Inventory Based Recommendation Algorithms

We propose two types of recommendation algorithms for e-commerce systems with supply limits, which has not been intensively studied in literature. One algorithm is a LP-based algorithm, which uses historical data to approximate customers arrival patterns and generate shadow prices for inventories. The price of inventory can be introduced to traditional recommendation algorithms to get adjusted rankings for recommendation. The other algorithm is to balance expected revenue and inventory consumption in each period. It uses a easy-to-calculate penalty function to reduce the chance of recommending low-inventory-level products. Both algorithms are suitable to online recommendation for grocery stores with both online and offline channels, and can incorporate the features of perishable products, which need be sold within limit time. Both algorithms are tested in a simulation using real parameters from Freshippo, a supermarket owned by Alibaba Group. The numerical results show that both algorithms can generate higher sales volume and higher revenue.

CCS Concepts: • **Applied computing** → **Operations research**; **Decision analysis**; • **Information systems** → **Recommender systems**.

Additional Key Words and Phrases: E-commerce, Top-K recommendation, Robust Ranking, Linear Programming

ACM Reference Format:
. 2020. Inventory Based Recommendation Algorithms. 1, 1 (June 2020), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In e-commerce systems, the prevailing recommendation algorithms estimate 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. Top-K products will be recommended. These algorithms may have potential problems because they ignore supply limit. Golrezaei et al. [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% 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 LP model with inventory constraints to generate dual prices of inventories. An adjusted price, i.e., true price minus the inventory dual price, can be used to adjust the rankings (eg, [2],[3], [4]). The LP approach requires satisfactory demand prediction and need to be resolved when arrival pattern changes, which may incur computational burdens and deter response time of online recommendation. Another approach is to generate an inventory balancing weight [1] for each product, which can be a ratio of the current inventory level to initial inventory level. The product of the ratio and the original ranking value can be used for ranking. This algorithm is

Author's address:

XXXX-XXXX/2020/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

easily implementable and requires little computation, but might be conservative because it does not incorporate arrival information directly.

We evaluate both inventory-based recommendation algorithms for grocery stores which have both online and offline channels. Online orders are fulfilled by stocks in the store. These stores offer many perishable products and face time variant customer arrivals in terms of both customer types and arrival rates. We propose a 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 experiment result suggests that both algorithm can achieve higher revenue comparing to prevailing greedy algorithms. Our experiment results also suggest that the LP-based algorithm is sensitive to input data, thus selecting suitable input data is crucial.

2 LITERATURE REVIEW

Deep learning methods boost the performance of recommendation systems due to the ability to capture complex nonlinear relationships between item and user. Various neural network methods (e.g. [5] [6], [7], [5], [8], etc) has been implemented in many companies. For a comprehensive and in-depth survey of this field, see [9] and [10].

Most studies in literature recommend Top-K products to customer 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 Adwords problem and online revenue management problems. In both problems, customers will not get ads from advertisers (or products) if the advertisers budget is running out (or product is out of stock). Mehta et al. [11] derive a balancing algorithm from linear programming (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], [13]. Feldman et al. [14] propose a Training-Based Primal-Dual algorithm to solve an ad allocation problem. Numerical results confirms that their algorithm performs well on real data set. Vee et al. [15] consider an online assignment with forecasting, in which the dual space of the optimization problem is used to create an allocation algorithm. Buchbinder et al. [16] present an algorithm, which achieves $1 - O(\sqrt{\gamma n log(mn/\epsilon)})$ competitive ratio. Gairing and Klimm [17] analyze the performance of a greedy algorithm in a randomized input model with queries arriving in a random permutation. Devanur et al. [18] can achieve a competitive ratio of $1 - O(\sqrt{\gamma n log(n/\epsilon)})$ if the bid for each assignment follows an unknown i.i.d distribution. Devanur et al. [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 set show significant improvements.

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

Different from 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 non-stationary, we also customize our algorithms to handle such time variant customer arrival patterns.

3 PROBLEM FORMULATION AND ALGORITHMS

We consider the following problem. An online retailer sells n products with price $r_1, r_2 \dots r_n$ 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 ($i = 1, 2, \dots n$). 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 total expected revenue from 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 a LP-based algorithm (LP) and an inventory balancing algorithm (IB).

3.1 LP-based Algorithm

We use a LP model to generate dual prices for inventories. 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 dual prices of inventories to be used in that segment.

