Cross-Selling Optimization for Customized Promotion

Nan Li* Yinghui Yang† Xifeng Yan*
*{nanli, xyan}@cs.ucsb.edu
†yiyang@ucdavis.edu

Abstract

The profit of a retail product not only comes from its own sales, but also comes from its influence on the sales of other products. How to promote the right products to the right customers becomes one of the key issues in marketing. In this paper, we propose a new formulation of promotion value by considering cross-selling effects within selected products and customers, which were largely ignored by existing work. We investigate the problem of customized promotion, which identifies promotional products and customers so that the promotion effect can be maximized. This problem can be decomposed into two subproblems: product selection and customer selection. The baseline methods entail an exhaustive traversal of all possible product and customer combinations, which is computationally intractable. As an alternative, we propose greedy and randomized algorithms to produce approximation solutions in an efficient manner. Experiments on both synthetic and real-world supermarket transaction data demonstrate the effectiveness and efficiency of the proposed algorithms.

Keywords: Product Promotion, Customer Selection, Transaction Mining, Optimization

1 Introduction.

With a majority of retailers keeping track of their sales records, they are becoming increasingly powerful at deciding what to promote and whom to promote to in order to maximize profit. Retailers often do sales to get rid of slow-moving merchandize, introduce new products and sell seasonal products. With the sales transaction data, it is becoming increasingly feasible and important to tailor promotions and sales according to the knowledge learned from the existing transactions. Through these transactions, customers' purchasing behavior can be learned, and promotions can be designed and customized according to such behavior in order to bring maximum profit to marketing campaigns.

The profit of a product not only comes from its own sales, but also comes from its influence on the sales of other items. How to properly select the right products to promote to achieve the maximum benefit is one of the core questions retailers often face. In parallel, the second important problem in promotion planning is how to select the right customers to promote the products to. In this paper, we study the two core problems of promotion planning - product selection and customer selection.

One of the most popular data mining techniques used for analyzing customer shopping basket data is the association rule discovery technique [1]. This technique discovers associations among products, from which we can infer whether a product or a set of products can positively affect the sales of other products. We propose an approach which can directly utilize the results of the association rule analysis. It takes the rules as input and generates a list of products on promotion. For each product, our approach further identifies a subset of customers for whom the promotional effect of this product will be maximized.

Our contributions. In this work, we quantitively formulate and address two practical optimization problems in product marketing - promotional product selection and customer selection in order to maximize promotion profits. A novel formulation of promotion value for products is derived to measure the promotional profitability of a certain product set upon the other products. It is further used as a metric to select products and target customers. We propose fast and scalable approximation algorithms to conduct product and customer selection because exhaustive traversal of all product and customer combinations yields exponential time complexity. The experimental evaluation on both synthetic data and real supermarket data proves the effectiveness, efficiency and scalability of the proposed methods.

The remainder of the paper is organized as follows. We discuss related work in Section 2. Section 3 formulates the product selection and customer selection problems. Promotion value function is defined in Section 4. Section 5 develops the methods used for product selection and Section 6 presents the methods for customer selection. Section 7 reports the experimental results and Section 8 concludes with discussion on future work.

2 Related Works.

In this section, we review some related works within the current literature, which can be categorized into four parts.

Catalog segmentation. An extensively studied prob-

*Computer Science Department, University of California, Santa Barbara.

†Graduate School of Management, University of California, Davis.

lem in product selection is catalog segmentation, which is formalized by Kleinberg et al. in [13] as the process of segmenting the product set according to the customers and sending different customers different catalogs for promotion. It is concluded in [13] that most of the discussed segmentation problems are NP-complete, which leads to subsequent activities of investigating approximation algorithms and their probabilistic bounds [9], [14].

Our study addresses a different problem, where the objective is to maximize a predefined promotion value. In order to measure promotion value, cross-selling effect [12], [22] between products is used to penalize product set containing associated products. Yang et al. in [22] propose heuristic tree search to address similar problems. As an improvement, our study extends [22] by refining the definition of promotion value functions via including products-dependent transaction-rule association and other important factors. More efficient approximation product selection algorithms are designed. Another extension to [22] is approximation algorithms for customer selection.

Product assortment. The core decision in the product assortment problem is to decide what items a store should carry, given limited retail space [5], [18], [19], [20]. In [3], [4], [5], the assortment first needs to be consistent with the store's image by including some basic products, and the added products should be selected to maximize cross-sales potential with the basic products. In [19] and [20], the value of the products that are not carried by the store will be considered as a loss to the store. Their goal is to minimize such loss. Our focus is to select products with the highest promotion value. The objective function of the product assortment problem is very different from that of the product selection problem and the techniques used for product assortment cannot be directly applied to the product selection problem. Moreover, we study the related customer selection problem which is not relevant to the product assortment problem.

Customer profiling. Customer profiling is an effective tool to determine what type of customers are most likely to purchase a certain type of products, so that more efficient marketing plans are designed. Association between customer purchase and customer demographics has been investigated for customer profiling. Chou et al. [8] utilize clustering techniques to identify prospective customers under various scenarios where different data sets are available. Wu et al. [21] use user demographic indicators to formulate the customer lifetime value (CLV), which is used as a metric for customer selection. Indicators other than demographics have drawn attention too [2]. In our study, we mine association rules from customer transactions and select customers based on promotion profitability analysis.

Association rule mining. In data mining, association rule mining refers to discovering interesting relations between entities in large databases. Introduced by Agrawal et at. [1], it has been extensively studied [10], [15] and used in various applications [1], [6], [16]. Mining association rules in market transaction data has been a well-studied topic in data mining. Agrawal et al. introduce an efficient algorithm to generate all significant association rules between products in the database [1]. Brin et al. further develop the notion of mining rules that identify correlations [6]. In our paper, we use FP-Growth based association rule mining [17] algorithm to identify product pairs from transaction history for promotion value modeling.

3 Problem Formulation.

3.1 Product Selection. In market planning, product selection refers to identifying an optimal set of products that maximizes the promotional influence on other products. A predefined measurement is designed to evaluate how much promotion value a product set yields based on transaction history. We denote each set of products purchased by a certain customer on a certain date as a transaction. We consider $T = \{t_1, t_2, \ldots, t_v\}$ as a set of historical transactions. The total product set is $P = \{p_1, p_2, \ldots, p_w\}$. Each $t_i (i = 1, 2, \ldots, v)$ contains a subset of P. In our study, we define the product selection problem as following.

Given a set of historical transactions T, a total product set P and a promotion product set size n, promotional product selection is to identify a subset of products γ^*, so that

$$\gamma^* = \mathrm{argmax}_{\gamma | \gamma \subseteq P, |\gamma| = n} \{f(\gamma)\},$$

where $f(\gamma)$ is the promotion value function of γ.

3.2 Customer Selection. Developing a better understanding of customer purchase behavior allows for the identification of prospective customers. In our study, we define the customer selection problem as following.

