertising[C]//Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. 2012: 1195-1203.

[23] McFadden D. The measurement of urban travel demand[J]. Journal of public economics, 1974, 3(4): 303-328.