Inventory Based Recommendation

PUT ABSTRACT HERE.

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. 1, 1 (May 2020), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In e-commerce systems, the prevailing recommendation algorithms estimate click through rate (ctr) and conversion rate (cvr) 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. [1] gives an example showing that under limited inventory and high customer heterogeneity, a system only recommending the most preferred product of an arrived customer 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 the 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 a good demand prediction and need to be resolved as the 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 is a ratio between current inventory level and initial inventory level. The products then can be ranked according to the product of the weight and the original ranking value. This algorithm is easily implementable and require little computation, but might be conservative because it does not incorporate arrival information directly.

We evaluate both inventory-based recommendation algorithms in the setting of grocery stores which also deliver to online orders within 3km radius from the store. Many products are perishable and customer arrival intensity is time variant, which will be incorporated in our algorithm. We run experiments based on real data from fromFreshippo, a supermarket owned by Alibaba Group, including customer arrival, product price, product type, ctr and cvr estimations, and inventory level at the beginning of each day. Our experiment result suggests that.....

Author's address:

2 LITERATURE REVIEW

Recently, deep learning is playing an important role in recommendation systems. Due to deep learning has ability to capture nonlinear user/item relationships and complex data representations in the higher layers, it has changed the structure of recommendation system significantly, and is able to provides higher quality recommendation results. In practice, deep learning recommendation systems has been implemented in many companies. [5] propose a deep neural network-based system to recommend video on YouTube. [6] present the wide & deep structure, which combines linear model and deep neural network to obtain better benefits of memorization and generalization. [7] apply an RNN-based news recommender system for Yahoo! News. [8] introduce the attention mechnism to recommendation system and obtain a better performance in Alibaba Group. For a comprehensive and in-depth survey of this field, see [9] and [10].

Different from recommending Top-K products to customer, we focus on the recommendation with constrained inventory in this paper. This problem can also be regarded as a online resource allocation problem, including Adwards problem, online assignment problem, online revenue management problem and so on. A lot of existing papers are analyzed based on linera program (LP). In the competitive adversarial model, [11] derive a balance algorithm from LP, which shows that the worst-case competitive ratio for Adwords is $1 - \frac{1}{e}$. Similar problems are studied by [12], [13]. Under random-order stochastic model, [14] propose the Training-Based Primal-Dual algorithm to solve Ad Allocation problem, numerical experiments confirms that their algorithm perform well on real data set. [15] consider a online assignment with forcasts, the dual space of the resulting optimization problem is used to create allocation algorithm. [16] presents a algorithm, which achives $1 - O(\sqrt{\gamma nlog(mn/\epsilon)})$ competitive ratio. [17] analyze the performance of the Greedy algorithm in a randomized input model with queries arriving in a random permutation. Suppose the bid for each assignment follows an unknown i.i.d distribution, then $1 - O(\sqrt{\gamma nlog(n/\epsilon)})$ competitive ratio has been shown in [18]. [3] study page-based online ad allocation considering general allocation constraints with multiple ads per each page. The key ingradient of their result is a novel primal-dual analysis, and experiments on real-world datasets show significant improvements.

This problem also attracts attention from operations reserach / management science community. On the basis of LP, [19] propose a learning-based algorithm, which is able to obtain a better competitive ratio. [1] introduce inventory balance algorithm in assortment problem and numerical experiments shows that their algorithm perform well on real data set. [20] extend the work by [1] and assume that rewards depend on the customer type and not just on the products sold. [21] study the recommendation at checkout under inventory constraint, they derive an algorithm with a 1/4 competitive ratio guarantee under adversarial arrivals.

3 PROBLEM FORMULATION AND ALGORITHMS

An online retailer sells n products with price $r_1, r_2 \ldots r_n$ for T periods, e.g. a whole day. Without Lost 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 $i = 1, 2, \ldots n$. The recommendation system then needs to recommend 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 NS 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 using in that segment.

$$\begin{aligned} \text{[P1]} \quad \max \quad & \sum_{i=1}^{n}\sum_{t=1}^{\hat{T}_s} \hat{cor}_{it}^{s}\hat{ctr}_{it}^{s} r_i x_{it} + \sum_{i=1}^{n} w_i I_i \\ \text{s.t.} \quad & \sum_{t=1}^{\hat{T}_s} \hat{cor}_{it}^{s}\hat{ctr}_{it}^{s} x_{it} + I_i = \hat{b}_i^s, \ \forall i = 1, 2 \ldots n, \\ & \sum_{i=1}^{n} x_{it} \leq K, \ \forall t = 1, 2, \ldots \hat{T}_s \\ & 0 \leq x_{it} \leq 1, \forall i, t \end{aligned} \tag{1}$$

where we randomly draw $\hat{T}_s$ of historical customers' click through rate ($\hat{ctr}_{it}^{s}$) and conversion rate ($\hat{ctr}_{it}^{s}$), 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. 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 duality theorems, we can solve the above LP model and get the dual inventory price for each product as the dual price for (1).

