MPIS: Maximal-Profit Item Selection with Cross-Selling Considerations

Raymond Chi-Wing Wong, Ada Wai-Chee Fu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
cwwong,adafu@cse.cuhk.edu.hk

Ke Wang
Department of Computer Science
Simon Fraser University, Canada
wangk@cs.sfu.ca

Abstract

In the literature of data mining, many different algorithms for association rule mining have been proposed. However, there is relatively little study on how association rules can aid in more specific targets. In this paper, one of the applications for association rules - maximal-profit item selection with cross -selling effect (MPIS) problem - is investigated. The problem is about selecting a subset of items which can give the maximal profit with the consideration of cross-selling. We prove that a simple version of this problem is NP-hard. We propose a new approach to the problem with the consideration of the loss rule *- a kind of association rule to model the cross-selling effect. We show that the problem can be transformed to a quadratic programming problem. In case quadratic programming is not applicable, we also propose a heuristic approach. Experiments are conducted to show that both of the proposed methods are highly effective and efficient.*

1 Introduction

Recent studies in the retailing market have shown a winning edge for customer-oriented business, which is based on decision making from better knowledge about the customer behaviour. Furthermore, the behaviour in terms of sales transactions is considered significant [6]. This is also called market basket analysis. We consider the scenario of a supermarket or a large store, typically there are a lot of different items offered, and the amount of transactions can be very large. For example [11] quoted the example of American supermarket chain Wal-Mart, which keeps about 20 million sales transactions per day. This growth of data requires sophisticated method in the analysis.

At about the same time, association rule mining [3] has been proposed by computer scientists, which aims at understanding the relationships among items in transactions or market baskets. However, it is generally true that the association rules in themselves do not serve the end purpose of the business people. We believe that association rules can aid in more specific targets. Here we investigate the application of association rule mining on the problem of market basket analysis. As pointed out in [6], a major task of talented merchants is to pick the profit generating items and discard the losing items. It may be simple enough to sort items by their profit and do the selection. However, by doing that we would have ignored a very important aspect in market analysis, and that is the cross-selling effect. The cross-selling effect arises because there can be items that do not generate much profit by themselves but they are the catalysts for the sales of other profitable items. Recently, some researchers [17] suggest that association rules can be used in the item selection problem with the consideration of relationships among items. Here we follow this line of work in what we consider as investigations of the application of data mining in the decision-making process of an enterprise.

In this paper, the problem of Maximal-Profit Item Selection with Cross-Selling Considerations (MPIS) is studied. With the consideration of the cross-selling effect, MPIS is the problem of finding a set of J items such that the total profit from the item selection is maximized, where J is an input parameter. This problem arises naturally since a store or a company typically changes the products they carry once in a while. The products that can generate the best profits should be retained and poor-profit items can be removed, then new items can be introduced into the stock. In this way the business can follow the market needs and generate the best possible results for both the business and the customers. In order to determine the profit value of an item, one can rely on expert knowledge. However, since this can be a highly complex issue especially for a large store with thousands of products for sale, we can try to apply data mining techniques, based on a history of customer purchase records.

Hence the problem is how to determine a subset of a given set of items based on a history of transaction records, so that the subset should give the best profits, with considerations of the cross-selling effects. We show that a simple version of this problem is NP-hard. We model the cross-selling factor with a special kind of association rule called *loss rule*. The rule is of the form $I_a \to \diamond d$, where I_a is an item and d is a set of items, and $\diamond d$ means any items in d. This loss rule helps to estimates the loss in profit for item I_a if the items in d are missing after the selection. The rule corresponds to the cross-selling effect between I_a and d.

To handle this problem, we propose a quadratic programming method (QP) and a heuristics method called MPIS_Alg. In QP, we express the total profit of the item selection in quadratic form and solve a quadratic optimization problem. Algorithm MPIS_Alg is a greedy approach which uses an estimate of the *benefits* of the items to prune items iteratively for maximal-profit. From our experiment, the profitabilities of our two proposed algorithms are greater than that of naive approach for all data sets. On average, the profitability of both QP and MPIS_Alg is 1.33 times higher than the naive approach for the synthetic data set. In a real drugstore data set, the best previous method HAP [26] gives a profitability that is about 2.9 times smaller than MPIS_Alg. When the number of items is large (as in the drugstore data set), the execution time of HAP is 6.5 times slower than MPIS_Alg. These shows that the MPIS_Alg is highly effective and efficient.

2 Problem Definition

Maximal-profit item selection (MPIS) is a problem of selecting a subset from a given set of items so that the estimated profit of the resulting selection is maximal among all choices. Our definition of the problem is close to [26]. Given a data set with m transactions, $t_1, t_2, ..., t_m$, and n items, $I_1, I_2, ..., I_n$. Let $I = \{I_1, I_2, ..., I_n\}$. The profit of item I_a in transaction t_i is given by $prof(I_a, t_i)$. [1] Let $S \subset I$ be a set of J selected items. In each transaction t_i, we define two symbols, t'_i and d_i, for the calculation of the total profit.

$$t'_i = t_i \cap S, \qquad d_i = t_i - t'_i$$

t'_i	set of items selected in S in transaction t_i
d_i	set of items not selected in S in transaction t_i

Suppose we select a subset S of items, it means that some items in $I_1, ..., I_n$ will be eliminated. The transactions $t_1, ..., t_m$ might not occur in exactly the same way if some items have been removed beforehand, since customers may not make some purchase if they know they cannot get some of the items. Therefore, the profit $prof(I_a, t_i)$ can be affected if some items are removed from the stock. This is caused by the cross-selling factor. The cross-selling factor is modeled by $csfactor(D, I_a)$, where D is a set of items, and $0 \leq csfactor(D, I_a) \leq 1$. $csfactor(D, I_a)$, is the fraction of the profit of I_a that will be lost in a transaction if the items in D are not available. Note that the cross-selling factor can be determined in different ways. One way is by the domain experts. We may also have a way to derive this factor from the given history of transactions.

