Data Mining and Knowledge Discovery, 11, 81–112, 2005

Data Mining for Inventory Item Selection with Cross-Selling Considerations

RAYMOND CHI-WING WONG cwwong@cse.cuhk.edu.hk
ADA WAI-CHEE FU adafu@cse.cuhk.edu.hk
Department of Computer Science and Engineering, The Chinese University of Hong Kong

KE WANG wangk@cs.sfu.ca
Department of Computer Science, Simon Fraser University, Canada

Editor: Geoff Webb

Received ; Revised

Pub: Pls. provide dates.

Abstract. Association rule mining, studied for over ten years in the literature of data mining, aims to help enterprises with sophisticated decision making, but the resulting rules typically cannot be directly applied and require further processing. In this paper, we propose a method for actionable recommendations from itemset analysis and investigate an application of the concepts of association rules—maximal-profit item selection with cross-selling effect (MPIS). This problem is about choosing a subset of items which can give the maximal profit with the consideration of cross-selling effect. A simple approach to this problem is shown to be NP-hard. A new approach is proposed with consideration of the *loss rule*—a rule similar to the association rule—to model the cross-selling effect. We show that MPIS can be approximated by a quadratic programming problem. We also propose a greedy approach and a genetic algorithm to deal with this problem. Experiments are conducted, which show that our proposed approaches are highly effective and efficient.

Keywords: data mining algorithm, cross-selling, item selection, association rule, quadratic programming, genetic algorithm

1. Introduction

Data mining technology, as one of the strong pillars in CRM (Customer Relationship Management), plays a vital role in business expansion. By knowing customer behavior based on past records, increased profits from cross-selling and other business strategies can be achieved. The behaviour in terms of sales transactions is considered significant (Blischok, 1995). Data mining on such transactions is 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 instance, Hedberg (1995) reports that the American supermarket chain Wal-Mart keeps about 20 million sales transactions per day. This vast amount of data requires sophisticated methods in the analysis.

Decision making in the business sector is considered one of the critical tasks in data mining. There is a study in Kleinberg et al. (1998) on the utility of data mining for such problems, which proposes a framework based on optimization for the evaluation of data mining operations. The general decision making problem is considered as a maximization problem as follows: $\max_{x \in \mathcal{D}} \sum_{i \in \mathcal{C}} \mathcal{G}(x, y_i)$ where $\mathcal{D}$ is the set of all possible decisions in the domain problem (e.g. inventory control), $\mathcal{C}$ is the set of customers, y_i is the data or history we have on customer i, and $\mathcal{G}(x, y_i)$ is the utility (benefit) from a decision x and y_i. However, with a further investigation of such decision problems, we notice that we are in fact dealing with a maximization problem of the following form:

$$\max_{x \in \mathcal{D}} \mathcal{G}(x, Y) \tag{1}$$

where Y is the set of all y_i, or the set of data and history collected about all customers. The above form is more appropriate when there are correlations among the behaviours of customers (e.g. cross-selling - the purchase of one item is related to the purchase of another item), or when there are interactions among the customers themselves (e.g. viral marketing, or marketing by word-of-month among customers). Such effects can be uncovered only by extracting patterns in Y and we cannot determine $\mathcal{G}()$ based on each single customer alone.

The problem under investigation in this paper is about such a problem: optimal product selection (Brijs et al., 1999, 2000; Wang and Su, 2002; Wong et al., 2003). The problem is to find a subset of the products to be discontinued so that the profit can be maximized. The formulation of the problem considers the important factor of cross-selling which is the influence of some products on the sales of other products. The cross-selling factor is embedded into the calculation of the maximum profit gain from a decision. This factor can be obtained from an analysis of the history of transactions kept from previous sales which corresponds to the set Y in formulation (1). [1]

Association rule mining (Agrawal et al., 1993) 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. Recently, some researchers (Brijs et al., 1999) 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.

We study the problem of Maximal-Profit Item Selection with cross-selling effect (MPIS) (Wong et al., 2003). MPIS is the problem of finding a set of J items with the consideration of the cross-selling effect such that the total profit from the item selection is maximized, where J is an input parameter. We assume that a history of transaction records is given for uncovering customer behaviours. 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 $I \rightarrow \Diamond d$, where I is an item and d is a set of items, and $\Diamond d$ means the purchase of any items in d. This loss rule helps to estimate the loss in profit of item I if all items in d are missing after the selection. The rule corresponds to the cross-selling effect between I and d.