Given a product set on promotion γ^*, a total customer set C and a size k, customer selection aims to choose the top k customers for each promotional product in γ^*, so that the total promotion value is maximized. The selected k customers for each product a is referred to as a's prospective customer set, $C_k(a)$. For each selected customer μ, let γ^*_μ be a subset of γ^* that μ is selected for. Customer selection problem identifies $\theta^* = \cup_{a \in \gamma^*} C_k(a)$, so that

$$\theta^* = \mathrm{argmax}_\theta \{\sum_{\mu \in \theta} f(\gamma^*_\mu)\},$$

where f is the aforementioned promotion value function.

A brute-force search over all possible product and customer combinations yields exponential time complexity. In our study, we propose efficient approximation algorithms for both product and customer selection. On small-scale synthetic data, the approximation algorithms are able to generate near-optimal product sets. Table 1 lists the notations we use throughout the paper.

Symbol	Description
$a, b, c, \ldots$	Products.
$\alpha, \beta, \gamma, \ldots$	Product sets.
$t_1, t_2, \ldots$	Transactions.
$r_1, r_2, \ldots$	Rules. $r : \alpha \to b$ denotes a generic rule.
P,T,C,R	The total product, transaction, customer and rule sets.
Δ	Mutual promotion value.
$\pi(\gamma)$,$\pi_\mu(a)$	Promotion value of γ, promotion value of product a for customer μ.
μ, θ	A customer, selected customer set.
n,k	Promotion product set size, # of selected customers per product.
$T(\gamma)$, $R(\gamma)$	The associated transaction and rule set of γ.
T_μ	μ's transactions.
$T_r(\gamma)$	Transactions in $T(\gamma)$ that support rule r.
$C_k(a)$	a's k selected(prospective) customers.

Table 1: Denotation Description

4 Promotion Value Function.

In order to calculate the promotion value of product set $\alpha \subseteq P$, a method to compute the promotion value of α over another product set β is desired. The total promotion value of α is the aggregated value of all its promotion values over other product sets. In recent literature, a widely-applied concept is lift [7], [11], [12], [22], which measures the mutual attractions between a pair of products. [12] illustrates this idea via an example: an "air conditioning unit" and "air conditioning unit accessory" might be very rarely bought with other products, but might be bought together frequently. In this paper, we introduce a scoring mechanism that is similar to lift and that indicates how much promotion effect can be generated by advertising one product set over another. We name it as mutual promotion value, which is formulated as follows.

DEFINITION 4.1. (Mutual Promotion Value) *The mutual promotion value $\Delta_T(\alpha \to \beta)$, of a set of products α over another set of products β, based on a set of transactions T, is formulated as*

$$\Delta_T(\alpha \to \beta) = \Pr(\beta|\alpha) - \Pr(\beta), \tag{4.1}$$

where $\Pr(\beta|\alpha)$ is the probability of occurrence of β in T conditionally on α and $\Pr(\beta)$ is the probability of occurrence of β in T.

A high Δ value indicates a high likelihood boost of purchasing β induced by purchasing α. The promotion value of α is an aggregated value of its Δ values over all other possible product sets.

4.1 Association Rule Mining.

If there is a rule $r : \alpha \to \beta$ found in T, with α as its antecedents and β as its consequents, $\Pr(\beta|\alpha)$ is known as the confidence of this rule and $\Pr(\beta)$ is the support of the consequent.

We compute the promotion value of a product set γ as following. a), Traverse through T to generate association rule set R. b), Find the rules that are associated with γ from R. c), Use these rules to form product set pairs. d), Compute the Δ value between α and β for each rule $r : \alpha \to \beta$, as shown in (4.1). e), Each rule's contribution value is computed based on its Δ value. f), Aggregate all rules' contribution values as the promotion value of γ.

We use the FP-Growth based method [10], [17] to mine rules from T. We ignore rules that have multiple products as consequents, simply because these rules can be converted to a set of rules with only one product as the consequent. Therefore hereinafter we use $r : \alpha \to b$ to denote a rule.

4.2 Promotion Value Function.

In this section, we elaborate the definition of our promotion value function. A novel transaction-rule association concept is designed to differentiate each rule's contribution for a given product set. A product set promotion value function is then proposed which agrees with human intuition.

Besides mutual promotion value Δ, there are additional aspects that are critical to a rule's contribution. For example, Δ does not indicate how many transactions actually exhibit such rule. It is also sensible to consider the affinity between an association rule and the product set on promotion, since the significance of each rule is contingent on the current product set. For example, rule $r : (a, b) \to c$ is not significant to product set $\{d, e\}$, however, it is significant to product set $\{a, b, c\}$. Before scrutinizing these factors, some preliminaries regarding associative relations are given in Definition 4.2.

DEFINITION 4.2. (Associative Relations) *Given a product set γ, we define a set of associated transactions, $T(\gamma)$, and a set of associated rules, $R(\gamma)$, as following:*

- *$T(\gamma)$: For each $t \in T(\gamma)$, t contains at least one product in γ.*
- *$R(\gamma)$: For each $r : \alpha \to b \in R(\gamma)$, α contains at least one product in γ, and b is not in γ.*

The reason to exclude rules whose consequent is in γ is because such rules indicate crossing-selling among products. Consider a product set $\alpha = \{a, b, c, d\}$ and a rule $r : (a, b) \to c$. r is not a good candidate rule since it implies if $\{a, b\}$ are promoted, it is not necessary to promote c anymore. Because consumers are likely to purchase c anyways. Therefore r should not contribute to the promotion value of α.

The reasons for choosing transactions and rules that contain at least one product in γ, instead of all the products in γ, is because the subsets of a product set also have critical impact during promotion. Consider product sets $\alpha = \{a, b, c\}$, $\beta = \{d, e, f\}$ and a rule $r : (a, b) \rightarrow g$. It conforms with human intuition that r renders α more in favor because promoting α encourages the purchase of g due to its subset $\{a, b\}$.

Each rule r in $R(\gamma)$ is further associated with a set of supporting transactions from $T(\gamma)$, denoted as $T_r(\gamma)$. $T_r(\gamma)$ measures how supportive $T(\gamma)$ is towards r, or how prevailing r is among transactions in $T(\gamma)$. We propose a novel concept, products-dependent transaction-rule association.

4.2.1 Products-Dependent Transaction-Rule Association. In [22], $T_r(\gamma)$ is defined as transactions containing all the products in r. Such mechanism is undesirable due to two reasons. First, it results in duplication. Each transaction supports multiple rules. When the contributions of those rules are aggregated, some transactions are counted multiple times. Secondly, this leads to a constant set of supporting transactions for each rule regardless of the current product set.

Adriano et al. [16] incorporate associative classification with a novel technique to associate each testing instance with one most competent rule-evaluating metric. We adopt a similar paradigm to associate each transaction, based on the current product set, with one most competent rule, thus each rule's contribution is computed in a products-dependent manner.

Suppose the promotion set is $\gamma = \{b, d, f\}$, the set of associated transactions is $T(\gamma) = \{t_1 = \{a, b, c, d, e\}, t_2 = \{a, b, c, d\}, t_3 = \{a, b, e\}, t_4 = \{d, e, f, g\}\}$ and the set of associated rules is $R(\gamma) = \{r_1 : b \rightarrow a, r_2 : (d, e) \rightarrow g, r_3 : (a, b) \rightarrow c\}$. If we associate each rule with transactions that contain all of its products, then $T_{r_1}(\gamma) = \{t_1, t_2, t_3\}$, $T_{r_2}(\gamma) = \{t_4\}$ and $T_{r_3}(\gamma) = \{t_1, t_2\}$. Such approach results in the aforementioned two problems. Instead, we designate only one rule to each transaction. This ensures that each rule's associated transaction set is disjoint from each other.

