An Efficient Premiumness and Utility-based Itemset Placement Scheme for Retail Stores

Parul Chaudhary[1], Anirban Mondal[2], Polepalli Krishna Reddy[3]

[1] Shiv Nadar University, India
pc230@snu.edu.in
[2] Ashoka University, India
anirban.mondal@ashoka.edu.in
[3] International Institute of Information Technology, Hyderabad, India
pkreddy@iiit.ac.in

Abstract. In retail stores, the placement of items on the shelf space significantly impacts the sales of items. In particular, the probability of sales of a given item is typically considerably higher when it is placed in a *premium* (i.e., highly visible/easily accessible) slot as opposed to a non-premium slot. In this paper, we address the problem of maximizing the revenue for the retailer by determining the placement of the itemsets in different types of slots with varied *premiumness* such that each item is placed *at least once* in any of the slots. We first propose the notion of *premiumness of slots* in a given retail store. Then we discuss a framework for *efficiently* identifying itemsets from a transactional database and placing these itemsets by mapping itemsets with different revenue to slots with varied premiumness for maximizing retailer revenue. Our performance evaluation on both synthetic and real datasets demonstrate that the proposed scheme indeed improves the retailer revenue by up to 45% w.r.t. a recent existing scheme.

Keywords: Utility mining; retail management; pattern mining; data mining.

1 INTRODUCTION

In retail stores, the placement of items on the shelf space significantly impacts sales of items and consequently, sales revenue as well [1-5]. However, strategic placement of items in a manual manner for maximizing the sales revenue is not practically feasible in case of medium-to-large retail stores, which may typically occupy more than a million square feet of retail floor space. Examples of such medium-to-large retail stores include Macy's Department Store at Herald Square (New York City, US), Shinsegae Centumcity Department Store (Busan, South Korea) [6] and Dubai Mall.

Consider a large retail store with multiple aisles, where each aisle contains items that are stocked in the slots of the shelves. Notably, all of these slots are not created equal i.e., they differ in terms of *premiumness*. The premiumness of a given slot represents the extent to which the slot is visible and/or physically accessible to the shoppers. It is a well-established fact in the retail industry that the probability of sales of a given item i increases with increase in the premiumness of the slot, in which i is placed [7].

Hence, this work considers that the probability of sales of a given item typically varies depending upon the *premiumness* of the slot in which the item is placed. Examples of slots with high premiumness include the *"impulse buy"* slots at the checkout counters of retail stores and slots that are near to the eye or shoulder level of users. Slots with low premiumness are typically located very high or very low in the retail store shelves.

In general, we can have N types of slots based on premiumness. Moreover, there would be ***multiple blocks of each of the slot types*** across the different aisles in a given large retail store, and each block would contain several slots of a given slot type. The research issue here is to devise a methodology for strategically placing the items in the slots of varied premiumness such that the revenue of the retailer is maximized.

We address the problem of maximizing the revenue for the retailer by determining the placement of the itemsets in the different types of slots with varied premiumness such that each item is placed *at least once* in any of the slots [24]. We need to ensure that every item is represented *at least once* in the slots of a retail store so that the retailer does not lose customers due to lack of the convenience of "one-stop" shopping.

A possible solution for the retailer could be to sort all of the items in descending order of their respective prices and then place the highest-priced items in the most premium (i.e., highly visible) slots and the lowest-priced items in the least premium slots. Similarly, mid-priced items could be placed in the slot types, whose premiumness is in the mid-range (i.e., moderately visible). However, this solution does not consider the fact that customers typically buy sets of items (i.e., *itemsets*) together; hence, it fails to exploit the *associations* between the items, thereby compromising retailer revenue. Alternatively, the retailer could fill the highly-premium slots with the most *frequent itemsets* [9-10]. However, frequent itemsets do not consider item prices (utility values), which can vary considerably across items (e.g., a Rolex watch versus a can of Coke). Since revenue depends upon both frequency of sales and item prices, this solution may not maximize revenue as some of the frequent itemsets could have low revenue.

Utility mining approaches have been proposed for extracting the patterns by considering the respective utility values of items and itemsets. In this regard, utility can be defined in various ways e.g., in terms of revenue, interestingness and user convenience, depending upon the application. Thus, utility mining approaches focus on determining high-utility itemsets from transactional databases. Existing utility mining approaches focus on the following aspects: determining minimal high-utility itemsets [11], creating representations of high-utility itemsets [12], reducing candidate itemset generation overheads by using the utility-list [16] and the UP-Tree [18], and identifying the upper-bounds and heuristics for pruning the search space [14-15].

Notably, *none* of the existing utility mining and pattern mining approaches have considered the problem of *itemset placement in retail slots of varied premiumness*.

In this paper, we use a utility mining approach for maximizing the revenue for the retailer by determining the placement of the itemsets in the different types of slots with varied premiumness. We designate this approach as the **Premiumness and Revenue-based Itemset Placement (PRIP)** scheme. We consider that if a set of items (i.e., an itemset) is frequently purchased together, it indicates that a typical customer is enticed to buy all of these items *together* instead of buying each of the items (in that itemset) individually. In other words, the customers, who purchased this itemset, would possibly

not have purchased it if one or more of the items had been missing from the itemset. We focus on identifying high-revenue itemsets from the past history of the user purchase transactions and placing them in slots with high premiumness. Consequently, the probability of sales of the high-revenue itemsets would increase, thereby contributing to higher revenue for the retailer. The key contributions of this work are three-fold:

1. We propose the notion of *premiumness of slots* in a given retail store and introduce the problem of placement of itemsets in slots of varied premiumness for facilitating improved revenue for the retailer.
2. We propose a framework for *efficiently* identifying itemsets from a transactional database and placing these itemsets by mapping itemsets with different revenue to slots with varied premiumness for maximizing retailer revenue.
3. We conducted an extensive performance evaluation using both synthetic and real datasets to demonstrate that the proposed scheme is indeed effective in improving the total retailer revenue by up to 45% w.r.t. a recent existing scheme.