To tackle this problem, we propose a quadratic programming method (QP), a heuristics method called MPIS_Alg and a genetic algorithm (GA) approach. Some preliminary results for QP and MPIS_Alg are shown in Wong et al. (2003). 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 the experiment, the profitabilities of these two proposed algorithms are greater than that of naive approach for all data sets. On average, the profitability of QP and MPIS_Alg is 1.33 times higher than the naive approach for the synthetic data set. Besides, when the number of items is large (as in the drugstore data set), the execution time of the best previous method, HAP (Wang and Su, 2002), is 6.5 times slower than MPIS_Alg. This shows that the MPIS_Alg is highly effective and efficient. GA can be viewed as an optimization method which encodes a possible solution of a problem in a chromosome-like data structure. The GA approach creates a number of chromosomes in a population and finds a solution by rearranging the chromosomes in the population. From our experiments, GA also gives high profitabilities and the efficiency is comparable to MPIS_Alg. For some datasets in our experiments, GA out-performs MPIS_Alg in both profitabilities and efficiency.

2. Problem definition

Item selection problem was first addressed by Brijs et al. (1999), (2000) and (Wang and Su, 2002). 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 Wang and Su (2002). Given a data set with m transactions, $t_1, t_2, \ldots, t_m$, and n items, $I_1, I_2, \ldots, I_n$, let $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$. The profit of item $I \in \mathcal{I}$ in transaction t_i is given by $prof(I, t_i)$.[2] Let $S \subset \mathcal{I}$ 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.

$$t_i' = t_i \cap S, \quad d_i = t_i - t_i' \tag{2}$$

t_i' represents a set of items selected in the transaction t_i while d_i represents a set of items not selected in t_i. Suppose a subset S of items is chosen. In other words, some items in $I_1, \ldots, I_n$ will be eliminated. The transactions $t_1, \ldots, t_m$ 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 anticipated profit $prof(I, t_i)$ of items in future transactions can be affected if some items are removed from the stock. This is caused by the cross-selling factor. Here, we assume the customers do not substitute for items that are not available.[3] The cross-selling factor is modeled by $csfactor(D, T')$, where D is a set of unselected items and T' is a set of selected items, and $0 \leq csfactor(D, T') \leq 1$. $csfactor(D, T')$, is the fraction of the profit of the items in T' that will be lost in a transaction if the items in D are not available and the items in T' are available. Note that the cross-selling factor can be determined in different ways. One way is by the domain experts. Here we advocate an alternative to derive this factor from the given history of transactions.

We define the total profit of item selection based on the given set of transactions. Roughly speaking, this is given by the original profits of the transactions subtracting the profit loss due to the items removed after making the selection. Assume customers react in the same way if one or more of their favourite products in their choice set is removed. Thus, we can handle each transaction in the same way. For each transaction, we calculate the profit remained of the selected items after the selection. Then, we sum up the profit remained for all the transactions. The sum is the total profit of the item selection of our problem.

A general formula of the total profit of item selection is shown as follows. Let $prof(t', t)$ be the total profit of the items in t' in transaction t. That is, $prof(t', t) = \sum_{I \in t'} prof(I, t)$.

Definition 1 (Total profit of item selection). The total profit of an item selection S is given by

$$P(S) = \sum_{i=1}^{m} prof(t_i', t_i)(1 - csfactor(d_i, t_i'))$$

We select a set of J items so that the total profit is the maximal among all such sets.

For the special cases when all items in transaction t_i are selected in the set S, 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. t_i' is an empty set, and the profit of transaction t_i becomes zero after we have made the selection. This is because, as the items in the transaction would no longer be in the stock, the transaction should not appear after we have made the selection.

For simplicity, we modify Definition 1 to Definition 2 by assuming the independence of csfactors in the same transaction. For instance, there is a transaction $t_i = \{I_1, I_2, I_3, I_4\}$ and I_1 and I_2 are removed. In Definition 1, the total profit is modeled as $prof(\{I_3, I_4\}, t_i)(1 - csfactor(\{I_1, I_2\}, \{I_3, I_4\}))$. In Definition 2, we simplify it by the summation of each item. That is,

$$prof(\{I_3\}, t_i)(1 - csfactor(\{I_1, I_2\}, \{I_3\})) + prof(\{I_4\}, t_i)(1 - csfactor(\{I_1, I_2\}, \{I_4\}))$$

Later we shall show that the selection problem for this simplified definition is already very hard. For clarify, we denote $csfactor(d_i, \{I\})$ by $csfactor(d_i, I)$. The simplified form of the total profit of item selection is shown as follows.