There are two approaches to implement this paradigm. The first is based on optimistic intuition, which designates each transaction the maximum-size rule. The size of a rule $r : \alpha \rightarrow b$ is defined as $|\alpha| + 1$. In the above example, t_1 will be designated r_3, t_2 will be designated r_3, t_3 will be designated r_1 and t_4 will be designated r_2. Thus $T_{r_1}(\gamma) = \{t_3\}$, $T_{r_2}(\gamma) = \{t_4\}$ and $T_{r_3}(\gamma) = \{t_1, t_2\}$. The second is based on pessimistic intuition, which oppositely designates each transaction the minimum-size rule. Therefore, t_1 will be designated r_1, t_2 will be designated r_1, t_3 will be designated r_1 and t_4 will be designated r_2. Thus $T_{r_1}(\gamma) = \{t_1, t_2, t_3\}$, $T_{r_2}(\gamma) = \{t_4\}$ and $T_{r_3}(\gamma) = \emptyset$.

Pessimistic approach is more conservative since it regards rules with more products in their antecedents less generic. Optimistic approach however, considers rules with more products more competent since they occupy a higher proportion of the transaction. In our study, we choose the optimistic approach since we consider maximum-size rules are more informative. We also consider the aspect that rules with more products usually have higher confidence.

Another noteworthy issue is to choose between multiple maximum-size rules for the transaction. In our study, we choose the one that has the maximum mutual promotion value, Δ.

4.2.2 Product Set Promotion Value. In this section, we propose a function to score the promotion value of a product set. Given a product set on promotion, γ, and an associated rule $r : \alpha \rightarrow b$, we summarize a list of important factors to be incorporated in the function.

Mutual promotion value Δ. The mutual promotion value measures the probability enhancement of purchasing b induced by purchasing α.

Supportiveness of $T(\gamma)$. It denotes how supportive the transactions in $T(\gamma)$ are towards r. The proposed transaction-rule association process is utilized to generate $T_r(\gamma)$ for each r. The number of transactions in $T_r(\gamma)$ is used as the indicator of such supportiveness.

Affinity between r **and** γ. The significance of $r : \alpha \rightarrow b$ depends on the product set, γ. We model this by the number of mutual products contained in α and γ.

Product profit. The ultimate goal is to maximize the sales profit of the products purchased due to the promotion, namely the consequents of the rules. Let $p(b)$ denote b's profit. In this study, we assume a uniform profit for all the products in P. However, the proposed function can be extended to product set with various profits.

We propose the promotion value function in Definition 4.3.

DEFINITION 4.3. (Promotion Value) *The promotion value of product set γ over other products in P, denoted as $\pi(\gamma)$, is computed as*

$$\begin{aligned} \pi(\gamma) = \textstyle\sum_{r:\alpha\rightarrow b\in R(\gamma)} [\tfrac{|\alpha\cap\gamma|}{|\alpha|} * \tfrac{|T_r(\gamma)|}{|T(\gamma)|} \\ *p(b) * \Delta_{T(\gamma)}(r)], \end{aligned} \tag{4.2}$$

where each term in the sum is the contribution of a rule. The contribution of rule r is the multiplication of r's affinity to the current product set γ, the supportiveness of $T(\gamma)$ to r, r's consequent's profit and the mutual promotion value between r's antecedents and consequent.

Algorithm 1 demonstrates the computation in more details. We also provide a running example shown in Table 2. Suppose we have a transaction set T_0 and a product set γ_0, see Table 2-(a) and Table 2-(b). If we apply FP-Growth based rule mining on T_0 and set $minOccur = 3$ and

Algorithm 1 Product Set Promotion Value Computation

```
PromotionValue($\gamma$, $T$, $R$)
Input: The selected product set $\gamma$, the total transaction set $T$, the total rule set $R$
Output: $\pi(\gamma)$
1: $\pi(\gamma) \Leftarrow 0, T(\gamma) \Leftarrow \emptyset, R(\gamma) \Leftarrow \emptyset$
2: traverse $T$ to initialize $T(\gamma)$
3: traverse $R$ to initialize $R(\gamma)$
4: designate each transaction in $T(\gamma)$ one rule in $R(\gamma)$
5: aggregate the contribution of all the rules in $R(\gamma)$ according to Definition 4.3, to $\pi(\gamma)$
6: return $\pi(\gamma)$
```

$minConf = 0.8$, where $minOccur$ denotes the minimum number of occurrences of the product and $minConf$ denotes the minimum confidence of the rule, we can obtain the set of rules R_0, in Table 2-(c). Example 1 demonstrates how to

(a) Transaction Set T_0

ID	Customer	Products
t_1	*Ann*	$\{a, b, c, d\}$
t_2	*Ann*	$\{a\}$
t_3	*Jon*	$\{b, c, d, e, f, g\}$
t_4	*Jon*	$\{c, d, e, f\}$
t_5	*Dan*	$\{b, f, g, h\}$
t_6	*Dan*	$\{b, c, d, f, h\}$
t_7	*Dan*	$\{a, b, c\}$

(b) Product Set γ_0

Product	# of Occur.
a	3
b	5
c	5
d	4
e	2
f	4
g	2
h	2

(c) Association Rule Set R_0

ID	Rule	Confidence	ID	Rule	Confidence
r_1	$d \rightarrow c$	1.0	r_2	$(b, d) \rightarrow c$	1.0
r_3	$(c, f) \rightarrow d$	1.0	r_4	$(d, f) \rightarrow c$	1.0
r_5	$b \rightarrow c$	0.8	r_6	$c \rightarrow b$	0.8
r_7	$c \rightarrow d$	0.8			

Table 2: Running Example with Sample Sets

compute the promotion value of a product set on this running example.

EXAMPLE 1. (Promotion Value Example) *Suppose* $\gamma = \{b, c\}$*, and we want to compute* $\pi(\gamma)$*. According to Definition 4.2, we can derive* $T(\gamma) = \{t_1, t_3, t_4, t_5, t_6, t_7\}$ *and* $R(\gamma) = \{r_3, r_7\}$[1]*. The transaction-rule association associates* r_3 *with* $\{t_3, t_4, t_6\}$ *and* r_7 *with* $\{t_1\}$*. Thus* $|T_{r_3}(\gamma)| = 3$ *and* $|T_{r_7}(\gamma)| = 1$*. The confidence of* r_3 *in* $T(\gamma)$ *is 1.0 and the support of the consequent of* r_3 *in* $T(\gamma)$ *is 0.67. The antecedents of* r_3 *and* γ *have one product in common. Thus if we assume a uniform product profit 0.2, we can compute the contribution of* r_3 *as* $0.5 * 0.5 * 0.2 * (1.0 - 0.67) \approx 0.017$*. Likewise, we can compute the contribution of* r_7 *as* 0.004*. Eventually we have* $\pi(\gamma) = 0.017 + 0.004 \approx 0.021$*.*

Algorithm 1 entails a transaction-rule association process, therefore its run time complexity in the worse case is $O(|T| * |R|)$.