We use revenue as an example of a utility measure throughout this paper; hence, we use the terms *revenue* and *utility* interchangeably. The remainder of this paper is organized as follows. Section 2 discusses related works and background information. Section 3 formulates the problem. Section 4 presents the proposed PRIP scheme. Section 5 reports the performance evaluation. Finally, we conclude in Section 6.

2 RELATED WORK AND BACKGROUND

This section discusses existing works and background information about the kUI index.

2.1 RELATED WORK

Association rule mining approaches [8-10] use the downward closure property [8] for finding frequent itemsets above a support threshold [9], but they do not consider item utility. Hence, utility mining approaches [11-19] have been proposed for finding high-utility patterns, but they do not satisfy the downward closure property.

The notion of MinHUIs (minimal high-utility itemsets) and a representation of high-utility itemsets [21] have been proposed in [13]. MinHUIs allude to the smallest itemsets that generate high utility. The HUG-Miner and GHUI-Miner algorithms [12] mine concise representations of high-utility itemsets and prune the number of candidate itemsets for extracting the high-utility itemsets. The EFIM algorithm [14] determines high-utility itemsets by using two upper-bounds, namely *sub-tree utility* and *local utility*, for pruning the search space. The EFIM-Closed algorithm [15] finds closed high-utility itemsets by leveraging pruning strategies and upper-bounds for utility.

The Utility Pattern Growth (UP-Growth) algorithm [18] uses the Utility Pattern Tree (UP-Tree) to maintain information about high-utility itemsets for generating candidate itemsets. The HUI-Miner algorithm [16] uses the utility-list for storing information (including utility) about the itemsets for avoiding expensive candidate itemset generation. The CHUI-Miner algorithm [17] for mining closed high-utility itemsets also computes

itemset utility values without generating candidates. The LHUI-Miner algorithm [25] mines local high-utility itemsets by considering that itemset utilities vary over time. The DMHUPS algorithm [26] uses a data structure, designated as IUData List, to discover multiple high-utility patterns simultaneously. The work in [19] prunes away low-utility itemsets and considers business goals as well when evaluating utility values.

Approaches for improving the placement of itemsets in retail stores have been discussed in our previous works in [22, 23]. The work in [22] focused on placing itemsets in the slots of the retail stores when the physical sizes of the items are variable e.g., large-sized golfing equipment versus a small can of Coke. The work in [23] aimed at diversifying the placement of itemsets in retail stores for improving the retailer's long-term revenue. Moreover, the work in [27] uses a mixed-integer programming model to study retail assortment planning and retail shelf space allocation for maximizing retailer revenue. However, these works do not consider the aspect of slot premiumness.

Notably, *none* of the existing utility mining and pattern mining approaches consider the problem of itemset placement in retail slots of varied *premiumness*. This limits their applicability to building practical systems for itemset placement in retail stores for maximizing the revenue of the retailer.

2.2 BACKGROUND

This section discusses the background about kUI index. We discuss the kUI index [23], which our proposed scheme PRIP exploits for extracting high-revenue itemsets of varied sizes. kUI is a multi-level index, where each level concerns a given itemset size. At the k^{th} level, the kUI index stores the top-λ high-revenue itemsets of itemset size k. Each level corresponds to a hash bucket. For indexing itemsets of N different sizes, the index has N hash buckets i.e., one hash bucket per itemset size. Hence, given a query for finding the top-λ high-revenue itemsets of a given size k, one can traverse quickly to the k^{th} hash bucket instead of having to traverse through all the hash buckets corresponding to $k = \{1, 2, \ldots, k\text{-}1\}$.

Now, for each level k in the kUI index, the corresponding hash bucket contains a pointer to a linked list of the top-η itemsets of size k. (Here, $\eta > \lambda$.) The entries of the linked list are of the form (*itemset*, σ, ρ, NR), where *itemset* refers to the given itemset under consideration, σ refers to the frequency of sales of the given itemset and ρ refers to the price of the given itemset. Here, NR (net revenue) is the product of σ (frequency of sales) and ρ (price) of items in *itemset*. The entries in the linked list are sorted in descending order of NR for quick retrieval of the top-λ itemsets of a given size k.

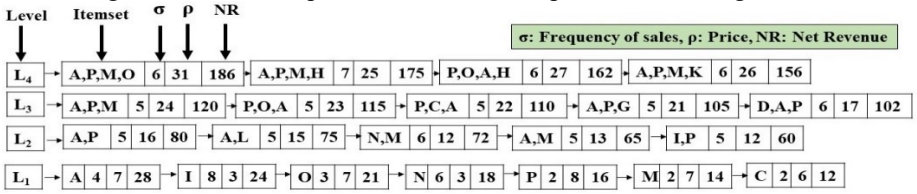


Fig. 1. Illustrative Example of the kUI Index

Figure 1 depicts an illustrative example of the kUI index. Observe how the itemsets (e.g., {A}, {I}) of size 1 correspond to level 1 of the index, the itemsets of size 2 (e.g., {A, P}, {A, L}) correspond to level 2 of the index and so on. Notice how the itemsets are sorted in descending order of NR.

3 CONTEXT OF THE PROBLEM

Consider a set D of user purchase transactions on a finite set Υ of *m* items $\{i_1, i_2, i_3,.., i_m\}$, where each transaction comprises a set of items from set Υ. Each item i_j of set Υ is associated with a price ρ_j and a frequency of sales σ_j. We assume that each item occurs in any given transaction *only once*; hence, σ_j is essentially equivalent to the number of transactions in which a given item has appeared. Furthermore, an itemset comprises a set of items from set Υ. We assume that each item of set Υ is physically of the same size i.e., each item consumes an equal amount of space e.g., on the retail store's shelves.