Definition 2 (Total profit of item selection). The total profit of an item selection S is given by

$$P(S) = \sum_{i=1}^{m} \sum_{I \in t_i'} prof(I, t_i)(1 - csfactor(d_i, I))$$

Table 1. An example.

Monitor	Keyboard	Telephone
1	1	0
1	1	0
1	1	0
0	0	1
0	0	1
0	0	1
1	1	1

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 $P(S)$.

Example: Suppose a shop carries office equipments of monitors, keyboards, and telephones, with profits of \$1000 K, \$100 K, \$300 K, respectively (See Table 1). Suppose now the shop decides to remove one of the 3 items from its stock, the question is which two should we choose to keep. If we simply examine the profits, we may choose to keep monitors and telephones, so that the total profit is \$1300 K. However, we know that there is strong cross-selling effect between monitor and keyboard. We can get this information from a sales record such as the following table of transactions, where each row records a transaction, with the value of 1 meaning a purchase. If the shop stops carrying keyboard, the customers of monitor may choose to shop elsewhere to get both items. The profit from monitor may drop greatly, and we may be left with profit of \$300 K from telephones. If we choose to keep both monitors and keyboards, then the profit can be expected to be \$1100 K which is higher. MPIS with the profit as defined in Definition 2 will give us the desired solution. Suppose we choose monitor and telephone. For a transaction t_i, with the purchase of monitor and keyboard, $d_i = \{keyboard\}$, $csfactor(d_i, monitor) = csfactor(\{keyboard\}, monitor) \rightarrow 1$, and $prof(monitor, t_i)(1 - csfactor(d_i, monitor)) \rightarrow 0$. This example illustrates the importance of the consideration of cross-selling factor in the profit estimation, and the usefulness of our definition for the determination of a selection.

The MPIS 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 $P(S) \geq B$?

To understand the difficulty of the problem, we consider the very simple version where $csfactor(d_i, I) = 1$ 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.

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

Proof: We shall transform the problem of CLIQUE to the MPIS problem. CLIQUE (Garey and Johnson, 1979) is an NP-complete problem defined as follows:

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 the above 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, $i = 1, 2, \ldots, |E|$, and I_j is an item in t_i. It is easy to see that this transformation can be constructed in polynomial time.

Consider the case where K vertices form a complete graph in G. Let the set of corresponding items for the clique be C. That is, in the MPIS problem, each corresponding item should co-exist with the other $K-1$ items in C in some transactions. Let this set of transactions be T_c. It is easy to see that if the set of items in the clique are chosen in the set S then the profit will be B. T_c is the set of the only transactions that contribute to the profit calculation. In a transaction not in T_c, if an item I_i in C exists, it co-exists with another item which is not in C. Since the cs factor is 1, the profit from item I_i in the transaction is 0. Conversely if S is a set in the MPIS problem with benefit above B, then there must exist a complete graph with at least K vertices in the CLIQUE problem. Since CLIQUE is an NP-complete problem, the above MPIS problem is NP-hard. □

3. Related work

Suppose we are given a set I of items, and a set of transactions. Each transaction is a subset of $\mathcal{I}$. An *association rule* has the form $X \rightarrow I$, where $X \subseteq \mathcal{I}$ and $I \in \mathcal{I}$; the *support* of such a rule is the fraction of transactions containing all items in X and item I; the confidence for the rule is the fraction of the transactions containing all items in set X that also contain item I. The problem is to find all rules with sufficient support and confidence. Some of the earlier work includes Mannila et al. (1994), Agrawal and Srikant (1994) and Mannila (1997). Note that in all such data mining research, it is assumed that some patterns can be mined from the history of transactions and these patterns will persist in the future, so that they can help to predict the customer behaviour for decision making.

The maximal-profit item selection problem has been studied recently in Brijs et al. (1999), (2000) and Wang and Su (2002) (PROFSET and HAP). In PROFSET, the cross-selling effects are modeled 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 frequent itemsets are counted in the total profit. The problem is formulated as binary programming that aims at maximizing the total profit.