Definition 1 Total Profit of Item Selection: *The total profit of an item selection S is given by*
$P = \sum_{i=1}^{m} \sum_{I_a \in t'_i} prof(I_a, t_i)(1 - csfactor(d_i, I_a))$

We are interested in selecting a set of J items so that the total profit is the maximal among all such sets.

MPIS: *Given a set of transactions with profits assigned to each item in each transaction, and the cross-selling factors, $csfactor()$, pick a set S of J items from all given items which gives a maximum profit.*

This problem is at least as difficult as the following decision problem, which we call the decision problem for MPIS:

MPIS Decision Problem: Given a set of items and a set of transactions with profits assigned to each item in each transaction, a minimum benefit B, and cross-selling factors, $csfactor()$, can we pick a set S of J items such that $P \geq B$?

In our proof in the following, we consider the very simple version where $csfactor(d_i, I_a) = 1$ for any non-empty set of d_i. That is, any missing item in the transaction will eliminate the profit of the other items. This may be a much simplified version of the problem, but it is still very difficult.

2.1 NP-hardness

Theorem 1 *The maximal-profit item selection (MPIS) decision problem where $csfactor(d_i, I_a) = 1$ for $d_i \neq \phi$ and $csfactor(d_i, I_a) = 0$ for $d_i = \phi$ is NP-hard.*

Proof sketch: We shall transform the problem of CLIQUE to the MPIS problem. CLIQUE [9] is an NP-complete problem defined as followd:

CLIQUE: Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, is there a subset $V' \subseteq V$ such that $|V'| \geq K$ and every two vertices in V' are joined by an edge in E ?

The transformation from CLIQUE to MPIS problem is described as follows: Set $J = K$, $B = K(K-1)$. For each vertex $v \in V$, construct an item. For each edge $e \in E$, where $e = (v_1, v_2)$, create a transaction with 2 items $\{v_1, v_2\}$. Set $prof(I_j, t_i) = 1$, where t_i is a transaction created in the above, $i = 1, 2, ..., |E|$, and I_j is an item in t_i.

It is easy to see that this transformation can be constructed in polynomial time. It is also easy to verify that when the problem is solved in the transformed MPIS, the original clique problem is also solved. Since CLIQUE is an NP-complete problem, the MPIS problem is NP-hard. □

3 Related Work

In recent years the problem of association rule mining has received much attention. We are given a set I of items, and a set of transactions. Each transaction is a subset of I. An *association rule* has the form $X \rightarrow I_j$, where $X \subseteq I$ and $I_j \in I$; the $support$ of such a rule is the fraction of transactions containing all items in X and item I_j; the confidence for the rule is the fraction of the transactions containing all items in set X that also contain item I_j. The problem is to find all rules with sufficient support and confidence. Some of the earlier work include [22, 4, 21].

3.1 Item Selection Related Work

There are some recent works on the maximal-profit item selection problem. PROFSET [8, 7] models the cross-selling effects by *frequent itemsets*, which are sets of items co-occurring frequently. A *maximal frequent itemset* is a frequent itemset which does not have a frequent item superset. The profit margins of maximal itemsets are counted in the total profit. The problem is formulated as 0-1 linear programming that aims to maximize the total profit.

However, PROFSET has several drawbacks as pointed out in [26]. More details can be found in [26].

HAP [26] is a solution of a similar problem. It applies the "hub-authority" profit ranking approach [23] to solve the maximal profit item-selection problem. Items are considered as vertices in a graph. A link from I_i to I_j represents the cross-selling effect from I_i to I_j. A node I_j is a good *authority* if there are many links of the form $I_i \rightarrow I_j$ with a strong

[1] This definition generalizes the case where profit of an item is fixed for all transactions. We note that the same item in different transactions can differ because the amount of the item purchased are different, or the item can be on discount for some transactions and the profit will be reduced. If the profit of an item is uniform over all transactions, we can set $prof(I_a, t_i)$ to be a constant over all i.

strength of the link. The HITS algorithm [18] is applied and the items with the highest resulting authorities will be the chosen items. It is shown that the result converges to the principal eigenvectors of a matrix defined in terms of the links, confidence values, and profit values.

However, HAP also has some weaknesses. (1) Problems of dead ends or spider traps as illustrated in [25] can arise. For example, if there is an isolated subgraph with a cycle while other items are not connected, then the authority weight and hub weight of all items in the cycle are accumulated and is increased to an extremely high value, giving an over-estimating ranking for these items. (2) In HAP, the authority weight of an item I_j depends on the profit of any other item I_i with the association rule $I_i \rightarrow I_j$. It is possible that some items with low/zero profit gain very high authority weights, and are selected by HAP. In fact the real data set we shall use in the experiments exhibits this phenomenon, and HAP cannot give a competitive solution.