We assume that all slots are of equal size, but they vary in terms of *premiumness*. Recall the notion of premiumness of the slots as discussed in Section 1. Slots in a retail store have varied premiumness i.e., certain slots have a higher probability of sales for items placed in them. We envisage N different types of slots with varied premiumness.

Problem statement: Consider a set D of user purchase transactions on a finite set Υ of *m* items. Each item is associated with a price ρ_j and a frequency of sales σ_j. Consider N types of slots with varied premiumness. This paper addresses the problem of maximizing the total revenue for the retailer by determining the placement of the itemsets in the different types of slots with varied premiumness such that each item is placed *at least once* in any of the slots to facilitate one-stop shopping (as motivated in Section 1).

4 PROPOSED SCHEME

We first explain the notions of *revenue contribution* and *aggregate revenue contribution*, which constitute the basis of our proposed scheme. Next, we discuss the scheme.

4.1 Revenue Contribution and Aggregate Revenue Contribution

In practice, some of the items typically contribute more to the revenue of the retailer than other items. An item *i* may be purchased together in association with other items (i.e., in an itemset) by a large number of users, and even if the price of *i* is low, it may still contribute significantly to the revenue of the retailer. We consider that if a given item *i* was missing from the slots, many of the users may not have purchased those itemsets i.e., the presence of item *i* has encouraged the sales of those itemsets.

Based on this motivation, we introduce the notion of the **aggregate revenue contribution (ARC)** of a given item across all of the user purchase transactions. Given a set of user purchase transactions, ARC essentially quantifies the extent to which a given item contributes (in terms of revenue) to the overall revenue of the retailer. For purposes

of computing ARC for a given item, we consider that each of the items in a given transaction contributed *equally* towards the user's purchase decision. This is because in practice, we would typically have no knowledge of the direction of causation of the sales i.e., given a transaction {A, B, C}, we do not know whether the customers purchased A because of B and C or if she purchased C because of A and B.

Purchase value (PV) of a given transaction T is the amount of money (revenue) that the user paid for buying all of the items in T. We compute the ***revenue contribution (RC)*** for each item in a given transaction T by dividing the purchase value (PV_T) of T by the number of items in T. Given a transaction T containing n_T items, we define the revenue contribution (RC_T) of each item in T as $RC_T = PV_T / n_T$. Now, for each item, we sum up its values of RC across all of the user purchase transactions to obtain the ***aggregated revenue contribution (ARC)*** for each item. Given q transactions, we compute the value of ARC for the k^{th} item i_k as $ARC_k = \sum_{j=1}^{q}(RC_j)$. Observe that if the j^{th} transaction does not contain the item i_k, the value of RC_j would be zero.

Figure 2 depicts five transactions involving items A to I and the corresponding purchase values of each of these transactions. Here, PV of transaction {A, C, G, I} is 63 and RC of each of the items in the itemset is 63/4 i.e., 15.75. Observe how the value of RC for item A differs across the itemsets. Finally, we sum up all of the RC values of item A to obtain the ARC of item A as (30+15.75+12= 57.75) i.e., 57.75.

PV = Purchase Value
RC = Revenue Contribution
ARC = Aggregated Revenue Contribution

Transaction	PV	RC
A, E	60	60/2= 30
A, C, G, I	63	63/4= 15.75
A, C, G	36	36/3= 12
B, C	24	24/2= 12
D, F, G	28	28/3= 9.33

Item	ARC
A	30+15.75+12= 57.75
C	15.75+12+12= 39.75
G	15.75+12+9.33= 37.08
E	30
B	12
D	9.33
F	9.33

Fig. 2. Illustrative Example for the Computation of RC and ARC

4.2 Premiumness and Revenue-based Itemset Placement (PRIP)

It is well-known in the retail industry that strategic placement of items in the retail store slots has significant impact on sales [7]. Slots in a retail store typically vary in terms of *premiumness* and items placed in slots of higher premiumness have a higher probability of sales [7]. If we arbitrarily (or randomly) assign and place the items in the slots, it may fail to maximize the revenue of the retailer. For example, a high-revenue item may get placed in a slot with low premiumness, thereby compromising retailer revenue. Thus, there is an opportunity to improve the revenue of the retailer by considering user purchase patterns in conjunction with a methodology for placing items in slots based on the premiumness of the slots and the revenue generation potential of items.

We propose a framework for *efficiently* identifying itemsets from a transactional database and placing these itemsets by mapping itemsets with different revenue to slots

with varied premiumness for maximizing retailer revenue. For placement of items in the retail slots, we extract potential 1-itemsets based on the values of ARC. We make the observation that the notion of ARC is only applicable to individual items and as such, it does not apply to multiple items in tandem. Hence, we extract itemsets with sizes greater than 1 based on the net revenue of those itemsets.

Our proposed scheme works as follows. First, for placing the 1-itemsets, we place the items with higher ARC values in the slots with higher premiumness, and we place the items with lower values of ARC in the slots with lower premiumness and so on. Then, for placing itemsets with sizes greater than 1 in the remaining available slots, we extract itemsets of varied sizes (starting from itemset size of 2 onwards) from the kUI index [23] (discussed in Section 2.2) to allocate itemsets of varied sizes to the slots of varied premiumness based on the net revenue of the itemsets. Observe that placing itemsets of varied sizes in the slots can potentially provide improved revenue for the retailer because it would possibly address the needs of different user segments.