A number of drawbacks of the original PROFSET are pointed out in Wang and Su (2002): (1) PROFSET is independent of the strength of the relationship between items (i.e. the level of confidence).[4] (2) PROFSET does not give a relative ranking of the selected items directly from the objective function.[5] (3) The purchase intention is not only illustrated in the maximal frequent itemsets because subsets of maximal frequent itemsets are also related to purchase intention, which are not considered in PROFSET.

HAP (Wang and Su, 2002) applies the "hub-authority" profit ranking approach (Safronov and Parashar, 2002) 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 strength of the link. The HITS algorithm (Kleinberg, 1999) 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 Ullman (2003) can arise. (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 have 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 good solution.

4. Cross selling effect by association rules

So far we have introduced the idea of the problem formulation but we have not specified how to determine the cross-selling effect *csfactor*. In previous work Wang and Su (2002), the concept of association rules is applied to this task. Here we also apply the ideas of association rules for the determination of *csfactor*.

We are going to estimate the possible profit from a given set of transactions. As in previous work on data mining, our assumption is that some patterns corresponding to customer behaviour can be discovered in the data set, and these patterns will remain unchanged in the future. Therefore, if all items are selected, the profit will remain the same since the customers will make similar purchases. Suppose we have made a selection S of J items ($J < n$). According to the customer behaviour, the decrease in available items may change the future purchases and similar transactions as the history would probably not occur in the future. In particular, some transaction types may be reduced in size and this leads to profit loss if some items in the transactions are missing.

Consider a transaction t_i in our transaction history. Suppose some item, say I, is selected in S but some items are not selected (i.e. d_i). If there is the customer behaviour that purchasing I always co-occurs with the purchase of at least one element in d_i then it would be unlikely for transaction t_i to exist after the selection of S. This is because t_i contains I and no element in d_i after the selection, which does not follow the pattern of customer behaviour. The profit generated by t_i from I should be removed from our estimated profit. The above customer behaviour can be modeled by the concepts of association rule. We model the cross-selling factor in the total profit of item selection $csfactor(d_i, I)$ by $conf(I \rightarrow \diamond d_i)$, where $\diamond d_i$ is given by the following:

Definition 3. Let $d_i = \{Y_1, Y_2, Y_3, \ldots, Y_q\}$ where Y_i refers to a single item for $i = 1, 2, \ldots, q$, then $\diamond d_i = Y_1 \vee Y_2 \vee Y_3 \vee \ldots \vee Y_q$.

P1: NVI
Data Mining and Knowledge Discovery SJNW243-05-NO00001359 May 9, 2005 21:6

Consider a transaction t_i in our transaction history. Suppose some item, say I, is selected in S but some items are not selected (i.e. d_i). If there is a customer behaviour that purchasing I always co-occurs with the purchase of at least one element in d_i then it would be unlikely for transaction t_i to exist after the selection of S. This is because t_i contains I and no element in d_i after the selection, which does not follow the pattern of customer behaviour. The profit generated by t_i from I should be removed from our estimated profit. The above customer behaviour can be modeled by the concepts of association rule. We model the cross-selling factor in the total profit of item selection *csfactor*(d_i, I) by $conf(I \to \Diamond d_i)$.

The rule $I \to \Diamond d_i$ is called a *loss rule*. The rule $I \to \Diamond d_i$ indicates that from the history, whenever a customer buys the item I, he/she also buys at least one of the items in d_i. Interpreting this as a pattern of customer behaviour, and assuming that the pattern will not change even when some items were removed from the stock[6], if none of the items in d_i are available then the customer also will not purchase I. This is because if the customer still purchases I, without purchasing any item in d_i, then the pattern would be changed. Therefore, the higher the confidence of $I \to \Diamond d_i$, the more likely the profit of I in t_i should not be counted. This is the reasoning behind the above definition.

The confidence of the loss rule $I \to \Diamond d_i$ is defined in a similar manner as for the association rule:

Definition 4. If $d_i \neq \phi$, $conf(I \to \Diamond d_i)$ is equal to

$$\frac{\text{number of transactions containing } I \text{ and any element in } d_i}{\text{number of transactions containing } I}$$

If $d_i = \phi$, $conf(I \to \Diamond d_i)$ is equal to 0.