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

We randomly draw $\hat{T}_s$ of historical customers' click through rate ($\hat{ctr}_{it}^{s}$) and conversion rate ($\hat{cvr}_{it}^{s}$), which represents the customers' arrival pattern segment s. $\hat{b}_i^s$ stands for the inventory scaled for the 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 non-perishable product. x_{it} is a decision variable which equals to 1 if product i is recommended to customer t, or zero otherwise. I_i is the leftover inventory. The model maximizes the total expected revenue from recommendation plus the salvage value of leftover inventory. Please note that we can simply set $w_i = 0$ for all products, if salvage value need not to be considered. The first constraint is the inventory balancing constraint in which the expected sold number plus the leftover equals to 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 one relaxes the integral requirement of x_{it}. It is easy to see that x_{it} still get binary solutions in the relaxed LP model. By strong duality theory, we can solve the above LP model and get the dual inventory price for each product as the dual price of (1).

It is clear that for each segment, the parameters ($\hat{cvr}_{it}^{s} \hat{ctr}_{it}^{s}, \hat{b}_i^s$) approximate the features of customers arriving in segment s. Under the assumption that customers in different segments are independently identically distributed, $\hat{cvr}_{it}^{s} \hat{ctr}_{it}^{s}$ 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 $\hat{cvr}_{it}^{s} \hat{ctr}_{it}^{s}$ from

customers arrived in the same segment in former days. Similarly, under the iid assumption, we can set $\hat{b}_i^s = \frac{c_i^s}{(S-s+1)}$, where c_i^s is the inventory of product i at the beginning of segment s. Otherwise, $\hat{b}_i^s = \frac{c_i^s \hat{N}_i^s}{(\hat{N}_i^s + \ldots + \hat{N}_i^S)}$, where $\hat{N}_i^s$ is the predicted demand for product i in segment s. The steps of the LP-based Algorithm is describe in Algorithm 1.

Algorithm 1 LP-based Algorithm

for s=1,2,...,S **do**
  Prepare parameters $(\hat{cvr}_{it}^s \hat{ctr}_{it}^s, \hat{b}_i^s)$ ▷ Segment-specified parameters
  Solve [P1] with above parameters
  Get the shadow price α_i^s of constraint (1)
  for Customer t in Period T_s **do**
    Remove products with zero inventory value
    Get the estimation of $cvr_{it}^s ctr_{it}^s$
    $score_{it} \leftarrow (r_i - \alpha_i^s) cvr_{it}^s ctr_{it}^s$
    Sort $\{score_{it}\}_{i=1}^n$ in descending order, obtain sorted rank $[i]$
    $S^t = \{i : [i] \leq K\}$
  end for
end for

3.2 Inventory Balancing Algorithm

The idea of Inventory balancing [1] tries to balance the expected adjusted revenue obtained and 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:

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

where I_i^{t-1} 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 $\Psi(I_i^{t-1}/C_i)$ decreases with the ratio of the current inventory to the initial inventory, which means a product with low inventory low inventory ratio is less likely to be offered.

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

Algorithm 1 and Algorithm 2 are different in two perspectives. Firstly, Algorithm 1 needs to solve a LP model for each segment. Shadow prices will be updated periodically to incorporate inventory information. Shadow prices stay unchanged within the current segment. To some extent, the accuracy of shadow prices highly depends on updating frequency. Algorithm 2 updates inventory levels at each period, and has little computational burden. Secondly, the LP model used in Algorithm

Algorithm 2 Inventory Balancing Algorithm

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

1 requires certain prediction precision and assumes that arrivals are independent to get accurate shadow price estimation. In other words, if the prediction is more accurate, Algorithm 1 performs better. Algorithm 2 is robust and does not need prediction and the independent arrival assumption, but might get a conservative total revenue.

4 NUMERICAL STUDY

In this section, we compare the algorithms based on Freshippo's real dataset. Freshippo, a Business-to-Customer supermarket owned by Alibaba Group, offers an efficient and flexible shopping experience by seamlessly blending the online and offline shopping mode. It provides vast selection of products in its brick-and-mortar stores, including perishable products such as live seafood, milk and vegetable, and non-perishable products such as bottled drinks, sauces, etc. A large portion of the revenue is from its online channel, i.e. Freshippo APP as shown in Figure 1, in which an order placed will be delivered to the customer's address in around 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 slide fingers across the screen.