5 Product Selection.

Our approach first selects product set for promotion and then selects, for each product on promotion, a set of customers to promote this product to. Such sequence approach is first due to the complexity of selecting product set and customer set simultaneously. Secondly, there are scenarios where only products need to be selected, such as supermarkets putting on discounts on certain products. Unlike sending out coupons, such sales activity is targeted at all the customers.

Given a promotional product set size n, the product selection algorithms seek to select a subset of products (size n) from P that maximizes the π value. A brute-force search to generate all possible product combinations takes $O(C^n_{|P|} * |T| * |R|)$ run time. This makes it computationally intractable when $|P|$ or n is large.

In this paper, we propose alternative approximation algorithms to return a near-optimal product set within much shorter time. An intuitive approach is to choose the top n products with the highest π values[2]. We name this as naive product search (NPS). NPS fails to tackle the prevalent cross-selling effect [12], [22]. For example, if $\{a, b, c\}$ are the top three products in P with the highest π values, it does not necessarily mean that choosing these three will maximize the profit. The motivation to choose c is undermined if there is a rule $(a, b) \rightarrow c$ prevailing in the transaction set T. Because this indicates people are inclined to purchase c after they have purchased a and b, therefore there is not as much need to promote c anymore. In the next sections, we propose two other approximation algorithms that are better structured to tackle this issue.

5.1 Heuristic Tree Search.

A heuristic tree search (HTS) is designed based on the tree search algorithm in [22]. HTS copes with the cross-selling effect via maximizing the π value of an entire product set, instead of summing up the π value of each product. HTS greedily chooses the node

[1] r_2, r_5 and r_6 also contain products from γ in their antecedents. However, they are not considered since their consequent is contained in γ.

[2] π can be computed on both product sets and single products.

that currently maximizes the π value of its path among all the current leaf nodes to expand. HTS returns a leaf node in the tree, and the path leading from the root node to the leaf represents the selected products.

Algorithm 2 describes the HTS algorithm. Each node in the search tree corresponds to a product in P. The π value of each node is the π value of the products on the path from the root to this node. Example 2 shows applying HTS on the running example in Table 2.

```
Algorithm 2 Heuristic Tree Search
HTS($n$, $P$, $T$, $R$)
Input: The promotion product set size $n$, the total product set $P$, the total transaction set $T$, the total rule set $R$
Output: The selected product set $\gamma_{HTS}$
1: $\gamma_{HTS} \Leftarrow \emptyset$, create an empty root node, whose $\pi$ is 0
2: expand the root node to $|P|$ children
3: compute the $\pi$ value for each child and add all the children to $l_{expand}$
4: for each node in $l_{expand}$, compute its adjusted value by subtracting its $\pi$ value with the average $\pi$ value of its parent and its parent's siblings
5: pick the node in $l_{expand}$ that has the highest adjusted value
6: if this node has a depth of $n$, return the products on the path from the root to this node; otherwise expand this node using products in $P$ that are not in its path
7: add all newly expanded nodes into $l_{expand}$ and go to step 4
```

EXAMPLE 2. (HTS Example) *Suppose $n = 2$. We only promote products that occur in the antecedents of R_0, thus $\tilde{\gamma}_0 = \{b, c, d, f\}$ is the candidate product set. As in Figure 1, we first create an empty root, which is expanded into 4 nodes at l_1, each of which corresponds to one product in $\tilde{\gamma}_0$. The numbers on the right side of each level is the average π of this level. We can compute that $\pi(b) = 0.012$, $\pi(c) = 0.012$, $\pi(d) = 0$ and $\pi(f) = 0.025$ (the first line underneath l_1). The adjusted value of these 4 nodes are the same as their original π values (the second line underneath l_1). f has the highest π_a in l_1, thus it is expanded into 3 nodes in l_2. Similarly, we can obtain that c has the highest π_a in l_2, and since $n = 2$, c is the node returned by HTS.*

The proof of HTS's correctness [22] is beyond the scope of this paper, thus is omitted here. The worse case of HTS is when the tree is fully-expanded on each level. However, since the order of products in a product set does not affect the π value of the set, a hypothetically fully-expanded search tree will have, on each level, duplicated nodes with the same π value. Therefore in reality, only part of the tree will be expanded; a fully expanded tree will never exist. The number of expanded nodes in the worse case is $C^n_{|P|}$. For each node, PromotionValue is invoked to compute the promotion value of products on the path. Therefore the run time of HTS is $O(C^n_{|P|} * |T| * |R|)$ in the worst case. Nonetheless, in most practical scenarios, HTS runs much faster than this worse bound, which will be shown later in the experimental evaluation. HTS also has the advantage of early stopping. In order to avoid worse case HTS, we further propose efficient randomized local search.

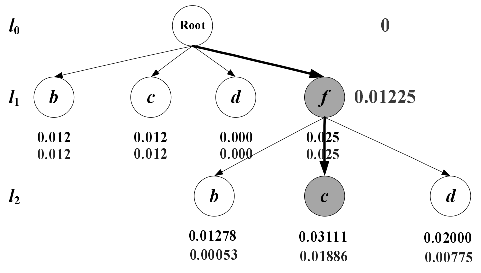


Figure 1: Applying HTS on Example in Table 2

5.2 Randomized Local Search. To approximate the optimal product set, randomized local search algorithm starts from a certain product set and refines it via swapping products based on π value until the predetermined number of iterations is reached. We implement the search using simulated annealing. In each iteration, if the swap increases π value, we keep the swap; otherwise we take it stochastically with a certain probability. This probability decreases over the iterations, based on a function. In other words, a move to take a swap that decreases π value is more likely to happen at the beginning. Local search does not guarantee a global optimum, but it has the advantage of fast convergence. It is able to generate a promising result within a reasonable amount of time. Suppose the number of iterations is δ, the run time of randomized local search is $O(\delta * |T| * |R|)$.

To further accelerate the convergence, instead of starting from a random point, we start the iterations from a heuristically good point. One alterative is to start from the products that have the highest π values, namely the result of NPS. Such local search is referred to local search with good start (LSG).

6 Customer Selection.

For customer selection, a brute-force search to traverse all possible customer combinations yields exponential time complexity $O((C^k_{|C|})^n * |T| * |R|)$, which will be analyzed later. We propose two efficient greedy customer selection algorithms, whose worst-case time complexity is both $O(n * |C| * |T| * |R|)$. A customer-specific promotion value function is proposed to model the promotion value of promoting a product a to a customer μ.

6.1 Customer-Product Promotion Value. Customer-product promotion value measures the value of promoting product a to customer μ, based on the following factors.

Customer-restrained product set promotion value. This value is computed based on the same formulation in Definition 4.3, except that we restrain the transaction set to the transaction set of this customer, denoted as T_μ. Let $T_\mu(a)$ denote those in T_μ that contain a and $T_{\mu,r}(a)$ those in $T_\mu(a)$ that support rule r, based on the transaction-rule association.