Note that if $I \in d_i$, then $conf(I \to \Diamond d_i) = 1$.

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

Definition 5 (Total profit of item selection (loss rule based)). The loss rule based total profit of item selection S is given by

$$P(S) = \sum_{i=1}^{m} \sum_{I \in t_i'} prof(I, t_i)(1 - conf(I \to \Diamond d_i))$$

where $t_i' = t_i \cap S$ and $d_i = t_i - t_i'$.

4.1. *On the choice of $I \to \Diamond d_i$*

It may seem that $\Diamond d_i \to I$ also shows the correlation between d_i and I, but it is not applicable here. Let us consider an example of a supermarket. Suppose it is observed that whenever a customer buys tea, milk or cream is usually purchased. However, milk or cream is often purchased without the purchase of any tea. That is, $\{tea\} \rightarrow \Diamond\{milk, cream\}$ has high confidence and $\Diamond\{milk, cream\} \rightarrow \{tea\}$ has low confidence. If the shop discontinues both milk and cream, it can highly affect the sales of tea. We can see that the high confidence of $\{tea\} \rightarrow \Diamond\{milk, cream\}$ will uncover the cross-selling effect. We can also see that the rule $\Diamond\{milk, cream\} \rightarrow \{tea\}$ is not relevant here. In fact, with its low confidence, it would indicate that there is no cross-selling correlation.

Another alternate candidate for the loss rule is $I \rightarrow \wedge(d_i)$, where $\wedge(d_i)$ is the conjunction of all items in d_i. This rule is the implication that the purchase of I will lead to the purchase of all items in d_i. Let us consider another example where the purchase of tea is often accompanied by the purchase of milk. However, the purchase of tea has not much relation to the purchase of pencil. Therefore, the confidence of $\{tea\} \rightarrow \{milk \wedge pencil\}$ is low while the confidence of $\{tea\} \rightarrow \{milk \vee pencil\}$ is high. It is clear that if we discontinue milk and pencil, then the sales of tea will be affected. However, if the rule of $\{tea\} \rightarrow \{milk \wedge pencil\}$ is used, then the expected effect on a transaction with tea, milk and pencil will not be achieved. Or suppose people usually have their tea with either milk or cream. The confidence of $\{tea\} \rightarrow \{milk \wedge cream\}$ is low while that of $\{tea\} \rightarrow \{milk \vee cream\}$ is high. If the rule of $\{tea\} \rightarrow \{milk \wedge cream\}$ is used, then we would not get the expected effect on a transaction with tea, milk and cream.

5. Quadratic programming

Mathematical programming (or constrained optimization), and its most popular special form, linear programming (LP), has found practical applications in the optimal allocation of scarce resources. The method has been applied for optimization problems in many companies and has saved millions of dollars in their operation (Hiller and Lieberman, 2001). The petroleum industry was an early intensive user of LP for solving fuel blending problems. 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 LP, 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 how the problem can be approximated by a quadratic programming problem. Let $s = (s_1, s_2, \ldots, s_n)^T$ 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 J items are to be selected.

With some overloading of the term t_i, we say that $t_i = (t_{i1}, t_{i2}, \ldots, t_{in})^T$ 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 $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, $t'_{ij} = t_{ij} \times s_j$ 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
$\lvert I_{i1}, \ldots I_{ij} \rvert$	number of transactions containing $\{I_{i1}, \ldots, I_{ij}\}$

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 $g(I_a, d_i)$.

Definition 6. Let $d_l \subset \mathcal{I}$, $d_l = \{Y_1, Y_2, \ldots, Y_q\}$ and $I_x \notin d_l$, where Y_i refers to a single item for $i = 1, 2, \ldots, q$.

$$\begin{aligned} g(I_x, d_l) = &\sum_{1 \le i \le q} |I_x Y_i| - \sum_{1 \le i < j \le q} |I_x Y_i Y_j| \\ &+ \sum_{1 \le i < j < k \le q} |I_x Y_i Y_j Y_k| - \ldots + (-1)^{n+1} |I_x Y_1 Y_2 \cdots Y_q| \end{aligned}$$

where $|I_x Y_i Y_j \ldots|$ is the number of transactions containing the items $I_x, Y_i, Y_j, \cdots$