It is clear that for each segment, the parameters ($\hat{cor}_{it}^{s}\hat{ctr}_{it}^{s}$, $\hat{b}_i^s$) approximate the preference of customers in segment s. For example, under the assumption that customers in different segments are independently identically distributed, $\hat{cor}_{it}^{s}\hat{ctr}_{it}^{s}$ can be drawn from customers who arrived in the last segment. If we believe the distribution of customers arriving in the same segment is not iid across the segments in the same day, but identical in previous days, we could draw $\hat{cor}_{it}^{s}\hat{ctr}_{it}^{s}$ from customers arrived in the same segment in former days. Similarly, under the iid assumption, $\hat{b}_i^s = \frac{c_i^s}{(NS-s+1)}$, where c_i^s is the inventory of product i at the beginning of segment s. [REVISE IT: GIVE the definition of your c_i^s, -s, N-s, etc. please double check the definition, at:chendu] otherwise, $\hat{b}_i^s = \frac{c_i^{-s}N_i^{-s}}{(N_i^{-s}+....+N_i^{-NS})}$. where c_i^{-s}, N_i^{-s} are the inventory of product i at the begining of segment s in the last selling horizon and the number of customers in the segment s in the last selling horizon.

Algorithm 1 scratches the steps of the LP-based algorithm.

3.2 Inventory Balancing Algorithm

The **Inventory Blancing** [1] takes into account both the revenue that would be obtained from the customer and the current inventory levels to decide which products to be recommended. An increasing penalty function $\Psi : [0, 1] \longrightarrow [0, 1]$ with $\Psi(0) = 0, \Psi(1) = 1$ is defined to penalize lower inventory levels. At period t, it solves the following optimization problem:

Algorithm 1 Dual Approach

```
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
```

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

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. It is easy-to-implement because only the inventory levels are updated at each period. 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]).

Formally, the **Inventory Balancing** algorithm is as following:

Algorithm 2 Inventory Balancing Algorithm

```
1: for period t=1,2,...,T do
2:     $score_{it} \leftarrow \Psi(I_i^{t-1}/c_{i0})(r_i - w_i)cvr_{it}ctr_{it}$
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
```

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, which means $score_{it}$ would not be modified in real time. Therefore, 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 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 the section, we discuss the design of our numerical study, which is based on Freshippo's real dataset. Freshippo, a Business-to-Customer supermarket owned by Alibaba Group, offers consumers a more-efficient and more-flexible shopping experience by seamlessly blending the online and offline shopping mode with the help of smart logistic technology and data. It provides vast selection of products in its brick-and-mortar stores, including perishable products like 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. 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, and as the customer pulls the screen upward, more products will be displayed. [Describe Inventory] and product size

Fig. 1. Freshippo Home Page

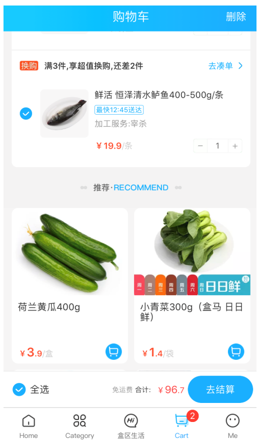

Fig. 2. Freshippo Cart Page

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 single store, which includes about 27 millions records. Each record contains SKUID (stock keeping unit identity), the predicted *CTR* (Click Through Rate, referring to the probability of clicking the displayed product), the predicted *CVR* (Conversion Rate, referring to the probability of purchasing after the customer clicks the corresponding product), *pvtimestamp* (page view timestamp) and *pvid* (page view identity). For each page view, there are hundreds [dozens to hundreds?] of products recalled from thousands of SKUs, where the half of them are perishable products as shown in Table 1, by upstream prediction algorithm, which implies that the amount of unique page view is about at the magnitude of 100 thousands. However, the number of recommended products finally shown in the app is only 20 because of the limited area on a phone.*** give a number or range.

