ReducE-Comm: Effective Inventory Reduction System for E-Commerce

Shay Gershtein
Tel Aviv University
shayg1@mail.tau.ac.il

Tova Milo
Tel Aviv University
milo@cs.tau.ac.il

Slava Novgorodov
eBay Research
snovgorodov@ebay.com

ABSTRACT

Many e-commerce platforms serve as an intermediary between companies/manufacturers and consumers, receiving a commission per purchase. To increase revenue, such sites tend to offer a wide variety of items. However, in many situations a smaller subset of the items should be selected and offered for sale, e.g., when opening an express branch or expanding to a new region, or when maintenance costs become prohibitive and redundant items should be disposed of. In all these cases selecting a reduced inventory which covers most consumer needs is an important goal.

In this demo we introduce ReducE-Comm - a highly parallelizable and scalable system that given a large set of items, a bound on the number of items that can be supported and information about consumer preferences/items relationships, allows to select a subset of the items which maximizes the likelihood of a purchase. Our system is interactive and facilitates real-time analysis, by providing detailed per-item impact statistics. We demonstrate the effectiveness of ReducE-Comm on real-world data and scenarios taken from a large e-commerce system, by interacting with the CIKM'19 audience who act as analysts aiming to intelligently reduce the inventory.

ACM Reference Format:
Shay Gershtein, Tova Milo, and Slava Novgorodov. 2019. ReducE-Comm: Effective Inventory Reduction System for E-Commerce. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3357861

1 INTRODUCTION

Due to the rapid growth of the e-commerce industry, online selling has become one of the most trending businesses of today. Many e-commerce platforms serve as an intermediary between companies/manufacturers and consumers, receiving a commission per purchase. To increase the number of sales, such sites tend to offer a large number of items . Nevertheless, they often pursue complementary projects where selecting and offering a reduced inventory is required. For example, when companies provide express delivery services (alongside existing services), offering items which are ready for next-day-delivery, these should be available in different warehouses for immediate shipping. It is often not feasible to ensure immediate availability for all items, as seen for example in Amazon Prime which offers a small percentage of the entire Amazon catalog. Similarly, expansion to new regions is often done gradually due to regulations, initially offering a small backlog of items. Finally, maintaining large inventories incurs a substantial maintenance overhead, hence companies tend to periodically dispose of some small percentage of items deemed to be least valuable. In all these examples the goal is to adhere to some size constraints while minimizing the loss in the number of predicted sales compared to offering the entire catalog.

A naïve, yet popular, solution is to focus on the top-k best selling items. This approach however entirely ignores the *hidden* relations between items. In particular, studies show that consumers, even when searching for a specific item, are often willing to buy in its absence what they consider to be a satisfying alternative [9]. For example, in the absence of a specific 50 inch Samsung TV a consumer may be willing to settle for a slightly bigger Samsung TV or for the same size TV from Philips. Retaining a set of items which are not only popular in-and-of-themselves, but are also likely to "cover" the inventory by serving as suitable alternatives for omitted items, can significantly improve the overall satisfaction of the customers.

The problem we address in this demo, which we call the *Preference Cover problem*, is to select, given a large set of items and a bound on the number of items that can be supported, which items to retain, such that consumer satisfaction and the likelihood of a purchase are maximized. To solve this problem we introduce ReducE-Comm, an end-to-end system, which, in addition to items selection, also facilitates real-time analysis. ReducE-Comm models consumer preferences via a *preference graph* - a directed graph with weights on both nodes and edges. The nodes correspond to items, and their weights reflect the items' purchase popularity (% out of total sold items). A directed edge from item A into item B indicates that, in A's absence, consumers consider B as a possible alternative. The edge weight reflects the probability that a consumer is willing to buy B as an alternative to A, if A is missing[1]. We discuss how ReducE-Comm derives the graph structure and edge weights from the available clickstream data in the System Architecture section.

ReducE-Comm uses the *preference graph* to devise effective algorithms for the selection of items. Before discussing the algorithms let us first illustrate through a simple example how the information provided in the graph is employed, and why selecting the top-k most sold items is not necessarily ideal.