We have

$$\begin{aligned} conf(I_j \to \Diamond d_i) &= \frac{\text{no. of trans. containing } I_j \text{ and at least one item in } d_i}{\text{no. of trans. containing item } I_j} \\ &= \frac{g(I_j, d_i)}{\text{no. of transactions containing item } I_j} \\ &= \frac{1}{\text{no. of transactions containing item } I_j} \\ &\quad \times \Bigg[\sum_{1 \le k \le n} |I_j I_k| \times d_{ik} - \sum_{1 \le k < k' \le n} |I_j I_k I_{k'}| \times d_{ik} d_{ik'} \\ &\quad + \sum_{1 \le k < k' < k'' \le n} |I_j I_k I_{k'} I_{k''}| \times d_{ik} d_{ik'} d_{ik''} - \cdots \\ &\quad + (-1)^{n+1} |I_j I_1 I_2 \ldots I_n| \times d_{i1} d_{i2} \ldots d_{in} \Bigg] \quad \text{(by Def. 5)} \\ &\approx \min\left(\frac{\sum_{1 \le k \le n} |I_j I_k| \times d_{ik}}{\text{no. of transactions containing item } I_j}, 1 \right) \\ &= \min\left(\frac{1}{n_j} \sum_{k=1}^{n} d_{ik} n_{jk}, 1 \right) \end{aligned}$$

The reason 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 $|I_j I_k I_l|$ is typically much smaller than $|I_j I_k|$, etc. From this approximation we can deduce the following theorem.

Theorem 2. *Given Observation* 1, *the total profit of item selection can be approximated by the following quadratic form.* $P = f^T s + \frac{1}{2} s^T H s$, *where* f *is a vector of size* n *and* H *is an* n *by* n *matrix.* $f = (f_j | f_j = \sum_{i=1}^{m} t_{ij}\, prof(I_j, t_i)(1 - \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk}) n)^T$ *for* $j = 1, 2, \ldots, n$ *and* $H = (h_{jk} | h_{jk} = \frac{2 n_{jk}}{n_j} \sum_{i=1}^{m} t_{ij}\, prof(I_j, t_i) t_{ik}$ *for* $j, k = 1, 2, \ldots, n)$.

The proof of Theorem 2 can be found in Appendix 10.

Corollary 1. *P can be approximated by $P' = f^T s + \frac{1}{2} s^T Q s$ where Q is a symmetric n by n matrix.*

The corollary follows because $P = f^T s + \frac{1}{2} s^T H s = f^T s + \frac{1}{2} s^T Q s$ where $Q = (q_{ij})$ and $q_{ij} = \frac{1}{2}(h_{ij} + h_{ji})$ for all $i, j = 1, 2, \ldots, n$. 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 $\sum_i s_i = J$:

$$\text{Maximize} \quad P' = f^T s + \frac{1}{2} s^T Q s$$
$$\text{such that} \quad \sum_{i=1}^{n} s_i = J, \text{ and } \quad s_i = 0 \text{ or } \quad s_i = 1 \text{ for } \quad i = 1, 2, \ldots, n$$

Any 0-1 quadratic programming problem is polynomially reducible to an unconstrained binary quadratic programming problem (Iasemidis et al., 2001). An unconstrained binary quadratic programming problem can be transformed to a binary linear programming problem (zero-one linear programming) (Beasley, 1998). More related properties can be found in Luo et al. (2001) and Horst et al. (2000). Zero-one linear programming and quadratic programming are known to be NP-complete (Sahni, 1974). 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 8.

The above quadratic programming model can also be enhanced by pushing additional constraints. Typically, retailers will have the need to impose several additional restrictions related to their retail domain knowledge. For instance, a retailer will only accept the decisions from a product selection model in so far that the assortment (after selection) offers sufficient variety (assortment width) and alternatives (assortment depth).[7]

6. Algorithm MPIS_Alg

As quadratic programming may not scale up to large data sizes, we propose next 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.

This algorithm takes account of the following factors: (1) We utilize the exact formula of the profitability in the iterations. This should steer the result better towards the goal of maximal profits compared to other approaches such as HAP (Wang and Su, 2002) that do not directly use the formula. (2) With the "neighborhood" consideration, the item pruning at each iteration usually affects only a minor portion of the set of items and hence introduces 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.

6.1. *Overall framework*

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** of I_k is given by

$$p_k = \frac{\sum_{t_i \in Z_k} prof(I_k, t_i)}{|Z_k|}.$$

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; $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$

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 proceed until J items remains.