4 Cross selling effect by Association Rules

In Section 2, we did not specify how to determine the cross-selling effect $csfactor$ of some items for other items. In previous work [26], the concept of association rules is applied to this task. Here we also apply the ideas of association rules for the determination of $csfactor$.

Let us estimate the possible profit from a given set of transaction. If all items are selected, the profit is the same as the given profit. Suppose we have made a selection S of J items from the set of items. Now some transactions may lose profits if some items are missing. Consider a transaction t_i in our transaction history, suppose some items, says I_a, are selected in S but some items are not selected (i.e. d_i). Then if we have a rule that purchasing I_a always "implies" at least one element in d_i then it would be impossible for transaction t_i to exist after the selection of S, since t_i contains I_a and no element in d_i after the selection. The profit generated by t_i from I_a should be removed from our estimated profit.

We can model the above rule by an association rule. In fact, we can model the cross-selling factor in the total profit of item selection $csfactor(d_i, I_a)$ by $conf(I_a \rightarrow \diamond d_i)$, where $\diamond d_i$ is given by the following:

Definition 2 *Let $d_i = \{Y_1, Y_2, Y_3, ..., Y_q\}$ where Y_i refers to a single item for $i = 1, 2, .., q$, then $\diamond d_i = Y_1 \vee Y_2 \vee Y_3 \vee .... \vee Y_q$.*

The rule $I_a \rightarrow \diamond d_i$ is called a *loss rule*. The rule $I_a \rightarrow \diamond d_i$ indicates that a customer who buys the item I_a must also buy at least one of the items in d_i. If none of the items in d_i are available then the customer also will not purchase I_a. Therefore, the higher the confidence of "$I_a \rightarrow \diamond d_i$", the more likely the profit of I_a in t_i should not be counted. This is the reasoning behind the above definition.

The total profit is to estimate the amount of profit we would get from the set of transaction $t_1, ...t_m$, if the set of items is reduced to the selected set S. From Definition 1, we have

Definition 3 Total Profit of Item Selection (association rule based): *The association rule based total profit of item selection S is given by*
$P = \sum_{i=1}^{m} \sum_{I_a \in t'_i} prof(I_a, t_i)(1 - conf(I_a \rightarrow \diamond d_i))$

For the special cases, if all items in transaction t_i are selected in the set S, then d_i is empty, t_i will not be affected and so the profit of transaction t_i would remain unchanged. If no item in transaction t_i is selected, then the customer could not have executed the transaction t_i, then t'_i is an empty set, and the profit of transaction t_i becomes zero after we have made the selection.

The loss rule $I_a \rightarrow \diamond d_i$ is treated as an association rule. The confidence of this rule is defined in a similar manner as for the association rule:

Definition 4 *$conf(I_a \rightarrow \diamond d_i)$ is computed as*
$$\frac{\textit{no. of transactions containing } I_a \textit{ and any element in } d_i}{\textit{no. of transactions containing } I_a}$$

5 Quadratic Programming

Linear programming or non-linear programming has been applied for optimization problems in many companies or businesses and has saved millions of dollars in their running [12]. The problem involves a number of decision variables, an objective function in terms of these variables to be maximized or minimized, and a set of constraints stated as inequalities in terms of the variables. In linear programming, the objective function is a linear function of the variables. In quadratic programming, the objective function must be quadratic. That means the terms in the objective function involve the *square* of a variable or the *product* of two variables. If s is the vector of all variables, a general form of such a function is $P = f^T s + \frac{1}{2} s^T Q s$ where f is a vector and Q is a symmetric matrix. If the variables take binary values of 0 and 1, the problem is called zero-one quadratic programming.

In this section, we propose to tackle the problem of MPIS by means of zero-one quadratic programming. We shall show that the problem can be approximated by a quadratic programming problem. Let $s = (s_1 s_2 ... s_n)^T$ be a binary vector representing which items are selected in the set S. $s_i = 1$ if item I_i is selected in the output. Otherwise, $s_i = 0$. The total profit of item selection P can be approximated by the quadratic form $f^T s + \frac{1}{2} s^T Q s$ where f is a vector of length n and Q is an n by n matrix in which the entries are derived from the given transactions. The objective is to maximize $f^T s + \frac{1}{2} s^T Q s$, subject to $\sum_{i=1}^{n} s_i = J$. The term $\sum_{i=1}^{n} s_i = J$ means that there are J items to be selected.

With a little overloading of the term t_i, we say that $t_i = (t_{i1} t_{i2} ... t_{in})^T$ is a binary vector representing which items are in the transaction t_i. $t_{ij} = 1$ if item I_j is in the transaction t_i. Otherwise, $t_{ij} = 0$. Similarly, t'_i is a binary vector representing which items are selected in S in the transaction t_i. d_i is a binary vector representing which items are not selected in S in the transaction t_i.

Then, we have the following. For $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$, $t'_{ij} = t_{ij} \times s_j$ and $d_{ij} = t_{ij} - t'_{ij}$.

n_i	number of transactions containing item I_i
n_{ij}	number of transactions containing I_i and I_j
$\|I_{i1},...I_{ij}\|$	number of transactions containing $\{I_{i1},...,I_{ij}\}$

Observation 1 *The confidence $conf(I_j \to \diamond d_i)$ can be approximated by $\frac{1}{n_j}\sum_{k=1}^{n} d_{ik}n_{jk}$.*

The above observation is based on the principle of inclusion-exclusion in set theory. To see this, let us consider the numerator in Definition 4 and let it equal to $g(I_a, d_i)$.