Customer temporal weight. Different customers have different temporal behaviors. Intuitively, those whose behavior exhibits recency should be more likely chosen. To implement this, each customer μ is computed a temporal weight, $\omega(\mu)$. The more recency a customer's purchase behavior exhibits, the higher weight this customer has. We divide the entire purchase period into a predetermined number of time slots and each slot has a weight. Recent time slots have higher weight and older ones have lower weight. The weighted time slots are used to compute $\omega(\mu)$, via aggregating over all μ's transactions.

Definition 6.1 formulates the customer-product promotion value (CPV).

DEFINITION 6.1. (Customer-Product Promotion Value) *The value of promoting product a to customer μ is referred to as the customer-product promotion value (CPV), $\pi_\mu(a)$. $\pi_\mu(a)$ is formulated as follows,*

$$\pi_\mu(a) = \omega(\mu) * \textstyle\sum_{r:\alpha\to b\in R(a)}\big[\frac{1}{|\alpha|} * \frac{|T\mu,r(a)|}{|T_\mu(a)|} * p(b) * \Delta_{T_\mu(a)}(r)\big], \tag{6.3}$$

where $\omega(\mu)$ is the temporal weight of customer μ and $R(a)$ is the associated rule set of product a. Each term in the sum is the contribution of a rule, which shares the same meaning as in Definition 4.3, except that the transaction set is restrained to customer μ's transactions.

The worst-case time complexity to compute $\pi_\mu(a)$ is $O(|T| * |R|)$. If k customers are chosen for each product on promotion, there are $C^k_{|C|}$ possibilities for each product. Therefore, a brute-force search over all possible customer combinations to generate the optimal customer set for a given product set of size n is: $O((C^k_{|C|})^n * |T| * |R|)$.

A naive approximation approach is to rank all the customers, for each selected product a, according to their $\pi_\mu(a)$ values. The top k customers with the highest values are chosen for each product. However, such approach overlooks potential cross-selling effect existing among one customer's transactions. Suppose the promotion set is $\{a, b\}$, and k customers are chosen for product a. Now we want to choose another set of k customers for b[3]. Assume μ is among b's top k customers with the highest values, then this naive approach will choose μ for b. However, if μ is already chosen for a and there is a rule $a \to b$ prevailing in μ's transactions, the value of promoting b to μ is impaired. This is because μ is likely to purchase b anyways once a is promoted to him/her.

[3]Note that different products' prospective customer sets are not necessarily disjoint.

6.2 Greedy Customer Selection. To tackle the above issue, we sort all the customers into a list according to their CPV values for each product. The first k customers on each list is considered the initial candidate set for each product. Before a customer is selected, we check if it has already been selected for previous products. If so, we compare the additional value of still choosing this customer with that of choosing the next available customer beyond the initial candidate set. We greedily choose the customer with a higher additional value. The generic version of such process is shown in Algorithm 3, where $\pi_\triangle(\mu, a)$ is the additional value that selecting μ for a induces, μ^* denotes the next unchosen customer that is not in a's initial candidate set and $C_k(a)$ denotes a's k chosen prospective customers.

Algorithm 3 Greedy Customer Selection

```
GreedyCS($\gamma$, $C$)
Input: The promotion product set $\gamma$, the total customer set $C$
Output: An enriched mapping set $\widetilde{\gamma}$
 1: compute CPV value $\pi_\mu(a)$ for all product-customer pairs in $\gamma \times C$, where $a \in \gamma$ and $\mu \in C$
 2: for each product $a$, rank all the customers according to their CPV values into a list in a descending manner
 3: rank products in $\gamma$ based on their aggregated CPV values of the first $k$ customers on their lists
 4: for the first product $a_0$, assign the first $k$ customers on its list as $C_k(a_0)$, add pair $(a_0, C_k(a_0))$ to $\widetilde{\gamma}$
 5: for each product $a$ in the remaining products do
 6:     for each customer $\mu$ in $a$'s list, if $\mu$ is not selected for previous products, add $\mu$ to $C_k(a)$, otherwise compute $\pi_\triangle(\mu, a)$
 7:     if $\pi_\triangle(\mu, a) > \pi_{\mu^*}(a)$, add $\mu$ to $C_k(a)$, otherwise add $\mu^*$ to $C_k(a)$
 8:     add pair $(a, C_k(a))$ to $\widetilde{\gamma}$
 9: end for
10: return $\widetilde{\gamma}$
```

We propose two methods to compute $\pi_\triangle(\mu, a)$, the cross-selling score method and the CPV increment method, which accordingly leads to two customer selection methods, the cross-selling minimizing search and the CPV maximizing search.

6.2.1 Cross-Selling Minimizing Search. We first introduce a concept named as cross-selling score to compute the additional value $\pi_\triangle(\mu, a)$, see Definition 6.2. Cross-selling score measures, for a and μ, how much cross-selling effect between a and those previous products μ is selected for, exists in μ's transactions. If the cross-selling score of μ for a is higher than a threshold, μ should not be selected for a. Otherwise, we compare the additional value of choosing μ with that of choosing μ^*.

DEFINITION 6.2. (Cross-Selling Score) *If μ is already selected for some previous products, denoted as $\hat{\gamma}_\mu$, the cross-selling score, $\mathcal{A}(\mu)$, measures how unfavorable it is to select μ again for product a based on cross-selling effect. Let r denote the rule $\hat{\gamma}_\mu \rightarrow a$, $\mathcal{A}(\mu)$ is computed as*

$$\mathcal{A}(\mu) = \frac{|T_{\mu,r}|}{|T_\mu|}. \tag{6.4}$$

The $\mathcal{A}(\mu)$ based additional value of selecting μ for a is computed as

$$\pi_\triangle(\mu, a) = (1 - \mathcal{A}(\mu)) * \sum_{\acute{a} \in \hat{\gamma}_\mu \cup a} \pi_\mu(\acute{a}). \tag{6.5}$$

The cross-selling minimizing search (CSM) aims at choosing the customers whose behavior does not exhibit a significant amount of cross-selling effect among the promoted products. It is a approach to use cross-selling score to compute $\pi_\triangle(\mu, a)$. Example 3 gives an example using the sample sets in Table 2.

EXAMPLE 3. (Cross-Selling Score Example) *Suppose the current product set on promotion is $\{b, c\}$ and $k = 1$. There are three customers shown in Table 2, Ann, Jon and Dan. Suppose that b has higher aggregated CPV value than c. We first compute $\pi_{Ann}(b)$, $\pi_{Jon}(b)$ and $\pi_{Dan}(b)$. Each customer's CPV value is computed over their own transactions. For example, we use t_1 and t_2 to compute $\pi_{Ann}(b)$. We can obtain $\pi_{Ann}(b) = 0$, $\pi_{Jon}(b) = 0$ and $\pi_{Dan}(b) \approx 0.01$. Therefore $C_1(b) = \{Dan\}$. The next step is to select one customer for c. When considering Dan as a candidate for $C_1(c)$, since Dan is already selected for b, we compute $\mathcal{A}(Dan)$. Among Dan's transactions, t_6 and t_7 support the rule $b \rightarrow c$, thus $\mathcal{A}(Dan) \approx 0.67$. We can also derive $\pi_{Dan}(c) = 0.025$. Therefore, $\pi_\triangle(Dan, c) \approx (1 - 0.67) * (0.01 + 0.025) \approx 0.012$.*

6.2.2 CPV Maximizing Search. Suppose the product set on promotion is γ, the distinct selected customer set for γ is θ, then for a customer μ in θ, its CPV is computed on all the products in γ that μ is selected for. According to our problem definition, the higher the aggregated CPV value over all customers is, the better the selected customer set is. This leads to an alternative way to compute $\pi_\triangle(\mu, a)$, as in Definition 6.3.