We also get data including products' prices, products' category (perishable or non-perishable), salvage values and initial inventory level at the beginning of the day. Table 1 shows the descriptive

Table 1. Descriptive Statistics about SKUs

		Mean	Std	Min	25%	50%	75%	Max
Non-perishable Product N = 8.58	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 N = 7.45	Price	2.46	2.80	1.00	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

statistics. In Table 1, we summarize price, salvage, and initial inventory according to category, and the numbers of SKUs in each category are $N = 8.58$ and $N = 7.45$. 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, shown numbers are deliberately blurred.

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 revenue $ctr_{it} * cvr_{it} * r_i$ and top K products are recommended, and a business-rule-based heuristic algorithm proposed by Alibaba (**REAL**). In the **REAL** algorithm, products are decreasingly ranked by a scores considering both purchase and traffic objectives, and the top K products are recommended. We follow the company's business rule and set $K = 20$. Additionally, linear programming problem is solved with open-source solver OR-tools.

To explore advantages and disadvantages of several algorithms, sensitivity analysis on whether salvage value is took into account is conducted to explore its impact. For **Dual Approach**, we also feed the linear programming model with customers in the previous segment or in the previous day to explore the influence of input customers. We represent them as *-1day* and *-1seg*, respectively.

4.3 Results

We compared results over both front-end revenue-related indices and back-end inventory-related measurements.

Revenue

We focus on the following measurements:

- Sales, the total number of sold products.
- Revenue, the aggregated price of sold products.
- Revenue to Initial Value, the ratio of aggregated value of sold products to the value of all initial products.
- Revenue+Salvage, the revenue plus salvage value of all remaining products.
- Revenue Per Pv, revenue obtained from each page view.
- Revenue Per Order, revenue obtained from each order.

Table 2 shows the revenue-related results among algorithms. We only displayed the relative increasing or decreasing compared to **GREEDY**, rather than showing the exact sales or revenue for the sake of confidentiality.

When the salvage value is not took into consideration, it is obvious that, for LP-based algorithm, since daily customer homogeneity and arrival stationarity may not exist, the improvement in terms of sales and revenue is very limited. For **IB** algorithm, the sales is boosted a lot, as well as the obtained total revenue. And for **REAL**, since the business-rule-based heuristic algorithm focuses

Table 2. Revenue-related Results

	Does not Consider Salvage				Consider Salvage		
	LP-based		IB	REAL	LP-based	IB	REAL
	-1day	-1seg			-1seg		
Sales	-2.84%	+2.33%	+4.29%	+4.56%	+4.16%	+2.36%	+10.02%
Revenue	+0.12%	+0.70%	+1.97%	-11.11%	+0.34%	+0.89%	-6.60%
RevenueToInitValue	+0.11%	+0.70%	+1.96%	-11.10%	+0.33%	+0.89%	-6.59%
Revenue+Salvage	-0.84%	-0.06%	+0.16%	-1.53%	-0.40%	-0.01%	-2.76%
RevenuePerPv	+0.11%	+0.67%	+2.01%	-11.06%	+0.35%	+0.82%	-6.57%
RevenuePerOrder	+2.89%	-1.31%	-2.54%	-14.11%	-4.10%	-1.66%	-14.33%

more on purchase and traffic objectives rather than revenue, the sales is quite astonishing. When salvage value is took into account, **IB** algorithm performs the best among almost all dimensions except sales.

Inventory

We also compared the inventory-related measurement from two dimensions: the sold-out rate and leftover rate, as shown in Table 3.

Table 3. Inventory-related Results

	Does not Consider Salvage				Consider Salvage		
	LP-based		IB	REAL	LP-based	IB	REAL
	-1day	-1seg			-1seg		
Sold-out Rate	-3.06%	+0.33%	+3.29%	+2.02%	+8.14%	+2.15%	+10.08%
Leftover Rate	+0.00%	-2.62%	-6.63%	-2.61%	-7.65%	-4.52%	-7.16%

Computational Time

Although LP-based algorithm is widely accepted, its computational time is not a neglectable point in real-time recommendation system. However, in our experiment, its computational time is 74.43s, on average, and the standard deviation is 73.64. Meanwhile, other algorithms only involve multiplication and addition, whose computational time is ignorable.

5 CONCLUSION

In this paper,

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 Mirrokni, and Qiqi Yan. Whole-page optimization and submodular welfare maximization with online bidders. 4(3):305–322, 2013.

[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.

[21] Xi Chen, Will Ma, David Simchi-Levi, and Linwei Xin. Assortment planning for recommendations at checkout under inventory constraints. *Available at SSRN 2853093*, 2016.