The proposed approach primarily consists of two phases. In the first phase, individual items are assigned to slots with varied premiumness based on the ARC of the items and the premiumness of the slots. In the second phase, itemsets of size greater than 1 are mapped to slots based on itemset revenue and (slot) premiumness. Notably, itemsets (of different sizes) are extracted from the kUI index. As discussed earlier in Section 2.2, recall that the kUI index contains itemsets of different sizes as well as the frequency of sales, price and the net revenue of each itemset. Moreover, the itemsets in the kUI index are kept sorted in descending order of net revenue. Algorithm 1 depicts our proposed PRIP scheme. It takes as input a database D of user purchase transactions, slots of N types and the number n_i of slots pertaining to each of the slot types, and the kUI index.

In Lines 1-4, we perform the necessary initializations. In Lines 5-7, observe how the database D of user purchase transactions is scanned to compute ARC for each item (see Section 4.1). In Lines 8-9, notice how we determine the number of 1-itemsets to be allocated to each slot type based on the relative percentages of slots corresponding to each slot type. In Line 10, the items are sorted in descending order of ARC.

In Lines 11-16, starting from the slot type with the highest premiumness, we progressively fill up the corresponding highest-premiumness slots with the individual items (i.e., the 1-itemsets) starting with the item having the highest ARC. By traversing this list of items sorted in descending order of ARC, once we have filled up the required number of high-ARC items (as determined from the formula in Line 7) in the highest-premiumness slots, we move on to the next high-premiumness slot and repeat the same process as above. This process is essentially continued until all of the 1-itemsets have been assigned to slots. Notably, after using the formula in Line 7, some of the low-ARC items may still remain unassigned to any slot due to the *"floor"* function used in the formula. Hence, in Line 17, for handling this corner case, we assign all of these remaining low-ARC items to the lowest-premiumness slot. Now all of the 1-itemsets have been assigned to slots i.e., each item is represented in the slots *at least once*.

In Lines 18-19, we update the number of slots available for each slot type to reflect that the 1-itemsets have already been placed in the respective slots. In Lines 20-34, observe how we use the kUI index to populate the remaining slots corresponding to each of the slot types. Starting from the highest-premiumness slot type, we keep filling

up the highest-premiumness slots as follows. We extract the top-revenue 2-itemset from the kUI index and place it in the highest-premiumness slot. Then we extract the top-revenue 3-itemset from the kUI index and place it in the highest-premiumness slot and so on. Upon reaching the topmost level of the kUI index, we do a round-robin and circle back to the next top-revenue 2-itemset, and then the next 3-itemset and so on until all of the highest-premiumness slots have been filled up. Once all the highest-premiumness slots have been exhausted, we move on to the next highest-premiumness slot and so on. This process is repeated until all of the slots have been filled up by items/itemsets.

Notably, in this work, we extract the itemsets in the afore-mentioned manner. However, it is also possible to fill up the slots with combinations such as (2+2+2), (3+3) or (2+4) itemsets and so on. We shall investigate such strategies in our future work.

Algorithm 1: **Premiumness and Revenue-based Itemset Placement (PRIP)**

Inputs: (a) A Transactional Database D on set I= $\{i_1,\ldots..i_m\}$ of m distinct items where each transaction T_i is of the form $<T_i, S_i, P_i>$, where S_i is set of items such that S_i is subset of I and P_i is the purchase value of T_i
(b) Slots of N types and n_i of slots pertaining to each of the slot type
(c) kUI Index

Output: Placement of the items/itemsets in the slots with varied premiumness

Begin

1. Initialize the aggregated revenue contribution ARC[*i*] of each item *i* to zero
2. Initialize slot type s_type to 1 and no. of items already placed ip to zero
3. Initialize an empty slot type array ST[i]
4. Initialize current available slot array CAS[i] and assign number of slot instances n_i to the array

/*Scan D to compute ARC of each item*/

5. **for each** transaction T_i in D {
6. **for each** item x belong to S_i {
7. ARC[x]= ARC[x]+ P_i/ $|S_i|$; } }

/*1-itemsets allocated to each slot type */

8. **for** (i = 1 to N) {
9. Y[i] = $[(n_i/\sum_{i=0}^{n} n_i)*m)]$; }
10. Sort the items in descending order of ARC[k] in list k-items into list Item[ ]
11. **for** each item k in Item[ ] {
12. **if** (Y[s_type]!=0) {
13. assign item k to slot type s_type
14. ip++ ; Y[s_type]— ; }
15. **if** (Y[s_type] = = 0) { s_type++; }
16. **if** (Y[s_type] = = N) { break; } }
17. Place the remaining items (m – ip) items into slot type s_type

/*Compute remaining CAS after 1-itemset placement*/

18. **for** (i = 1 to N) **{**
19. CAS[i] = CAS[i] - Y[i]; }

/*Scan kUI index to fill remaining slots*/

20. Scan the kUI index into a 2D array kUI[L, num[L]] ;
21. Set all elements of array h[lv] to 0
22. **for** each slot type (i=1 to N) {
23. **for** each level lv (lv = 2 to maxL){
24. h[lv]++;
25. **if** (CAS[i] < lv) {
26. lv--; h[lv]++;
27. Select the itemset h[lv]
28. Place itemset h[lv] in slot type i
29. CAS[i] = CAS[i] – lv;
30. break }
31. **else** {
32. Select the itemset h[lv] from lv
33. Place itemset h[lv] in slot type i
34. CAS[i] = CAS[i] – lv; }} **End**

4.4 Illustrative example of Proposed Scheme

Figure 3 depicts an illustrative example for our proposed PRIP scheme.