Definition 5 *Let $D \subset I$, $D = \{Y_1, Y_2, ..., Y_q\}$ and $I_x \notin D$, where Y_i refers to a single item for $i = 1, 2, .., q$.*

$$g(I_x, D) = \sum_{1\leq i\leq q}|I_xY_i| - \sum_{1\leq i<j\leq q}|I_xY_iY_j| + \sum_{1\leq i<j<k\leq q}|I_xY_iY_jY_k| - ... + (-1)^{n+1}|I_xY_1Y_2...Y_q|$$

where $|I_xY_iY_j...|$ is the number of transactions containing the items I_x, Y_i, Y_j, ...

□

We have

$$conf(I_j \to \diamond d_i) = \frac{g(I_j, d_i)}{\text{no. of transactions containing item } I_j}$$
$$\approx \min\left(\frac{\sum_{1\leq k\leq n}|I_jI_k| \times d_{ik}}{\text{no. of transactions containing item } I_j}, 1\right)$$
$$= \min\left(\frac{1}{n_j}\sum_{k=1}^{n} d_{ik}n_{jk}, 1\right)$$

The reason why the above approximation is acceptable is that the number of transactions containing a set of items $\mathcal{J}$ is typically smaller than the number of transactions containing a subset of $\mathcal{J}$. Hence $|I_jI_kI_l|$ is typically much smaller than $|I_jI_k|$, etc. From this approximation we can deduce the following theorem.

Theorem 2 *The total profit of item selection can be approximated by the quadratic form $P = f^Ts + \frac{1}{2}s^THs$ where f is a vector of size n and H is an n by n matrix.*

Proof sketch:

$$P \approx \sum_{i=1}^{m}\sum_{j=1}^{n} t'_{ij}prof(I_j, t_i)\left(1 - \frac{1}{n_j}\sum_{k=1}^{n} d_{ik}n_{jk}\right)$$
$$= \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}s_jprof(I_j, t_i)\left(1 - \frac{\sum_{k=1}^{n}(t_{ik} - t'_{ik})n_{jk}}{n_j}\right)$$
$$= f^Ts + \frac{1}{2}s^THs$$

where

$$f = (f_j | f_j = \sum_{i=1}^{m} t_{ij}prof(I_j, t_i)(1 - \frac{1}{n_j}\sum_{k=1}^{n} t_{ik}n_{jk}) \text{ for } j = 1, 2, ..., n)^T$$

$$H = (h_{jk} | h_{jk} = \frac{2n_{jk}}{n_j}\sum_{i=1}^{m} t_{ij}prof(I_j, t_i)t_{ik} \text{ for } j, k = 1, 2, ..., n)$$

□

Corollary 1 *P can be approximated by $P' = f^Ts + \frac{1}{2}s^TQs$ where Q is a symmetric n by n matrix.*

The corollary follows because

$$P = f^Ts + \frac{1}{2}s^THs = f^Ts + \frac{1}{2}s^TQs$$

where

$Q = (q_{ij})$ and $q_{ij} = \frac{1}{2}(h_{ij} + h_{ji})$ for all $i, j = 1, 2, ..., n$

Since the value of s_i is either 0 or 1, from the above corollary, we have approximated the problem of MPIS by that of 0-1 quadratic programming with the maximization of P' and an equality constraint of $\sum_i s_i = J$:

Maximize $P' = f^Ts + \frac{1}{2}s^TQs$
such that $\sum_{i=1}^{n} s_i = J$, *and*
$s_i = 0$ *or* $s_i = 1$ *for* $i = 1, 2, .., n$

Any 0-1 quadratic programming problem is polynomially reducible to an unconstrained binary quadratic programming problem [16]. An unconstrained binary quadratic programming problem can be transformed to a binary linear programming problem (zero-one linear programming) [5]. More related properties can be found in [20] and [14]. Zero-one linear programming and quadratic programming are known to be NP-complete [24]. However, there exist programming tools which can typically return good results within a reasonable time for moderate problem sizes. We shall apply such a tool in our experiments which will be presented in Section 7.

6 Algorithm MPIS_Alg

Since quadratic programming is a difficult problem, and existing algorithms may not scale up to large data sizes, we propose also a heuristical algorithm called Maximal-Profit Item Selection (MPIS_Alg). This is an iterative algorithm. In each iteration, we estimate a selected item set with respect to each item based on its "neighborhood" in terms of cross-selling effects, and hence try to estimate a profit for each item that would include the cross-selling effect. With the estimated profit we can give a ranking for the items so that some pruning can be achieved in each iteration. The possible items for selections will become more and more refined with the iterations and when the possible set reaches the selection size, we return it as the result.

There are some factors that make this algorithm desirable: (1) We utilize the exact formula of the profitability in the iterations. This will steer the result better toward the goal of maximal profits compared to other approaches [26] that do not directly use the formula. (2) With the "neighborhood" consideration, the item pruning at each iteration usually affect only a minor portion of the set of items and hence introduce only a small amount of computation for an iteration. Compared to the HAP approach where the entire cross-selling matrix is involved in each iteration, our approach can be much more efficient when the number of items is large.

Before describing the algorithm, we define a few terms that we use. If a transaction contains I_k only, the transaction is an *individual transaction* for I_k. The **individual count** c_k, of an item I_k is the total number of individual transactions for I_k. The individual count reflects the frequency of an item appearing without association with other items.