Example 1.1. *Assume that there are five available items - iPhone 8 64GB in three different colors (A - Gold, B - Space Gray, C - Silver),*

[1]We assume that all transitive relationships, when/if exist, are directly represented in the graph by single edges.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357861

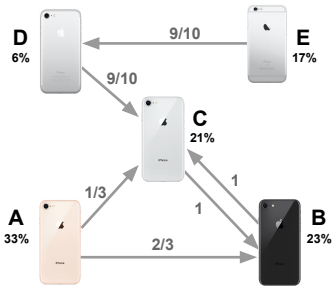

Figure 1: Sample graph of items

iPhone 7 Silver 64GB (D) and iPhone 6 Silver 64GB (E). Out of these items we wish to select and retain only two. Consider the preference graph depicted in Figure 1. We can see that A is the best selling item (purchased by 33% of the customers) while D is the least sold (6%). We can also see that consumers interested in item E (resp. D) are likely to settle in its absence for D (C), but will not transitively buy C. Such behavior is common, for instance, when D (resp. C) is a one-step upgrade of E (D): people are often flexible and willing to add a small amount of money in return for an upgrade, but a two-steps delta may be overly expensive. We can also see in the graph that consumers interested in C (B) will settle in its absence for B (C), and that B is a more likely replacement for A than C.

If we select the top two most sold items, A and B, we are likely to cover about 77% of the requests (the ones interested in A and B, as well C, who in its absence are likely to purchase B). Interestingly, a more careful analysis (which we only describe here intuitively and will formalize in the following section) shows that in fact retaining B and D (the least sold item!) is the optimal solution, covering about 87.3% of the requests. Intuitively this is because B covers much of the requests for A, B, and C, whereas D covers most of the requests for itself as well as for E. In this simple example the set of requests served by the two retained items are disjoint, but a similar analysis can be applied to general overlapping cases.

The explicit formula for determining the probability of a purchase is contingent on the dependencies between choosing different alternatives. Such dependencies can be complicated, hence for a model to be practical it should simplify them in a manner which approximates well real life settings. We consider in this demo two variants of the Preference Cover problem, the *Independent* and the *Normalized* variants, which differ by the semantics of edge dependencies. We have shown in our experiments [2] that consumer data from different e-commerce settings tends to be captured well by at least one of these variants. The Independent variant assumes independence between all alternatives. Whereas, the Normalized variant assumes that each consumer has at most one item that she considers as a suitable alternative, and thus the sum of weights of outgoing edges from any given item is bounded by 1 (hence the name Normalized).

Due to the NP-Hardness of both variants of the problem, ReducE-Comm employs a dedicated highly parallelizable and scalable greedy scheme which provides approximated solutions for both. The solutions come with approximation guarantees, which differ for each variant. For the Independent variant the approximation guarantee is optimal for a polynomial time algorithm, as it tightly matches an inapproximability bound. For the Normalized variant, optimality is an open problem but the algorithm matches the best approximation factor known for practical algorithms of an extensively studied equivalent problem. We prove all the results stated above in [2].

Our solution has several added values. First, our algorithms also allow to directly solve the complementary minimization problem, where, instead of an upper bound on the number of items to retain, one is given a lower bound on the percentage of item requests that should be covered (and the goal is to identify a minimal size set of items that achieves the required coverage). Note that a naïve solution for this complementary problem can be obtained, via binary search on the target set size, by running any algorithm for the original problem. But this incurs a $O(\log n)$ factor overhead, n being the number of available items. Our direct greedy approach allows to avoid this overhead. Additionally, the data computed by the algorithms also allows to support real-time analysis. In particular, ReducE-Comm produces, on top of the selected subset of items, detailed statistics for each item. The analyst can see for every non-selected item, what percentage of the requests for this item each retained alternative covers, and adjust the requirements (e.g. requiring specific items to be selected or omitted from the selected subset), with the system computing a modified solution in real time.

Finally, we note that in the problem setting that we study here (which is common to many intermediary platforms [1]) the commission-per-purchase is considered fixed and the intermediary platform is indifferent to the items' cost/revenue or the physical storage that they require.

Demonstration Overview. We demonstrate the operation of ReducE-Comm over real-world e-commerce data. Our demonstration reenacts a scenario where an e-commerce platform is opening a new express branch, and wishes to select items for this reduced store variant. The audience will play the role of the analysts that design the new store and need to offer the optimal subset of items.

2 TECHNICAL BACKGROUND

We next intuitively introduce the *Preference Cover* problem, and briefly describe its two concrete variants - Independent and Normalized. Full technical details can be found in [2].

Formally, we represent consumer preferences via a *preference graph* which, along with an integer k, serve as input for the Preference Cover problem. A preference graph $G = (V, E, W_V, W_E)$ is a directed graph with weighted vertices and edges. The vertex set V corresponds to n items. For each vertex v, its weight, $W_V(v) \in [0, 1]$, is defined as the probability of v being requested by a consumer. The sum of all node weights is therefore 1. For each edge from v into a node u, its weight, $W_E(v, u) \in (0, 1]$, implies the probability of u matching a request for v as an alternative.