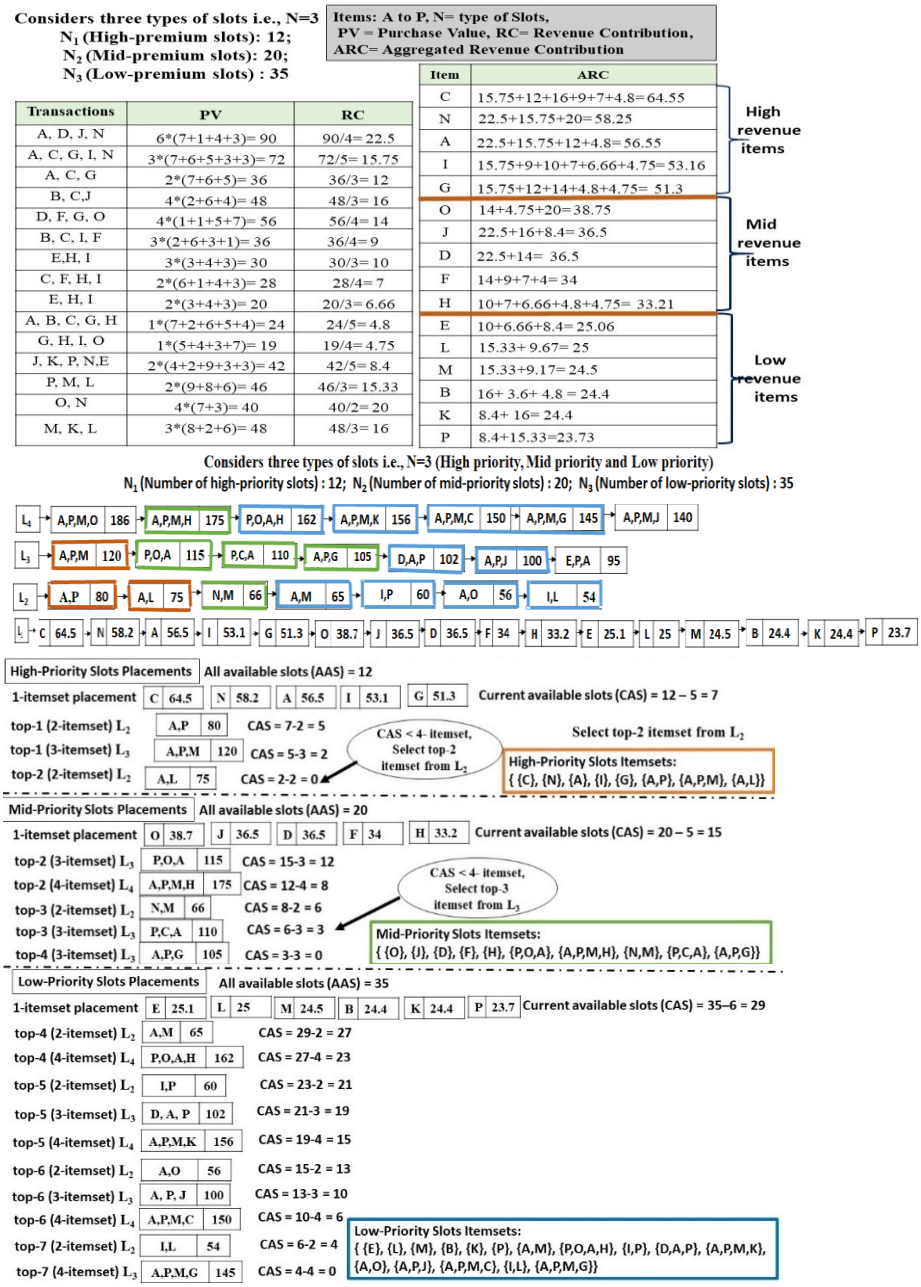


Fig. 3. Illustrative example of PRIP Scheme

Figure 3 indicates transactions with their respective values of RC and ARC. For the sake of clarity, this example considers three types of slots i.e., N=3 (high-premiumness, mid-premiumness and low-premiumness slot types) and the total number of slots corresponding to these slot types are 12, 20 and 35 respectively. Observe how the items are sorted in descending order of ARC. Using the formula in Line 8 of Algorithm 1, items {C, N, A, I, G} are selected as high revenue items, {O, J, D, F, H} are selected as mid-revenue items and the remaining {E, L, M, B, K, P} are the low-revenue items.

High-revenue items are mapped to high-premiumness slots, mid-revenue items are mapped to mid-premiumness slots and low-revenue items are mapped to low-premiumness slots. After placement of 1-itemsets, the remaining slots of high-premiumness = 12-5 i.e., 7, mid-premiumness = 20-5 i.e., 15 and low-premiumness = 35-6 i.e., 29.

To fill the remaining slot instances in each slot type, we extract itemsets from the kUI index. Starting from the highest-premiumness slot, we keep filling up the highest-premiumness slots as follows. We extract top-1 itemset from level 2 of the index i.e., itemset {A, P} and place it. Then we compute the number of remaining slots as (7-2) i.e., 5. Then, we extract the top-1 itemset from level 3 i.e., {A, P, M} and place it. Next, we update the remaining available slots as (5-3) i.e., 2. To fill the remaining 2 slots, itemsets from the next level of the kUI index cannot be extracted because remaining slots are less than itemset size at the current level i.e., 2 < level 4-itemset. Hence, we extract the top-2 itemset from level 2 of the kUI index i.e., {A, L} and place the itemset in the remaining slots. Now, there are no high-premiumness slots left.

Next, the mid-premiumness slot are filled in a similar manner. We extract top-2 itemset from level 2 of the index i.e., itemset {A, L}, but the itemset is already placed in the high-premiumness slot. Hence, we move to the next level i.e., L=3 and extract top-2 itemset from level 3 i.e., {P, O, A}. Next, we extract the top-2 itemset from level 4 i.e., {A, P, M, H} and we have now reached the highest level of the kUI index and still, there are 8 remaining available slots. Now, we extract the top-3 itemset from level 2 i.e., {N, M} and top-3 itemset from level 3 i.e., {P, C, A} and place it. Next, we update remaining available slots as (6-3) i.e., 3. Now, the remaining slots are less than the itemset size of the next level i.e. 3 < level 4-itemset. Hence, we extract the top-4 itemset from level 3 i.e., {A, P, G} and place it. Now, there are no slots left in the mid-premiumness slot type. Similarly, the low-premiumness slots are filled by extracting the next top-revenue itemset from each level until we exhaust all of the available slots.