In order to estimate the significance of an item in terms of profits, the cross-selling factor should be considered, which depends on a selection set S. Since S does not exist initially, we shall estimate a chosen item set for each item. For each item I_i, we find $J - 1$ "best" items in its "neighborhood", the set of the $J - 1$ items is denoted by S_i. Hence the neighborhood is roughly a set of items that co-occur frequently with I_i with high profits. Set S_i is called the **estimation set** for I_i.

In order to determine set S_i, we calculate the following values for all j: $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$. $support(X)$ refers to the number of transactions containing X in the given database. $e_{i,j}$ is called the **estimated value** of item I_j with respect to item I_i. The first part $p_j \times c_j$ is due to the benefit of the individual profit given by item I_j as this count is not related to other items. The reason is that if an item is highly profitable by itself, then it should have a high chance to be selected. The second part $(p_j + p_i) \times support(I_i, I_j)$ is a profit value given by the items I_i and I_j. If the two items co-occur frequently and have high individual profits, this value will be higher. Note that the computation of this value involves only the supports of size 2 itemsets. It is a simple step in a greedy approach. To determine S_i, we select the items I_j corresponding to the top $e_{i,j}$ values.

After the estimation set S_i is determined for item I_i, we can estimate the resulting profit assuming that items in $S_i \cup \{I_i\}$ are selected. The profit is called the **item benefit** of I_i, denoted by b_i. The item benefit b_i is defined as: $b_i = P(S_i \cup \{I_i\})$, where $P()$ is defined in Definition 5. Hence b_i models the significane of I_i in terms of profits, with the consideration of the cross-selling effect.

Next, items are ranked by their item benefits, and one item I_x with the lowest item benefits is pruned. We shall prune items one at a time until J items are left. 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. Thus, I_x is removed from such S_i, and we have to select the item I_k, which has not yet been selected yet, with the greatest estimate value $e_{i,k}$. I_k will be included into S_i. As S_i is changed, the benefit b_i also has to be updated. However, note that it is possible that S_i does not change for some I_i if the items in S_i are not pruned. In such cases, updating of b_i is not needed. After updating b_i's, the items are ranked again by b_i's. Another item is pruned and this process of benefit updating and item pruning is repeated until only J items remain unpruned. Next we describe both phases in pseudocode.

Preparation phase

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

Main phase

1. **Estimation Set Creation -**
   In this step, the estimation sets for all items, $S_1, S_2, \ldots, S_n$ are computed.
   For each item I_i, calculate the **estimated value** of item I_j from item I_i: $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$. 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, $b_i \leftarrow P(S_i \cup \{I_i\})$

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 $b_i \leftarrow P(S_i \cup \{I_i\})$

4. **Iteration -** Repeat Step 3 until J items remain.

6.2. Enhancement Step

We can add a **Item Pruning** step in between Step 1 and Step 2 in the above to enhance the performance. It prunes items with apparently small benefit. The basic idea is to compute both a lower 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
   $L_i = P(\{I_i\})$ and $H_i = P(S_i \cup \{I_i\}) - P(S_i)$
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 the best estimation set S_i is selected. 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.

Let $\sum_{i=1}^{m} prof(I, t_i)(1-conf(I \rightarrow \diamond d_i))$ be the profit contribution from I in P(S). Since $I_i \notin S_i$, the profit contribution from I_i is zero in $P(S_i)$. Hence $H_i = P(S_i \cup I_i) - P(S_i)$ contains all the profit contribution from I_i in $P(S_i \cup I_i)$. This value should be greater than or equal to the profit that I_i generates when it is the only item selected (L_i), because of less profit loss from cross-selling 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 profits 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 the estimated benefits are computed. 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 with 1000 items, if the number of items to be selected, J, is 500, there are only 881 remaining items after the pruning step, and the resulting profitability is not affected. Note that if J is large, this enhancement step can be skipped.

The implementation Details can be found in Appendix A.

7. Genetic algorithm

Genetic algorithms (GA) are a family of evolution-inspired computational models. A genetic algorithm encodes a possible solution of a problem in a chromosome-like data structure. The algorithm keeps track of a number of chromosomes in a population. By utilizing some reformation operators to these chromosomes in the population, the algorithm can iteratively create a new population in order to preserve critical information in the previous population.