Let Z_k be the set of transactions that contain I_k, the **average profit** is given by $p_k = (\sum_{t_i \in Z_k} prof(I_k, t_i))/|Z_k|$.

Definition 6 *We define $\widehat{P}(A)$ to be the estimated profit assuming that the items in set A are selected:*

$$\widehat{P}(A) = \sum_{i=1}^{m}\sum_{I_a \in t'_i} prof(I_a, t_i)(1 - conf(I_a \to \diamond d_i))$$

The formula $\widehat{P}(A)$ is equal to that used in Definition 3. If $A = S$, where S is the output selection set, $\widehat{P}(A)$ is equal to the final output estimated profit.

c_i	individual count of item I_i
p_i	average profits of item I_i
b_i	Benefit of item I_i
S_i	estimation set for item I_i
$e_{i,j}$	Estimated value of item I_j from item I_i; $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$

6.1 Overall Framework

In the algorithm MPIS_Alg, there are two phases - (1) *Preparation Phase* and (2) *Main Phase*. In the Preparation Phase, the frequency and the individual count of each item and the size 2 itemsets are returned. In the Main Phase, the *benefit* of each item is evaluated. Initially the result set contains all items, a number of iterative steps of removing items with minimum estimated benefit proceeds until J items remains.

Preparation Phase

1. count the number of occurrences of each item, $n_1, n_2, ..., n_n$.
   obtain the individual count for each item, $c_1, c_2, ..., c_n$
2. generate all size 2 itemsets, with their counts.

Main Phase

1. **Estimation Set Creation -**

   In this step, the estimation sets for all items, $S_1, S_2, ..., S_n$ are computed.
   For each item I_i, calculate the **estimated value** of item I_j from item I_i: $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$, where $support(I_i, I_j)$ is the support of the itemset $\{I_i, I_j\}$. Among these I_j items, choose $J-1$ items with the highest estimated values. Put these items into the estimation set S_i for I_i.
2. **Item Benefit Calculation -** determine the estimated benefit b_i of each item I_i, $b_i \leftarrow \widehat{P}(S_i \cup \{I_i\})$
3. **Item Selection and Item Benefit Update**

   Let $\mathcal{I}'$ be the set of items that has not been pruned.
   (a) prune an item I_x with a smallest benefit b_x value among the items in $\mathcal{I}'$
   (b) for each remaining item I_i in $\mathcal{I}'$,
   If I_x is in S_i,
      i. remove I_x in the set S_i. Choose the item I_k which has not been selected yet in S_i with the greatest value of $e_{i,k}$. Insert I_k into the set S_i.
      ii. Calculate $b_i \leftarrow \widehat{P}(S_i \cup \{I_i\})$
4. **Iteration -** Repeat Step 3 until J items remain.

6.2 Enhancement Step

We can add a pruning step in between Step 1 and Step 2 in the above to enhance the performance. We call this the **Item Pruning** step and it prunes items with apparently small benefit. The basic idea is to compute both a lowest value and an upper value for the profit of each item. These values are generated by varying the estimated selection set for an item.

1. For each item I_i, calculate L_i and H_i, where
   $L_i = \widehat{P}(\{I_i\})$ and $H_i = \widehat{P}(S_i \cup \{I_i\}) - \widehat{P}(S_i)$
2. Find the J-th largest value (L^J) among all L_j
3. For each I_i, remove item I_i if $H_i < L^J$

L_i is an estimate of the lowest possible profit contributed by I_i; we assume that the selected set contains only I_i. In this case, the cross-selling effect may greatly reduce the profit generated from I_i. H_i is the opposite of L_i; we assume that as many as possible of the items related to I_i are selected in S_i. H_i is equal to the profit gain from adding item I_i to set S_i. Hence the cross-selling effect will diminish the profit to a much lesser extent.

For I_i, the initial profit is zero in $\widehat{P}(S_i)$, since it is not in S_i. After I_i is included in S_i, the profit from I_i should be greater than or equal to the profit that I_i generates when it is the only item selected, because of less cross-selling profit loss factors. Hence H_i and L_i satisfy the following property:

Lemma 1 $H_i \geq L_i$.

Item I_i is pruned if H_i is smaller than the values of L_j of the first J items which have the highest values of L_j. The rationale is that I_i has little chance of contributing more profit than other items.

When this pruning step is inserted, Step 2 in the Main Phase above will not need to compute the estimated benefit for all items, only the items that remain (are not pruned) will be considered when computing the estimated benefits. However, the set S_i would be updated if it initially contains items that are pruned.

Our experiments show that this step is very effective. In the IBM synthetic data set, there are 1000 items. If the number of items to be selected, J, is 500, there are only 881 remaining items after the pruning step. Note that if J is large, this enhancement step can be skipped.

6.3 Implementation Details

Here we describe how some of the steps are implemented. Some sophisticated mechanisms such as the FP-tree techniques are employed to make the computation efficient even with a vast amount of items and transactions.