Given a number k, our goal is to choose a subset $S \subseteq V$, $|S| = k$, of items, marking them as *retained*. Given a request for item v, if it is retained, the request is considered *matched*. Otherwise, if v is not retained, a request has some probability of being matched by another retained neighboring item of v, as indicated by the weights of edges outgoing from v. We define a target function, termed as the *cover function*, $C : 2^V \rightarrow [0, 1]$, s.t. assuming $S \subseteq V$ is the retained set of items, $C(S)$ is the probability a request drawn from the distribution indicated by the node weights is matched. The *Preference Cover problem* aims to compute $\arg\max_{S, |S|=k} C(S)$.

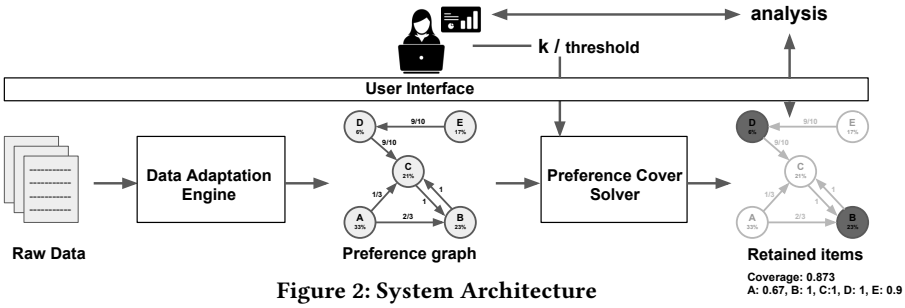

Figure 2: System Architecture

An explicit formula for computing $C(\cdot)$ is contingent on the dependencies between the probabilities indicated by the edges. In this demo we consider two variants of the problem which approximate well common real life scenarios: the *Independent* variant assumes that the probabilities modeled by edges are independent, while the *Normalized* variant assumes that each given consumer considers at most one item as a most suitable alternative.

In both variants when considering alternatives for an item v, we only take into account v's neighbors, and do not consider longer paths, corresponding to the process of considering an alternative followed by an alternative to that alternative and so on. Such paths are already taken into account into the edges and their weights, thus intuitively the preference graph is the transitive closure of a graph modeling the probabilities to correspond to such paths.

We now formally describe the two variants. In the presentation below, given a set S of retained items and a node v, $R_v(S) = \{u|(v,u) \in E, u \in S\}$ denotes its set of retained alternatives.

Independent variant. In the *Independent* variant we assume complete independence between edges - the probability a given alternative matches a request is not affected by whether or not a different alternative matches it. Thus, the probability of **not** matching a request for a non-retained item v, which occurs when no retained alternative is suitable, is, due to independence, the product of all such probabilities, $\prod_{u \in R_v(S)}(1 - W_E(v,u))$, implying that the probability of the complement of this event, which is matching the request, is $1 - \prod_{u \in R_v(S)}(1 - W_E(v,u))$. Hence, the formula is:

$$C(S) = \sum_{v \in S} W_V(v) + \sum_{v \in V \setminus S} \left[W_V(v) \cdot (1 - \prod_{u \in R_v(S)} (1 - W_E(v,u))) \right]$$

Note that the first addend in the last formula is due to the fact that requests for retained items are matched w.p. 1. Similarly, the second addend corresponds to summing over all items not in S, for each such item v adding the probability it is both requested ($W_V(v)$) and covered by S ($(1 - \prod_{u \in R_v(S)}(1 - W_E(v,u)))$.

Normalized Variant. In the *Normalized* variant we assume that each consumer considers at most one item as an alternative. Alternatives are therefore dependent, in the sense that an alternative matching the request implies that all others do not. It follows that the sum of the weights of all edges outgoing from any given node is at most 1, and given a request for a non-retained item v, the probability it is matched is: $\sum_{u \in R_v(S)} W_E(v,u)$. Hence, the formula for $C(S)$ is:

$$C(S) = \sum_{v \in S} W_V(v) + \sum_{v \in V \setminus S} \left[W_V(v) \cdot \sum_{u \in R_v(S)} W_E(v,u) \right]$$

Intuitively, the Independent variant fits when the opinion on the suitability of a given alternative is not demonstrated to be strongly dependent on most consumers' opinion of other alternatives. The dependencies are either insignificant or cancel out when summed over the entire user base. On the other hand, the Normalized variant is suited for domains where consumer requests are often specific in nature, and the number of suitable alternatives is very small.