DEFINITION 6.3. (CPV Increment) *If μ is already selected for some previous products, denoted as $\hat{\gamma}_\mu$, the additional value of selecting μ for product a based on CPV increment is computed as in* (6.6),

$$\pi_\triangle(\mu, a) = \pi_\mu(\hat{\gamma}_\mu \cup a) - \pi_\mu(\hat{\gamma}_\mu). \tag{6.6}$$

The CPV maximizing search (CPM) directly maximizes the CPV increment of each selected customer. For each product, we rank all the customers according to the difference they cause on their CPV if they are selected for this product. The top k customers with the highest difference values are chosen for this product. This method computes $\pi_\triangle(\mu, a)$ using CPV increment. Example 4 gives an example using the sample sets in Table 2.

EXAMPLE 4. (CPV Increment Example) *We adopt the same scenario used in Example 3. When considering Dan for $C_1(c)$, since Dan is already selected for b, we compute $\pi_\triangle(Dan, c)$ as $\pi_{Dan}(b, c) - \pi_{Dan}(b)$. We can derive $\pi_{Dan}(b, c) \approx 0.02$. Thus $\pi_\triangle(Dan, c) \approx 0.02 - 0.01 = 0.01$.*

The dominant part in step 1 to step 4 of Algorithm 3 is step 1. It amounts to a run time $O(n * |C| * |T| * |R|)$. The following steps in the algorithm introduce a loop that has $n * k$ iterations, each of which triggers a $O(|T| * |R|)$ computation. The total time complexity for the above two customer selection algorithms is therefore $O(n * |C| * |T| * |R| + n * k * |T| * |R|) = O(n * |C| * |T| * |R|)$.

7 Experimental Evaluations.

Our experimental evaluation is designed to answer the following two questions: a), Effectiveness. How effective are the proposed product methods? b), Efficiency. How fast and scalable are the proposed methods?

7.1 Experiment Setup.

Data sets. Two data sets are used, a small-scale synthetic data set, *Synth*, and a large-scale real data set collected from a large department store in China, *CH-SMarket*, as summarized in Table 3.

Name	**Transaction Number**	**Product Number**	**Customer Number**
Synth	10,015	8,954	3,199
CH-SMarket	89,887	28,369	4,960

Table 3: Summary of Experiment Data

Table 3 specifies the number of transactions, number of distinct products and number of distinct customers. Among the 8,954 products in *Synth*, 27 products occur in the antecedents; among the 28,369 products in *CH-SMarket*, 307 products occur in the antecedents. We only consider those that occur in the antecedents since we only promote products that encourage the purchase of other products. We use the LUCS-KDD library[4] to implement FP-Growth based association rule mining. The rule mining parameters are $minSup = 0.05\%$ and $minConf = 70\%$ for *Synth*, and $minSup = 0.01\%$ and $minConf = 50\%$ for *CH-SMarket*, where $minSup$ is the minimum support value. A small $minSup$ value is due to a huge number of transactions and products and each product occurs only a couple of times in the entire set. 41 rules are generated from *Synth* and 625 rules are generated from *CH-SMarket*.

Parameter settings and machine configurations. We assume a uniform profit for all products. We set $p = 20\%$. The number of iterations of local search δ, is set to 100 for *Synth* and 1000 for *CH-SMarket*. The cross-selling score threshold for cross-selling based customer selection, is set to 0.5. For computational cost, we report the wall-clock time. All experiments are run on the same machine with Fedora Release 8, Linux V. 2.6.23, Intel Xeon 8-Core, 2.5GHz and 40G RAM.

7.2 Product Selection.

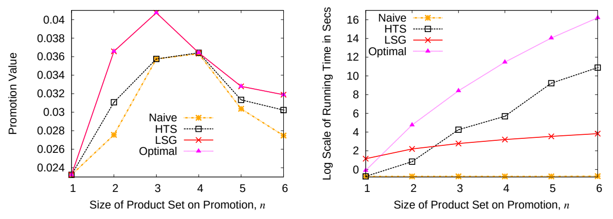
In this section, we compare the performance of NPS, HTS and LSG. We evaluate their performance from three perspectives, the π value, the wall-clock execution time and the total correlation value. In order to evaluate the effectiveness, we further introduce a baseline method, which traverses all possible product combinations of size n for the optimal set. A pruning technique is implemented to compute an upper bound for each candidate set, thus unpromising candidates are pruned. We ensure that computing the upper bound is less costly than computing π. This baseline method is referred to optimal search (OS), which generates the optimal product set.

Experiment results demonstrate that LSG performs the best in terms of effectiveness, efficiency and scalability. Both HTS and LSG outperform NPS in maximizing promotion values, especially LSG. On *Synth* data set, LSG achieves exactly the same results as OS, which generates optimal result sets. Besides, LSG has a prominent advantage over both HTS and OS w.r.t. run time. The run time of LSG scales well with n, whereas HTS and OS both suffer from poor scalability and time inefficiency. In addition, LSG is able to generate more orthogonal product set compared to NPS.

7.2.1 Experiment on Synth.

All four methods are compared on *Synth*. The number of products on promotion, n, ranges from 1 to 6. The main purpose of this phase is to thoroughly compare the performance of all the proposed methods from various perspectives.

Promotion value evaluation. For a certain n, each method generates a set of products on promotion. We compute the promotion value π for each set. Figure 2(a) visualizes the comparison of NPS, HTS, LSG and OS on π value using *Synth*, where "Naive" refers to NPS and "Optimal" refers to OS. It is shown that the line of LSG



(a) Promotion Value π vs. n (b) Run Time vs. n

Figure 2: Product Selection Results on *Synth* Set

overlaps completely with that of OS, which proves the effectiveness of LSG. NPS performs poorly compared to other three methods, which is expected due to its inability to tackle cross-selling effect. HTS fails to generate as good set as LSG, however still outperforms NPS due to its heuristic search process.

Run time evaluation. We observe the changes of run time over various values of n in order to measure the efficiency and scalability. As shown in Figure 2(b), which is drawn in logarithm scale, OS induces an exponential run time over n. HTS also scales very badly with n. LSG is able to maintain a near-constant logarithm run time, which proves its run time efficiency and scalability over n. Though NPS achieves the best run time, its π value evaluation is not satisfying. Therefore, LSG performs the best on *Synth* due to its effectiveness, efficiency and scalability.

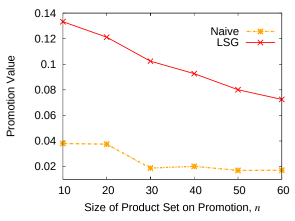
7.2.2 Experiment on CH-SMarket.