5 PERFORMANCE EVALUATION

This section reports the performance evaluation. We performed our implementation and experiments on a 64-bit Core i5 processor running Windows 10 with 8 GB memory.

We conducted the experiments using a synthetic dataset as well as a real dataset. The real dataset is *Retail* dataset, which we obtained from the SPMF open-source data mining library [20]. It is an anonymous retail market basket data from an anonymous Belgian retail store. The dataset has 16,470 items and the number of transactions in the dataset is 88,162. We also generated a synthetic dataset, designated as T10I6N15K|D|2,000K, using the IBM data generator. The parameters of the synthetic

dataset are as follows: T (the average size of the transactions) is 10; I (the average size of potential maximal itemsets) is 6; N (the number of distinct items) is 15K; |D| (the total number of transactions) is 2,000K. The dataset has 15,000 items. For the sake of brevity, we shall henceforth refer to this data as the **synthetic dataset**.

The Retail dataset and the synthetic dataset do not provide utility (price) values. Hence, for each of these datasets, we generated the price of the items in the range [0.01, 1.0] as follows. We divided the price range into six approximately equal-width categories i.e., [0.01- 0.16], [0.17- 0.33], [0.34-0.50], [0.51-0.67], [0.68-0.84] and [0.85-1]. For generating the price for each item, we randomly select one of these categories and generate a random number within the price range of that category.

For our experiments, we divided the transactions dataset into two parts, namely *training set* (containing 70% of the transactions) and *test set* (containing the remaining 30% of the transactions). For our PRIP scheme, we placed the itemsets based on the kUI index built from the training set. We evaluate the performance on the test set.

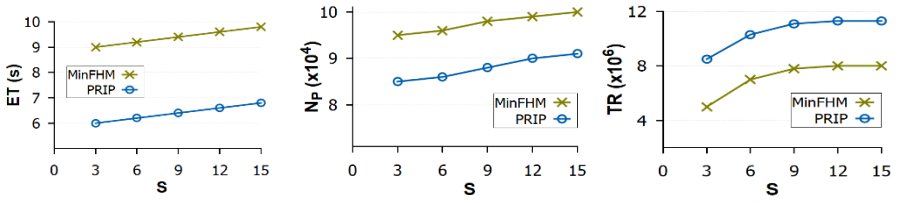
Our performance metrics are execution time (ET), the number of patterns generated (N_P) and Total revenue (TR). ET is the execution time for the determination of the placement of itemsets in the slots for the training set. N_P is the number of patterns (itemsets) that a given scheme needs to examine for the itemset placement for the training set. TR, which is the total retailer revenue for the test set, is computed as follows. We iterate through each transaction *t* in the test set and add to TR only the prices of the items (in *t*), which had already been placed in the retail slots during the training phase.

To quantify the notion of premiumness, we assign probability of sales to the different slot types in the range of [0.01, 1]. Given N slot types, we divide the [0.01, 1] range into N sub-ranges and assign these sub-ranges to the slot types to quantify the respective probability of sales for items/itemsets placed in these slot types. Consider N = 3. Hence, the corresponding probabilities for the three different slot types in increasing order of premiumness would be in the following sub-ranges: [0.01 – 0.34], [0.35 – 0.68] and [0.69 – 1.0]. These ranges are *approximately* equally distributed.

Table 1. Parameters of Performance Evaluation

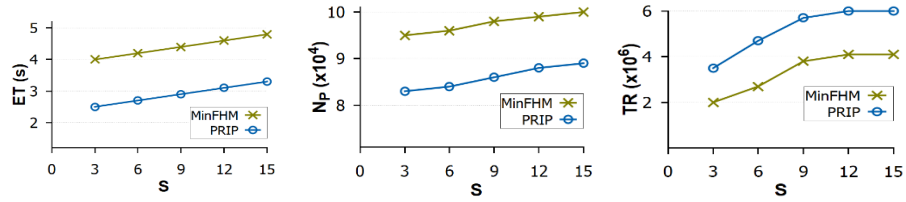
Parameter	Default	Variations
Total number of slots (10^4) (T_S)	10	2, 4, 6, 8
Number of slot type (S)	6	3, 9, 12,15
Revenue threshold (α)	30%	
Zipf factor (Z_{NS})	0.7	0.1, 0.3, 0.5, 0.9

The number λ of itemsets at each level of the kUI index was 5000, and the kUI index had 10 levels. Table 1 summarizes the parameters of the performance study. In Table 1, the Zipf factor (Z_{NS}) represents skew in the number of slot instances across slot types. Z_{NS} is defined in the range [0.1, 0.9], where 0.1 indicates a relatively uniform distribution, while 0.9 indicates a highly skewed distribution. When Z_{NS}=0.1, there would be approximately an equal number of slots corresponding to each slot type. In contrast, when Z_{NS}=0.9, there would be a disproportionately higher number of low-premiumness slots and a relatively fewer number of high-premiumness slots.