6.3.1 Reading transactions from an FP-tree

In a number of cases, the transactions in the database are examined for computation; for example, in the preparation step, when we generate all size 2 itemsets; in the item benefit calculation, to determine the profit of a selection. If we actually scan the given database, which typically contains one record for each transaction, the computation will be very costly. Here we make use of the FP-tree structure [10]. We construct an FP-tree $\mathcal{FPT}$ once for all transactions, setting the support threshold to zero, and recording the occurrence count of itemsets at each tree node. With the zero threshold, $\mathcal{FPT}$ retains all information in the given set of transactions. Then we can traverse $\mathcal{FPT}$ instead of scanning the original database. The advantage of $\mathcal{FPT}$ is that it forms a single path for transactions with repeated patterns. In many applications, there exist many transactions with the same pattern, especially when the

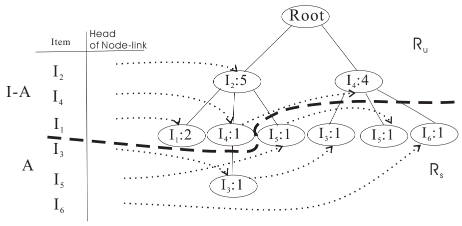

Figure 1. An example of an FP-MPIS-tree

number of transactions is large. These repeated patterns are processed only once with $\mathcal{FPT}$. From our experiments this mechanism can greatly reduce the overall running time.

6.3.2 Calculating Profit with the FP-MPIS-tree

In the definition of the profit of an item selection $\widehat{P}(A)$ (see Definition 6), we need to compute the number of transactions containing some selected items I_a and any item in set d_i (the value of $g(I_a, d_i)$), where $I_a \in A$ and $d_i \subseteq I - A$. This is computed for many selections for each iteration, hence the efficiency is important. For this task, we use the FP-MPIS-tree data structure.

In the FP-MPIS-tree, we divide the items into two sets, $I - A$ and A. Set A corresponds to items selected while $I - A$ contains those not selected. The items in set $I - A$ are inserted into FP-MPIS-tree near to the root. Similar to the FP-tree, the ordering of items in each set in the FP-MPIS-tree is based on the frequencies of items. An example is shown in Figure 1. In the figure, the set of selected items is $A = \{I_3, I_5, I_6\}$ and the set of unselected items is $I - A = \{I_1, I_2, I_4\}$.

To compute $g(I_a, d)$, we first look up the horizontal linked list (dotted links in Figure 1) of item I_a in the FP-MPIS-tree. For each node Q in the linked list, we call the function parseFPTree(Q, d). The function returns a count, we add up all the counts returned from the nodes Q and it is the value of $g(I_a, d)$.

Function parseFPTree(N, d) computes the number of transactions containing item I_a and at least one item in d in the path from root of FP-tree to N. Starting from the node N, we traverse the tree upwards towards the root of the FP-tree until we find a node M containing one element in set d or we hit the root node. If M exists, the count stored in node N is returned. The call of function parseFPTree(N, d) is quite efficient as we do not need to traverse downwards from node N. This is because all nodes below node N are selected items, no item in d will be found below N.

A further refinement for the FP-MPIS-tree is to insert only transactions that contain both selected and non-selected items. For transactions with only selected items, the profit for each selected item is simply given. For transactions with only non-selected item, the profit contribution will be zero. This refinement can greatly reduce the size of the FP-MPIS-tree. Note also that the FP-MPIS-tree is built from the FP-tree $\mathcal{FPT}$ and not from the original database.

6.3.3 Item Benefit Update

In each iteration, after we remove item I_x, we need to check the selection S_i for each item I_i in $\mathcal{I}'$. If S_i contains item I_x, it should be updated because item I_x has been removed, also a new item I_k will be selected to be included into S_i. As S_i is changed, the benefits b_i also have to be updated.

Let $S_i' \cup \{I_x\}$ be the selection before we remove item I_x while $S_i' \cup \{I_k\}$ be the selection after we have removed item I_x and added item I_k in the selection S_i. We can do the item benefit update by scanning only those transactions $\mathcal{T}$ containing at least one of item I_x and item I_k. Let $\widehat{P}'(A, \mathcal{T})$ be the profit of the item selection A generated by transactions in $\mathcal{T}$. The item benefit is updated: $b_i \leftarrow b_i + \widehat{P}'(S_i' \cup \{I_k\}, \mathcal{T}) - \widehat{P}'(S_i' \cup \{I_x\}, \mathcal{T})$. The computation of $\widehat{P}'(A, \mathcal{T})$ can be done in a similar manner as $\widehat{P}(A)$ but $\widehat{P}'(A, \mathcal{T})$ considers only transactions $\mathcal{T}$, instead of all transactions. As there are fewer transactions in $\mathcal{T}$ compared to the whole database, the update can be done very efficiently.

7 Empirical Study

We use the Pentium IV 1.5 GHz PC to conduct our experiment. Frontline System Solver is used to solve the QP problem. All algorithms other than QP are implemented in C/C++. The **profitability** is in terms of the percentage of the total profit in the data set. We compare our methods with HAP and the naive approach. The naive approach simply calculates the profits generated by each item for all transactions and select the J items with the greatest profits. Several synthetic data sets and a real data set are to be tested in our experiments.

We have tried a number of quadratic programming tools, including LINDO, TOMLAB, GAMS, BARON, OPTRIS, WSAT, Frontline System Solver, MOSEK and OPBDP. We choose Frontline System Solver (Premium Solver - Premium Solver Platform) [1] because it performs the best out of these solvers.

7.1 Synthetic Data Set

In our experiment, we use the IBM synthetic data generator in [2] to generate the data set with the following parameters (same as the parameters of [26]): 1,000 items, 10,000 transactions, 10 items per transaction on average, and 4 items per frequent itemset on average. The price distribution can be approximated by a lognormal distribution, as pointed out in [15]. We use the same settings as [26]. That is, 10% of items have the low profit range between \$0.1 and \$1, 80% of items have the medium profit range between \$1 and \$5, and 10% of items have the high profit range between \$5 and \$10.