Algorithms. Our greedy algorithm, that fits both variants, at each of its k iterations selects an item which maximizes the marginal gain. Concretely, at each iteration it computes, for each item not selected so far, the cover function value obtained by selecting it. The item whose addition achieves the highest value is then selected.

Scalability and performance. Our greedy approach, along with its parallelization properties, allows the solution to be highly scalable, to such an extent that we can in real time solve the problem for $k = n$, thus simultaneously deriving solutions for all values of k, by considering only the output prefix of the corresponding size. Concretely, previous calculations are stored and leveraged for efficient computation, focusing only on the effects of adding new items, despite any overlap with previously added items. As the computations of the gain obtained by adding different items are independent, they can be performed in parallel. Moreover, for any given added item, its cover as an alternative of any non-retained item can be computed in parallel as well. The pseudocode of our algorithm and a detailed performance analysis can be found in [2].

Approximation guarantees. While the problem is NP-hard, our PTIME algorithm comes with approximation guarantees for each variant. For the Independent variant the guarantee is optimal, as it matches an inapproximability bound. For the Normalized variant, optimality is an open problem but the algorithm matches the best guarantee known for practical algorithms of an equivalent problem.

3 SYSTEM ARCHITECTURE

We implemented ReducE-Comm using Python and Flask. The system consists of a User Interface and two main modules: the *Data Adaptation Engine* and the *Preference Cover Solver*. The analyst interacts with the system via the UI. The Data Adaptation Engine takes as input raw e-commerce data and builds the corresponding preference graph. The raw data is essentially a clickstream that is collected by almost all e-commerce platforms for future analytics. The most minimalist clickstream, which is assumed to be available, consists of clicked and purchased items, grouped by separate user sessions. The data is analyzed to derive the graph and determine which of the two models - Independent or Normalized - fits the data. For space constraints we omit the description of this process and refer the reader to the technical report in [2] for full details.

The constructed preference graph is then passed as input to the Preference Cover Solver, along with k, the required number of retained items. The solver runs our greedy algorithm, adapted to the given variant. The solver produces a list S of retained items (in the order in which the items were added by the algorithm), accompanied with valuable metadata, such as $C(S)$, and how well each item is covered by its retained alternatives.

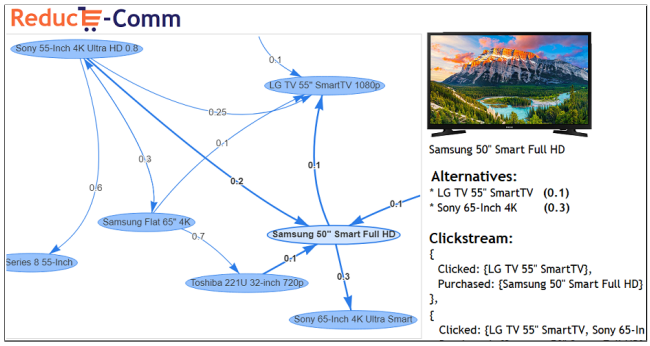

Figure 3: Graph creation

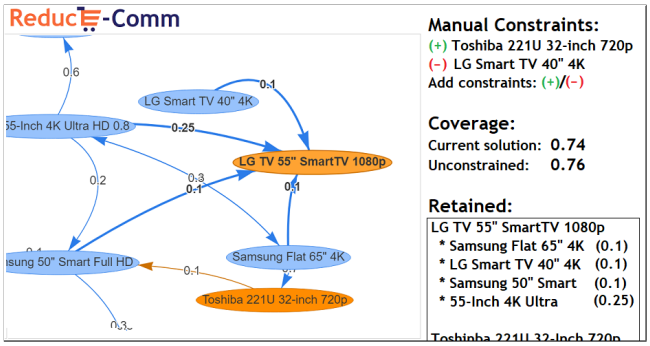

Figure 4: Items reduction result

4 DEMONSTRATION SCENARIO

We demonstrate the operation of ReducE-Comm, a system that enables efficient inventory reduction, supports various types of "what-if" analysis and provides detailed statistics about the effects of the proposed reduction. Our demonstration reenacts a real-life scenario where an e-commerce platform is opening a new branch with express delivery operations, and wishes to select items for this reduced store variant. The audience will play the role of the e-commerce analysts that design the new store and need to select the best subset of items to offer, such that the sales are maximized.