Fig. 1. Freshippo Home Page

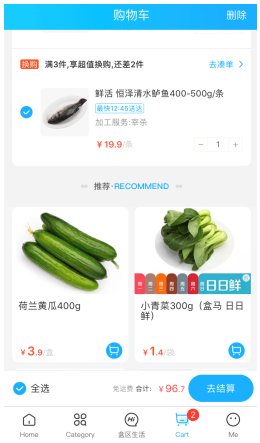

Fig. 2. Freshippo Cart Page

4.1 Dataset

We test our algorithm on a shop-day level dataset. Specifically, we use page view data on 1st May 2020 from 5:55 AM to 22:55 PM in a store, which includes about 27 millions records. 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). About Twenty products are shown on each page view.

Table 1. 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

We get data including products' prices, products' category (non-perishable or perishable), salvage values and initial inventory level at the beginning of the day. The descriptive statistics of price, salvage value (i.e., the residual price if a product cannot be sold within that day. Obviously, a perishable product's salvage value is low.), and initial inventory are shown in Table 1. It is obvious that perishable products have a little bit 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.

4.2 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 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 perspective: Firstly, whether salvage cost is considered. We set w_i according to the value of Table 1 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 to grocery stores and ignored by most existing algorithms. Secondly, we test how the input affect the performance of the LP-based algorithm. We used two types of inputs: one is the historical data from the same segment in the previous day (-1day), the other is the historical data from the previous segment (-1seg).

4.3 Results

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

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 revenue and 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 that of total SKUs.
- Leftover Rate: the ratio of the quantity of left inventory to that of 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. Results are displayed in Table 2, in which only the increasing or decreasing percentages compared to **GREEDY** without salvage are shown.

Table 2. 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%

When salvage value is not considered, **IB** algorithm gets the highest sales volumen, 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 cutomers' arrival at the same time of days may be different, and utilizing previous-segment customers' information is preferred.

When salvage value is considered, as shown in the right part of Table 2, **IB** gets the highest revenue, the highest revenue plus leftover slavage value, and sells more perishable products. It is easy to see that *Peri.Ratio* of **LP-based** (**IB**) increase 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 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 tries to balance the sales volume and revenue.

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

5 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 LP-based algorithm can achieve higher revenue comparing to prevailing greedy algorithms. And LP-based algorithm is sensitive to customer heterogeneity and non-stationary arrival pattern. We believe that properly selecting input data is essential, and endowing the LP-based algorithm with dynamics by incorporating inventory-balancing penalty coefficient could further improves revenue performance. We leave that exploration for future large-scale online experiment.

REFERENCES

[1] Negin Golrezaei, Hamid Nazerzadeh, and Paat Rusmevichientong. Real-time optimization of personalized assortments. *Management Science*, 60(6):1532–1551, 2014.

[2] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1307–1315, 2011.

[3] Nikhil R Devanur, Zhiyi Huang, Nitish Korula, Vahab S Mirrokni, and Qiqi Yan. Whole-page optimization and submodular welfare maximization with online bidders. *ACM Transactions on Economics and Computation (TEAC)*, 4(3): 1–20, 2016.

[4] Wolfgang Dvořák and Monika Henzinger. Online ad assignment with an ad exchange. In *International Workshop on Approximation and Online Algorithms*, pages 156–167. Springer, 2014.

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[7] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1933–1942, 2017.

[8] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068, 2018.

[9] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[10] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37, 2019.

[11] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007.

[12] Jon Feldman, Nitish Korula, Vahab Mirrokni, Shanmugavelayutham Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *International workshop on internet and network economics*, pages 374–385. Springer, 2009.

[13] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer, 2007.

[14] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein. Online stochastic ad allocation: Efficiency and fairness. Technical report, 2010.

[15] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 109–118, 2010.

[16] Niv Buchbinder, Moran Feldman, Yuval Filmus, and Mohit Garg. Online submodular maximization: beating 1/2 made simple. *Mathematical Programming*, pages 1–21, 2020.

[17] Martin Gairing and Max Klimm. Greedy metric minimum online matchings with random arrivals. *Operations Research Letters*, 47(2):88–91, 2019.

[18] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *Journal of the ACM (JACM)*, 66(1):1–41, 2019.

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

[20] Guillermo Gallego, Anran Li, Van-Anh Truong, and Xinshang Wang. Online personalized resource allocation with customer choice. Technical report, Working Paper. http://arxiv. org/abs/1511.01837 v1, 2016.