7.2 Real Data Set

The real data set is obtained from a large drug store in Canada over a period of 3 month. In this data set, there are 26,128 items and 193,995 transactions. On average, each transaction contains 2.86 items. About 40% of the transactions contain a single item, 22% contain 2 items, 13% contain 3 items, the percentages for increasing sizes decrease smoothly, and there are about 3% of the transactions with more than 10 items. The greatest transaction size is 88 items. In this data set, the profit distribution of items is shown in the following table.

Profit Range	Proportion	Profit Range	Proportion
$0-$0.1	2.03%	$5-$10	10.43%
$0.1-$1	25.05%	$10-$100	7.75%
$1-$5	54.59%	$100-$400	0.15%

7.3 Results for Synthetic Data

In the first experiment, we have the same setup as in [26] but the profit follows lognormal distribution. The result is shown in Figure 2. In the figure, it is noted that the profitability lines for MPIS_Alg, QP and HAP are overlapping and the execution-time line for HAP is slightly greater than that for naive.

For profitability, we observe that, for the data set, the naive approach gives the lowest profitability among all algorithms. This is because the naive approach does not consider any cross-selling effect. Naturally the profitabilities of all algorithms increase when the number of items selected increases.

From the graph of the execution time against the selection size, the execution time of MPIS_Alg increases from 0% selection, reaching a maximum when about half the items are selected, and then decreases afterwards. Here the execution time depends on two factors. The first factor is related to the complexity of each iteration. If there are more items to be selected, the benefit calculation is more complex and updates to the benefit are more likely. The initial increase is related to the first factor. The second factor is related to the number of iterations in the algorithm. When J, the number of items selected, increases, the number of items to be removed in the iteration step decreases. Thus, the number of iterations decreases if J is large compared with n. The first factor is dominant when the selection is below 50% but the second factor becomes dominant when the selection is larger than 50%.

The quadratic programming approach (QP) used in the chosen Solver uses a variant of the Simplex method to determine a feasible region and then uses the methods described in [13] to find the solution. As the approach uses an iterative step based on the current state to determine the next step, the execution time is quite fluctuating as the execution time is mainly dependent on the problem (or which state the algorithm is in).

HAP is an iterative approach to find the authority weight of each item. The formula for the update of the authority weight is in the form $a = Ma$, where a is a vector of dimension n representing the authority weight of n items and M is an $n \times n$ matrix used in HAP to update the authority weight. In our experiment, we observed that the authority weights converge rapidly.

QP takes the longest execution time compared with other algorithms. Naive gives the shortest execution time as there are only simple operations. HAP gives the second shortest execution time for this small synthetic data set. We note that the number of iterations involved are quite small. MPIS_Alg has the second greatest execution time, but it scales much better with increasing number of items, where it can outform HAP many folds (see the next subsection).

7.4 Results for Real Data Set

With the drug store data set, we have conducted similar experiments as with the synthetic data. However, the Quadratic Programming (QP) Solver [1] does not handle more than 2000 variables. In the real data set, there are 26,128 variables (i.e. items), hence it is not possible to experiment with our QP tool.

The results of the experiments are shown in Figure 3. In the results, HAP gives the lowest profitability. The reason is as follows. In the dataset, there are some items with zero-profit and high authority weight (described in Section 3), yielding a low estimated total profit of the item selection. Suppose item I_i has zero profit, it is likely a good buy and hence can lead to high support. If there are sufficient number of purchases of other item, says item I_j, with item I_i and if item I_i usually occur in the transactions containing item I_j, the confidence of the rule $I_j \to I_i$ is quite high. This creates a high authority weight for item I_i. Items like I_i would lead to smaller profitability for HAP.

MPIS_Alg gives a greater profitability than naive approach in the real data set. For instance, if $J = 20,902$, the difference in profitabilities between these two approaches is 2%. In the real data set, the total profit is equal to $1,006,970. The difference in 2% profitability corresponds to $20,139.4, which is a significant value. If J=8709, the difference in profitabilities between the two approaches is about 8%, which corresponds to $80,557.6.

On average, the execution time of HAP is 6.5 times slower than MPIS_Alg when the problem size is large. HAP requires 6 days to find the item selection while MPIS_Alg requires about 1 day to find the solution. Since item selection is typically performed once in a while, only when a store should update the types of products it carries, the execution time is acceptable. Though the naive method is much faster, the profit gain consideration from MPIS_Alg would make it the better choice for an application.

The execution time of HAP increases significantly when the number of items increases compared with MPIS_Alg. In HAP, a *cross-selling matrix* B is updated iteratively. The matrix is of the order $n \times n$. For the real data set $n = 26,128$, and n^2 will be very large. Let a be the $n \times 1$ vector representing the authority weight of each item. In HAP, there is a process to update Ma iteratively, where $M = B^T B$. This matrix multiplication of matrix M with vector a is highly costly. Let us consider the memory required for matrix M. If double data type (8 bytes) is used for storage of each entry, then the matrix requires a memory size of about 5.08GB. If float data type (4 bytes) is required, then about 2.5GB memory is required. This large matrix cannot fit into the physical memory, causing a lot of disk accesses for virtual memory. Since the matrix M is sparse, a hash data structure can be used, so that only non-zero entries are stored. We have adopted the hash structure for the real data set, and fouud that less than 5MB memory is needed. Our results in Figure 3 are based on this enhanced hashing approach. However, the computation with this reduced size is still very massive.