The demonstration consists of two parts. First, we show how the graph is derived from clickstream data. This graph is normally computed offline, but to enable real-time performance we compute it here over a single category of items (Electronics). Second, we show over a graph, created in advance pertaining to the entire catalog, how our algorithms select reduced item sets, and how this solution changes when various user constraints are introduced.

Graph Creation The analyst will start by loading the clickstreams from the Electronics category. Once the graph creation process is completed, we will visualize the graph, and explain how it was derived from the raw data. Namely, for any node or edge pointed by the user, the system will highlight the relevant part of the clickstream (as depicted in Figure 3), and we will explain how it was created and how its corresponding weight was assigned.

Analysis To continue our demonstration, we will load the full catalog graph, and the analyst will initiate the process of inventory reduction by specifying the number of items she wants to retain (k). When the algorithm terminates (it has a real-time performance, hence its entire operation can be easily incorporated in the live demonstration) we will examine the result, which details for each item suggested for removal, to what extent it is covered by each of its retained alternatives. Next, the analyst may adjust the solution by requiring that some omitted items are retained (e.g. if the e-commerce platform has an agreement with specific suppliers) or that some retained items are removed. ReducE-Comm will interactively adjust the solution, recomputing one that complies to the constraints , and promptly present it to the analyst, along with a compact visualization of the changes (thereby allowing for a "what-if" analysis). Figure 4 depicts results produced by the algorithm along with the manual constraints subsequently added by the analyst, and the outcome of this modification.

We conclude our demonstration with a discussion of other potential usages of the system, such as solving the problem where the analyst asks for a subset of minimal size which exceeds a specified threshold for the percentage of matched requests.

Related Work E-commerce related problems attracted the interest of many researchers in recent years. Some of the extensively studied problems are product classification (e.g. [8]), ranking of products in search results (e.g. [5]), products recommendations (e.g [7]) and generation of product data (e.g. descriptions [6]).

Close to us in spirit are works on *diversity* [10], producing the most diverse subset of elements. The similarity is evident when elements are weighted by importance and when, as in [4], each non-selected element must be covered by a similar selected item. However, a key difference is that we do not maximize diversity, rather it is a feature typical of good solutions, yet not necessary. Moreover, in weighted diversity problems the goal is to maximize the weight of the diverse subset, whereas in our model one also partially counts weights of adjacent items, to account for alternatives. Another related line of work is recommendations [7], as it also deals with selecting a subset of items to increase purchasing probability. However, there are important qualitative and quantitative differences. Primarily, recommendations are typically personalized w.r.t. a given user or product, and deal with a much smaller k.

Closest to our work is a subfield of Operations Research called Assortment Optimization. Notably, the Markov chain choice model (e.g. [3]) bears resemblance to the Normalized variant of our model, but is more complicated by considering item prices and multiple-step graph paths. Due to this complexity, these works examine only small scale item sets, and the results are not scalable to big data.

Acknowledgements This work has been partially funded by the Israel Innovation Authority, the Israel Science Foundation, the Binational US-Israel Science foundation, Len Blavatnik and the Blavatnik Family foundation.

REFERENCES

[1] Marketplace Pricing Model. https://marketplace.webkul.com/marketplace-pricing-model-subscription-vs-commission/.
[2] ReducE-Comm (Tech. Report) http://slavanov.com/research/reduce-comm_tr.pdf.
[3] J. Blanchet, G. Gallego, and V. Goyal. A markov chain approximation to choice modeling. *Operations Research*, 64(4):886–905, 2016.
[4] M. Drosou and E. Pitoura. Multiple radii disc diversity: Result diversification based on dissimilarity and coverage. *TODS*, 40(1):4, 2015.
[5] S. K. Karmaker Santu, P. Sondhi, and C. Zhai. On application of learning to rank for e-commerce search. In *SIGIR*, 2017.
[6] S. Novgorodov, I. Guy, G. Elad, and K. Radinsky. Generating product descriptions from user reviews. In *Proc. of WWW*, 2019.
[7] L. Qi, X. Xu, X. Zhang, W. Dou, C. Hu, Y. Zhou, and J. Yu. Structural balance theory-based e-commerce recommendation over big rating data. *IEEE Transactions on Big Data*, 4(3):301–312, 2018.
[8] C. Sun, N. Rampalli, F. Yang, and A. Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *PVLDB*, 7(13), 2014.
[9] W. Verbeke, P. Farris, and R. Thurik. Consumer response to the preferred brand out-of-stock situation. *European Journal of Marketing*, 32:1008–1028, 1998.
[10] Y. Wang, A. Meliou, and G. Miklau. Re-index: Diversifying answers to range queries. *PVLDB*, 11(7):773–786, 2018.