It is experimentally proven in the last section that HTS and OS both scale badly with n. Therefore in this section, for a larger-scale data set *CH-SMarket*, we compare the performance of NPS and LSG. n ranges from 10 to 60. The main purpose of this phase is to further validate the conclusions derived from experiments on *Synth* and to produce the product set for the next step, the customer selection.

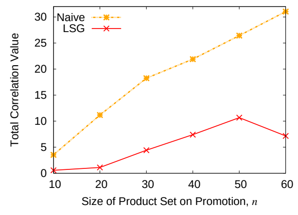
Promotion value evaluation. Similarly, Figure 3(a) shows the comparison of NPS and LSG on π using *CH-SMarket*. It is shown that LSG prominently outperforms NPS in maximizing promotion value. It is also interesting to see that the π value actually slightly decreases when n increases. This demonstrates that promoting more products does not necessarily imply higher promotion effect, since a larger set of products on promotion indicates a higher probability of cross-selling effect between promoted products.

Correlation coefficient evaluation. Another perspective is to measure how orthogonal the products are. A good product set should have products uncorrelated to each other. When a supermarket designs a brochure for product promo-

[4]http://www.csc.liv.ac.uk/ frans/KDD/Software/FPgrowth/fpGrowth.html



(a) Promotion Value π vs. n



(b) Total Correlation Value $Corr$ vs. n

Figure 3: Product Selection Results on *CH-SMarket* Set

Product Name	**Number of Occurrence**
Whiskas® Mackerel	10
Yushiyuan® Preserved Apple	9
OLAY® Translucent Aqua Foundation	9
Shiseido® Revital Moisturizer	8
ZIPPO® Gift Package	8

Table 4: Most Frequently Selected Products on *CH-SMarket* Set, $n = 30$

tion, it does not want to include similar products repeatedly; instead, it wants to have the products as distinct as possible. In this respect, for each generated product set γ, we design a total correlation coefficient value $Corr(\gamma)$ to measure the orthogonality of γ. $Corr(\gamma)$ is the sum of all the pair-wise correlation values, as shown in (7.7),

$$Corr(\gamma) = \sum_{a,b \in \gamma} \frac{|T(a) \cap T(b)|}{\sqrt{|T(a)| * |T(b)|}}, \tag{7.7}$$

where $T(a) \cap T(b)$ denotes the set of transactions that contain both products a and b. Figure 3(b) shows the comparison of NPS and LSG on the total correlation value using *CH-SMarket*. It is demonstrated that LSG again outperforms NPS prominently in orthogonality.

Case study: most frequently selected products. An important aspect of product selection is to uncover the patterns of popular products. What type of products are more likely to be selected by our methods? Do our methods truly promote the right products? In order to answer these questions, we run different experiments to generate product sets selected by different methods and parameters. We include another type of randomized local search, which starts from randomly selected n products. We name it local search with a random start (LSR). We vary the number of iterations for LSG and LSR to get various product sets, when n is fixed. We observe the most frequently occurred products among these sets. Table 4 shows the top five most frequently selected products by NPS, LSR-500, LSR-800, LSR-1000, LSR-1200, LSR-1500, LSG-500, LSG-800, LSG-1000, LSG-1200 and LSG-1500, where the number after "-" denotes the number of iterations. The second column shows the number of occurrence of the product in the 11 sets.

An interesting phenomenon we call "brand-effect" can be seen from Table 4, where products from brands which have a relatively larger line of products, such as Whiskas® and Shiseido®, tend to get selected by our methods. This is because products from such brands tend to appear more frequently in association rules. For example, if a customer has purchased OLAY® Translucent Aqua Foundation, he or she might be very interested to continue purchasing other OLAY® products. If we name a rule that has all of its products from the same brand an inner-brand rule, it is observed that out of the 625 rules generated from *CH-SMarket*, 455 rules are inner-brand rules.

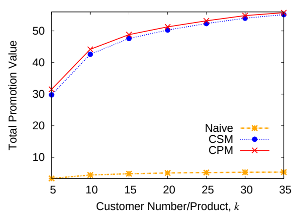
7.3 Customer Selection.

In this section, we use one result set ($n = 50$ and $\delta = 1000$) generated by LSG from the *CH-SMarket* set to further identify prospective customers.

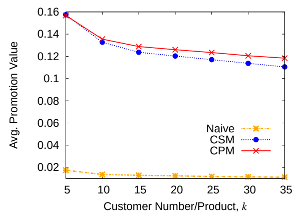
We compare the performance of CSM and CPM. We further introduce a baseline method, which selects, for each product a on promotion, the top k customers with the highest customer-product promotion values $\pi_\mu(a)$. This baseline method is referred to as naive customer search (NCS). The values of k are $\{5, 10, 15, 20, 25, 30, 35\}$. We evaluate the selected customer set using the aggregated customer-product promotion value and the total correlation value.

Experiment results demonstrate that CPM performs the best in terms of maximizing promotion value and minimizing correlation value. Both CSM and CPM prominently outperform NCS. In addition, CSM is able to generate the biggest number of distinct customers.

Promotion value evaluation. Suppose the product set on promotion is γ, the total promotion value of selected customers is the sum of π_μ over all the selected customers. Each π_μ is the promotion value of the products in γ that customer μ is selected for. Figure 4(a) shows the comparison of NCS, CSM and CPM on total promotion value using *CH-SMarket*, where "Naive" refers to NCS. We also compare



(a) Total Customer-Product Promotion Value vs. k

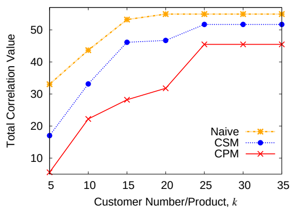


(b) Avg. Customer-Product Promotion Value vs. k

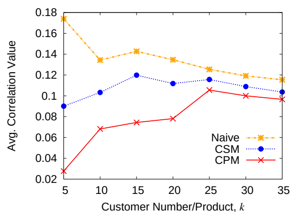
Figure 4: Customer-Product Promotion Value vs. k on *CH-SMarket* Set

the average promotion value per customer, which is the total promotion value divided by the number of distinct selected customers, as shown in Figure 4(b). It is shown that CSM and CPM outperform NCS in terms of both total value and average value. CPM performs the best.

Correlation coefficient evaluation. The total correlation value is the sum of the total correlation values of the product set (see (7.7)) for each selected customer. A lower correlation value indicates a more orthogonal distribution of products to prospective customers. Figure 5(a) shows the comparison of NCS, CSM and CPM on total correlation value using *CH-SMarket*. We also compare the average



(a) Total Correlation Value vs. k



(b) Avg. Correlation Value vs. k

Figure 5: Correlation Value vs. k on *CH-SMarket* Set

correlation value per customer, which is the total correlation value divided by the number of distinct selected customers, as shown in Figure 5(b). As indicated, CPM again, achieves the best performance. NCS performs poorly in terms of both total and average correlation values. Our experiments prove that the proposed CPM greedy method serves to be an effective approach for customer selection.