We have also tried other sets of experiments where not all the items are considered but only those above a minimum support threshold of 0.05% or 0.1% are considered. However, the resulting profitabilities are much lower than those shown in Figure 3. For instance, if J = 500 and min-support = 0.05%, the profitability of naive and MPIS_Alg is about 1.3%. But, if all items are considered, the profitability of those approaches is about 25%. This is explained by the existence of items that generate high profits but which are not purchased frequently enough to be counted given the support thresholds.

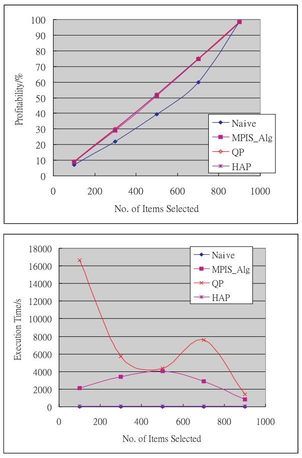

Figure 2. The Synthetic Data Set

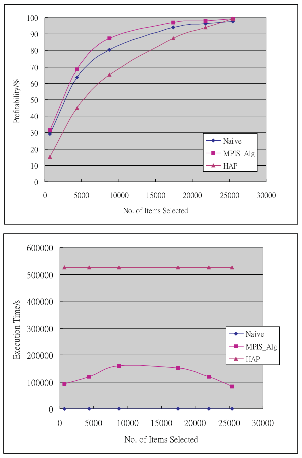

Figure 3. The drug store data set

8 Conclusion

One of the applications of association rule - the maximal-profit item selection problem with cross-selling effect (MPIS) is discussed in this paper. We propose a modeling by the loss rule, which is used in the formulation of the total profit of the item selection. We propose both a quadratic programming approach and a heuristical approach to solve the MPIS problem. We show by experiments that these methods are efficient and highly effective. We believe that much future work can be done. The heuristical method can be enhanced with known methodologies such as hill climbing. Expert knowledge can be included in the methods, and the definition of the problem can be changed in different ways to reflect different user environments.

Acknowledgements We would like to thank M.Y. Su for his generous help in providing the source codes for the HAP solution and his other advices. We also thank Ping Quin of DBMiner Technology for providing the real dataset. This research is supported by the Hong Kong RGC Earmarked Grant UGC REF.CUHK 4179/01E.

References

[1] Frontline systems solver, http://www.solver.com/.

[2] R. Agrawal. Ibm synthetic data generator, http://www.almaden.ibm.com/cs/quest/syndata.html.

[3] R. Agrawal, T. Imilienski, and Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[5] J.E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. In *Technical report, the Management School, Imperial College, London*, Dec 1998.

[6] T. Blischok. Every transaction tells a story. In *Chain Store Age Executive with Shopping Center Age 71 (3)*, pages 50–57, 1995.

[7] 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. In *SIGKDD*, 2000.

[8] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *SIGKDD*, 1999.

[9] M.R. Garey and D.S. Johnson. Computers and intractability: A guide to the theory of np-completeness. In *Freeman*, 1979.

[10] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, 2000.

[11] S. Hedberg. The data gold rush. In *BYTE, October*, pages 83–99, 1995.

[12] Hiller and Lieberman. Introduction to operations research. In *McGraw Hill, Seventh Edition*, 2001.

[13] B. V. Hohenbalken. A finite algorithm to maximize certain pseudoconcave functions on polytopes. In *Mathematical Programming 8*, 1975.

[14] R. Horst, P. M. Pardalos, and N. V. Thoai. Introduction to global optimization. In *Kluwer Academic Publishers, Second Edition*, 2000.

[15] J. C. Hull. Options, futures, and other derivatives. In *Prentice Hall International, Inc. (3rd Edition)*, 1997.

[16] L.D. Iasemidis, P. Pardalos, J.C. Sackellares, and D.S. Shiau. Quadratic binary programming and dynamical system approach to determine the predictability of epileptic seizures. In *Journal of Combinatorial Optimization, Kluwer Academic,*, pages 9–26, 2001.

[17] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. In *Knowledge Discovery Journal*, 1998.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 1998, Also in JACM 46:5, 1999.

[19] S. J. Leon. Linear algebra with applications. In *Prentice Hall, Fifth Edition*, 1998.

[20] J. Luo, K. R. Pattipati, and P.K. Willett. A sub-optimal soft decision pda method for binary quadratic programming. In *Proc. of the IEEE Systems, Man, and Cybernetics Conference*, 2001.

[21] H. Mannila. Methods and problems in data mining. In *Proc. of Int. Conf. on Database Theory*, 1997.

[22] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *KDD*, 1994.

[23] V. Safronov and M. Parashar. Optimizing web servers using page rank prefetching for clustered accesses. In *World Wide Web: Internet and Web Information Systems Volume 5, Number 1*, 2002.

[24] S. Sahni. Computationally related problems. In *SIAM J. Comput. 3*, pages 262–279, 1974.

[25] J. Ullman. Lecture notes on searching the web, http://www-db.stanford.edu/ ullman/mining/mining.html.

[26] K. Wang and M.Y. Su. Item selection by "hub-authority" profit ranking. In *SIGKDD*, 2002.