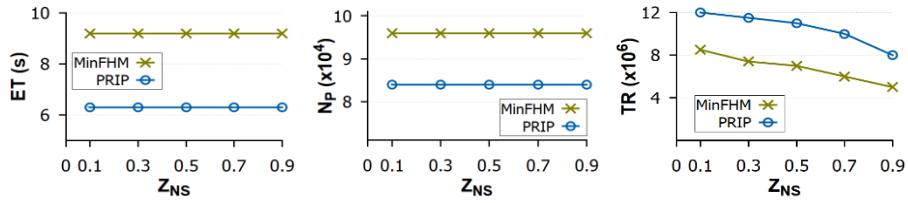
(a) Execution Time (b) Number of Patterns (c) Total Revenue

Fig. 4. Effect of variations in the number of slot types (Synthetic dataset)



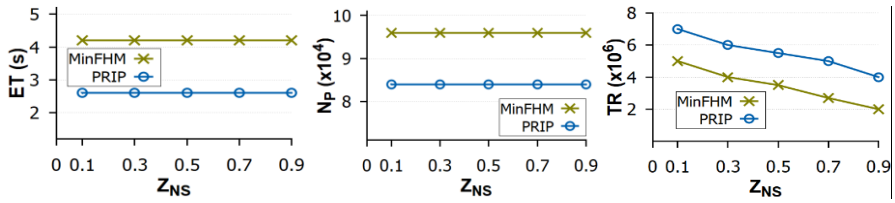
(a) Execution Time (b) Number of Patterns (c) Total Revenue

Fig. 5. Effect of variations in the number of slot types (Retail dataset)



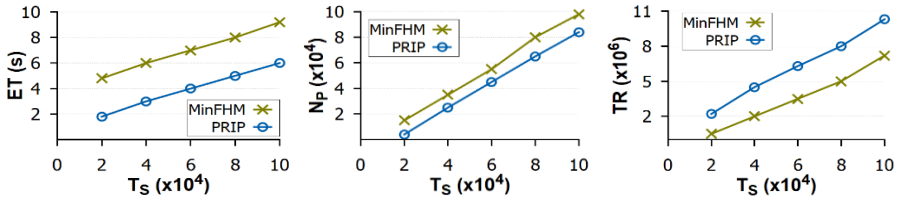
(a) Execution Time (b) Number of Patterns (c) Total Revenue

Fig. 6. Effect of variations in skew across slot types (Synthetic dataset)



(a) Execution Time (b) Number of Patterns (c) Total Revenue

Fig. 7. Effect of variations in skew across slot types (Retail dataset)



(a) Execution Time (b) Number of Patterns (c) Total Revenue

Fig. 8. Effect of variations in the total number of slots (Synthetic dataset)

As reference, we adapted the recent MinFHM scheme [13]. Given a transactional database with utility information and a minimum utility threshold as input, **MinFHM** outputs a set of **minimal high-utility itemsets** having utility no less than that of ***min_utility***. By scanning the training set, the algorithm creates a utility-list structure for each item and then uses this structure to determine upper-bounds on the utility of extensions of each itemset. We adapted the MinFHM scheme as follows. First, we use the MinFHM scheme to generate all the itemsets across different itemset sizes (k). From these extracted itemsets from the training set, we randomly selected itemsets for placement at the different types of available slots until we exhausted all of the available slots. We also ensured that each item is represented at least once on any type of slot. We shall henceforth refer to this scheme as **MinFHM**.

Effect of variations in the number of slot types: Figures 4 and 5 depict the effect of variations in the number S of slot types for the synthetic and Retail datasets respectively. The results in Figure 4 indicate that as S increases, both the schemes incur more N_P because the number of patterns (itemsets) to be examined increases. This also contributes to increase in ET. PRIP outperforms MinFHM in terms of ET and N_P because PRIP considers only the top- λ high-revenue itemsets of different sizes (by exploiting the kUI index). On the other hand, MinFHM examines all the patterns that exceed a pre-specified minimum utility threshold. Hence, the number of patterns to be examined in case of MinFHM is greater than that of PRIP.

The results in Figure 4(c) indicate that PRIP provides higher TR than MinFHM. This occurs because as S increases, the effect of slot premiumness considered by PRIP becomes more dominant. In contrast, MinFHM does not consider slot premiumness. Thus, while PRIP places high-revenue itemsets to slots with high premiumness, MinFHM may place high-revenue itemsets in slots with low premiumness because MinFHM randomly places itemsets on slots irrespective of the slot premiumness. However, beyond S=9, both schemes exhibit a saturation effect because TR is essentially upper-limited by the total revenue value associated with the typical user purchase transactions.

The results in Figure 5 follow similar trends as those of Figure 4; the difference in the actual values of the performance metrics arises due to the respective dataset sizes.

Effect of variations in skew across slot types: Figures 6 and 7 depict the effect of variations of skew in the number of slots across the slot types for the synthetic and Retail datasets. As discussed previously, such skew is modelled by Z_{NS}. The results in Figure 6 indicate that ET and N_P remain comparable for both schemes as Z_{NS} increases. This occurs because the total number of slots to be filled remain the same, irrespective of the value of Z_{NS}. Hence, the time required for extraction and placement of itemsets remains comparable as Z_{NS} changes. PRIP outperforms MinFHM in terms of ET and N_P due to the reasons explained for Figure 4. The results in Figure 6(c) indicate that both schemes show decrease in the value of TR with increase in Z_{NS}. This occurs because as the value of Z_{NS} increases, the skew increases. Hence, the number of lower premiumness slots become more w.r.t. the higher-premium slots. Due to a decreased number of slots with higher premiumness, TR decreases.

The results in Figure 7 follow similar trends as those of Figure 6; the difference in the actual values of the performance metrics arises due to the respective dataset sizes.