Number of distinct customers. Number of distinct customers gives an idea of the range of customers covered by this promotion. This evaluates customer set from another perspective. Supermarkets usually want to promote their products to a wider range of people, which also implies a wider range of demographic features. our study does not rely on demographic data, therefore we use the number of distinct customers to preliminarily evaluate the three methods. Figure 6 shows the comparison of NCS, CSM and CPM on distinct customer numbers using *CH-SMarket*. It is shown that CSM generates the highest number of distinct customers. This helps us conclude that CSM is a good approach to consider if one wants to propagate the promotion as widely as possible. On the other hand, CPM performs better in terms of total promotion effect and orthogonality.

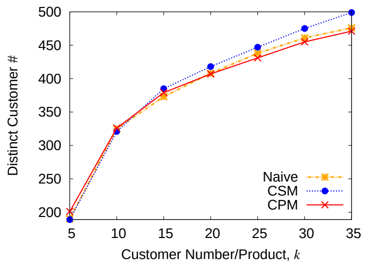


Figure 6: Distinct Customer No. vs. k on *CH-SMarket* Set

Customer	Products	Products in γ	Top Purchased Brands
A	6,146	7	Pedigree® (7),Shiseido® (1)
B	5,836	5	Shiseido® (8),Master Kong® (2)

Table 5: Two Most Frequently Selected Customers on *CH-SMarket* Set, $n = 30$, $k = 5$

Case study: most frequently selected customers. Similar to product selection, we probe into frequently selected customers to observe patterns. Such information assists us to further verify the effectiveness of our algorithms. We seek to answer the following questions: what type of customers are selected by our algorithms? What is their historical behavior like w.r.t. the product set on promotion? In this section, we only probe into the selected customers by CSM and CPM when $n = 30$ and $k = 5$ due to limited space.

The product set on promotion, γ, is generated by LSG when $\delta = 1000$. It is observed that these 30 products come from 16 different brands, such as Shiseido®, Pedigree®, Heinz®, etc. This phenomenon conforms with the previously discovered "brand-effect". If the top customers are those that are selected for most of the products in γ, the two customer sets by CSM and CPM yield the same top two customers. For identity protection, we denote them as customer A and B. The second column in Table 5 is the total number of distinct products the customer has purchased, the third column is the number of distinct products the customer has purchased in γ and the fourth column is the purchased brands from the 16 brands, among the customer's top 20 products. The number in the parentheses in the fourth column is the number of distinct products purchased from this brand. As we can see, both A and B exhibit intense historical purchase for brands within γ. A purchases Pedigree® frequently and B purchases Shiseido® frequently. In addition, the first column shows that both A and B are avid consumers. Based on the assumption that consumers who have a certain purchase pattern in the past will likely continue such pattern, we believe it is sensible to select customers such as A and B.

8 Conclusion.

In this paper, we propose a hybrid data mining framework to optimize product selection and customer selection, in order to maximize promotional profits. We formally formulate the definition for product selection, customer selection and product set promotion value. Association rule mining based promotion value function is designed to evaluate a product set. Approximation algorithms are proposed to address the problems in a more efficient manner. Our experiment results on both synthetic and real-world data sets prove the efficiency and effectiveness of the proposed methods.

In the future, we will first apply the algorithms in real stores for practical validation. This will be the key to proving the quality of the proposed algorithms. Secondly, for customer temporal weight calculation, recency rule applies only for products that are purchased frequently; for those that are not, being purchased recently actually implies that they will not be purchased again in the near future.Our next version of customer temporal weight function will tackle this issue as well. Thirdly, more information will be included into the promotion value computation, such as the temporal and categorical information of the products, and the demographic information of the customers. Last but not the least, possibility of optimizing product and customer selection simultaneously instead of in sequence will also be explored.

9 Acknowledgement

The work was supported by NSF Grant IIS-0905084.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, *Mining association rules between sets of items in large databases*, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 1993, pp. 207-216.

[2] L. K. Branting, *Learning feature weights from customer return-set selections*, Knowl. Inf. Syst., 2004, pp. 188-202.

[3] T. Brijs, B. Goethals, G. Swinnen, K. Vanhoof, and G. Wets, *A data mining framework for optimal product selection in retail supermarket data: the generalized profset model*, Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2000, pp. 300-304.

[4] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, *Using association rules for product assortment decisions: a case study*, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 1999, pp. 254-260.

[5] ———, *Building an association rules framework to improve product assortment decisions*, Data Min. Knowl. Discov., 2004, pp. 7-23.

[6] S. Brin, R. Motwani, and C. Silverstein, *Beyond market baskets: Generalizing association rules to correlations*, 1997.

[7] P. Chan, *A non-invasive learning approach to building web user profiles*, Workshop on Web Usage Analysis and User Profiling, Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, Auguest, 1999.

[8] P. B. Chou, E. Grossman, D. Gunopulos, and P. Kamesam, *Identifying prospective customers*, Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2000, pp. 447-456.

[9] M. Ester, R. Ge, W. Jin, and Z. J. Hu, *A microeconomic data mining problem: customer-oriented catalog segmentation*, Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2004, pp. 557-562.

[10] J. W. Han, J. Pei, and Y. W. Yin, *Mining frequent patterns without candidate generation*, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 2000, pp. 1–12.

[11] B. Kitts, *The ecology of the retail store*, 2000, working paper.

[12] B. Kitts, D. Freed, and M. Vrieze, *Crosssell: A fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities*, Proceedings of ACM SIGKDD International Conference, ACM Press, 2000, pp. 437-446.

[13] J. Kleinberg, C. Papadimitriou, and P. Raghavan, *A microeconomic view of data mining*, Data Min. Knowl. Discov., 1998, pp. 311-324, .

[14] ———, *Segmentation problems*. J. ACM, 2004, pp. 263–280.

[15] H. Toivonen, *Sampling large databases for association rules*, Proceedings of the 22th International Conference on Very Large Data Bases, San Francisco, CA, USA, 1996, pp. 134-145.

[16] A. Veloso, M. Zaki, W. Meira Jr., and M. Gonalves, *The metric dilemma: Competenceconscious associative classification*, Proceedings of the Ninth SIAM International Conference on Data Mining, April-May, 2009,pp. 918-929.

[17] K. Wang, L. Tang, J. W. Han, and J. Q. Liu, *Top down fp-growth for association rule mining*, In PAKDD 02, London, UK, 2002, Springer-Verlag, pp. 334–340,

[18] K. Wang, and M. Y. T. Su, *Item selection by "hubauthority" profit ranking*, Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2002, pp. 652-657.

[19] R. C. W. Wong, A. W. C. Fu, and K. Wang, *Mpis: Maximal-profit item selection with cross-selling considerations*, Proceedings of IEEE International Conference on Data Mining, 2003.

[20] ———, *Data mining for inventory item selection with cross-selling considerations*, Data Min. Knowl. Discov., 2005, pp. 81-112.

[21] L. H. Wu, L. Liu, and J. Li, *Evaluating customer lifetime value for customer recommendation*, Proceedings of ICSSSM 05, June 2005, pp. 138-143.

[22] Y. H. Yang, C. H. Hao, and A. S. Ge, *Product Selection for Store Promotions*, Proceedings of the 3rd China Summer Workshop on Information Management, June 2009.