Effect of variations in the total number of slots: Figure 8 depicts the effect of variations in the total number T_S of slots for the synthetic dataset. The results in Figures 8(a) and 8(b) indicate that ET and N_P increase for both schemes with increase in T_S. As T_S increases, more slots need to be filled, thereby necessitating more patterns to be examined. PRIP outperforms MinFHM in terms of ET and N_P due to the reasons explained earlier for Figure 4. The results in Figure 8(c) indicate that TR increases for both the schemes as T_S increases. As T_S increases, more itemsets are used to fill up an increased number of slots, thereby increasing TR. PRIP provides higher TR than that of MinFHM due to the reasons explained earlier for Figures 4(c). We also performed this experiment with the Retail dataset. The results exhibited similar trends. Due to space constraint, we do not present those results in this paper.

6 CONCLUSION

Given that retail slots typically vary in terms of premiumness, we have proposed a scheme for mapping and placing itemsets to slots based on itemset revenue and slot premiumness with the objective of maximizing the revenue for the retailer. Based on the knowledge extracted from users' purchase patterns, we have proposed a framework and a scheme for determining the placement of the itemsets in different types of slots with varied premiumness such that each item is placed at least once in any of the slots. Our framework is capable of *efficiently* identifying itemsets from a transactional database and placing these itemsets by mapping itemsets with different revenue to slots with varied premiumness for maximizing retailer revenue. Our performance evaluation using both synthetic and real datasets indicates that the proposed scheme is indeed effective in improving the total retailer revenue by up to 45% w.r.t. a recent existing scheme. Given that pattern mining is an important and active research area, the proposed framework is versatile and generic in the sense that different kinds of existing pattern mining approaches can also be investigated for further improving the revenue of the retailer.

References

1. Hansen, P., Heinsbroek, H.: Product selection and space allocation in supermarkets. European Journal of Operational Research 3(6), 474-84 (1979)
2. Yang, M.H., Chen, W.C.: A study on shelf space allocation and management. International Journal of Production Economics 60, 309-317 (1999)
3. Yang, M.H.: An efficient algorithm to allocate shelf space. European Journal of Operational Research 131(1), 107-118 (2001)
4. Chen, M.C., Lin, C.P.: A data mining approach to product assortment and shelf space allocation. Expert Systems with Applications 32(4), 976-986 (2007)
5. Chen, Y.L., Chen, J.M., Tung, C.W.: A data mining approach for retail knowledge discovery with consideration of the effect of shelf-space adjacency on sales. Decision Support Systems 42(3), 1503-1520 (2006)
6. World's largest retail store, https://www.thebalance.com/largest-retail-stores-2892923
7. Hart, C.: The retail accordion and assortment strategies: an exploratory study. The International Review of Retail, Distribution and Consumer Research 9(2), 111-126 (1999)

8. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. Proc. VLDB, Vol. 1215, pp. 487-499 (1994)
9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. Proc. ACM SIGMOD, Vol. 29, pp. 1-12 (2000)
10. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. Proc. ICDT, pp. 398-416 (1999)
11. Liu, Y., Liao, W.K., Choudhary, A.: A fast high utility itemsets mining algorithm. Proc. of 1st International workshop on Utility-based data mining, pp. 90-99 (2005)
12. Fournier-Viger, P., Wu, C.W., Tseng, V.S.: Novel concise representations of high utility itemsets using generator patterns, Proc. ADMA, pp. 30-43 (2014)
13. Fournier-Viger, P., Lin, J.C.W., Wu, C.W., Tseng, V.S., Faghihi, U.: Mining Minimal High-Utility Itemsets, Proc. DEXA, pp. 88-101 (2016)
14. Zida, S., Fournier-Viger, P., Lin, J.C.W., Wu, C.W., Tseng, V.S.: EFIM: a highly efficient algorithm for high-utility itemset mining. Proc. MICAI, pp. 530-546 (2015)
15. Fournier-Viger, P., Zida, S., Lin, J.C.W., Wu, C.W., Tseng, V.S.: EFIM-Closed: Fast and Memory Efficient Discovery of Closed High-Utility Itemsets, Proc. MICAI, pp. 199-213 (2016)
16. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation, Proc. CIKM, pp. 55-64 (2012)
17. Tseng, V.S., Wu, C.W., Fournier-Viger, P., Philip, S.Y.: Efficient algorithms for mining the concise and lossless representation of high utility itemsets, IEEE TKDE, pp. 726-739 (2015)
18. Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S.: UP-Growth: an efficient algorithm for high utility itemset mining, Proc. ACM SIGKDD, pp. 253-262 (2010)
19. Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets, Proc. ICDM, pp. 19-26 (2003)
20. http://www.philippe-fournier-viger.com/spmf/dataset
21. Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning, Proc. ISMIS, pp. 83-92 (2014)
22. Chaudhary, P., Mondal, A., Reddy, P.K.: A Flexible and Efficient Indexing Scheme for Placement of Top-Utility Itemsets for Different Slot Sizes, Proc. BDA, pp. 257-277 (2017)
23. Chaudhary, P., Mondal, A., Reddy, P.K.: A Diversification-Aware Itemset Placement Framework for Long-Term Sustainability of Retail Businesses, Proc. DEXA, pp. 103-118 (2018)
24. Corsten, D., Gruen, T.: Desperately seeking shelf availability: an examination of the extent, the causes, and the efforts to address retail out-of-stocks. International Journal of Retail & Distribution Management, pp. 605-617 (2003)
25. Fournier-Viger, P., Zhang, Y., Lin, J.C.W., Fujita, H., Koh, Y.S.: Mining local high utility itemsets, Proc. DEXA, pp. 450-460 (2018)
26. Jaysawal, B.P., Huang, J.W.: DMHUPS: Discovering Multiple High Utility Patterns Simultaneously. International Journal of Knowledge and Information Systems 59(2), 337-359 (2019)
27. Flamand, T., Ghoniem, A., Haouari, M., Maddah, B.: Integrated assortment planning and store-wide shelf space allocation: An optimization-based approach. International Journal of Management Science: Omega (81), pp. 134-149 (2018)