Our genetic algorithm (GA) is based on the steady-state GA described in Syswerda (1989). It also uses a strategy of the weaker-parent replacement described in Mahfoud (1992) and Cavicchio (1970). The details of the genetic algorithm can be found in Mahfoud (1992) and Cavicchio (1970). In our experiment, we adopt the number of generations as the stopping criteria.

Solution representation. We need to represent the solution of MPIS (i.e. which items should be selected). There are n items in total. We need to select a set S of items of size J out of n items such that the total profit P of such an item selection is maximized. We represent the solution as a gene in a binary representation. Each gene is represented by an n-dimensional binary vector, v. Each entry v_i is either 0 or 1 to represent the choice of selection of item I_i for $i = 1, 2, \ldots, n$. If item I_i is selected, v_i will be equal to 1. Otherwise, v_i is equal to 0. In this representation, there is a restriction on the number of 1's such that $\sum_{i=1}^{n} v_i = J$. This means that the total number of selected items is equal to J. For example, if $n = 5$ and $J = 2$, and the selection set S contains I_1 and I_3, then the vector v is equal to (10100).

Population generation. The initial population contains N genes, each generated with a random selection of J items. In binary representation, we just select J positions randomly in the genes and set those positions to 1 and other positions to 0. For example, if $n = 4$ and $J = 2$, and we select I_1 and I_3 randomly, the gene becomes 1010.

Fitness function. Our fitness function is the same as Definition 5. That is, given a set S of selected items of size J, the total profit of item selection P is:

$$P(S) = \sum_{i=1}^{m} \sum_{I_a \in t_i'} prof(I_a, t_i)(1 - conf(I_a \to \Diamond d_i))$$

We can adopt the computation method of $P(S)$ in Algorithm MPIS_Alg (See Appendix A).

P1: NVI
Data Mining and Knowledge Discovery SJNW243-05-NO00001359 May 9, 2005 21:6

Selection for the second parent. In the algorithm, we need to select two parents for crossover. If there are n genes in the population, n pairs of parents are selected. Each gene in the population is the first gene of one of the pairs. The second parent in the pair is chosen from the population in a roulette wheel approach. Each gene in the population will be evaluated with the fitness function. We can rank all genes in the population according to this fitness value. The gene with the smallest fitness value is assigned rank 1; the gene with the second smallest fitness value is assigned rank 2; ....; the gene with the largest fitness value is assigned with the largest rank. After assigning the rank to each gene, we can calculate the selection probability of a gene G according to the ranks as follows:

$$\text{selection\_probability}(G) = \frac{rank(G)}{\sum_x rank(x)}$$

We now introduce the two major operators in GA—crossover and mutation.

7.1. *Crossover*

During crossover, a child will be generated from two parents in the following steps:

1. The child contains the intersection of the two parent genes, C. That is, the child will contain all items that both parents contain. Let k be the number of such items.
2. Usually, the number of selected items generated in Step 1, k, is smaller than the desired number of selected items, J. We need to generate the remaining $J - k$ items chosen from the items in set D, where D is the set containing the items in parent 1 but not in parent 2 and the items in parent 2 but not in parent 1. The method of the selection of $J - k$ items from set D is described as follows.
   Consider a matrix L. Each entry $L_{i,j}$ in the matrix estimates the profit loss of item I_i if item I_j is not selected. We model this entry by

$$L_{i,j} = \mathit{profit}(I_i) \times \frac{\text{no. of transactions containing } I_i \text{ and } I_j}{\text{no. of transactions containing } I_i}$$

   where $profit(I_i)$ is the total profit of I_i for all transactions.
   For each item $I_j \in D$, we will calculate $L_j = \sum_{I_i \in C} L_{i,j}$
   Rank all items $I_j \in D$ according to L_j, with the smallest value ranked 1, the second smallest ranked 2, ... and the greatest ranked the highest.
   Then, we select the remaining items I_j from D in a roulette wheel approach with the following selection probability:

$$\text{selection\_probability}(I_j) = \frac{rank(I_j)}{\sum_{I_k \in D} rank(I_k)}$$

This means that those item I_j which leads to a larger profit loss if item I_j is absent will have a higher probability to be selected. This is because if we do not select item I_j, then there will be a greater profit loss to other items in set C.

7.2. *Mutation*

During mutation, we randomly choose a selected item I_s and an unselected item I_u in the gene. It is noted that, in a gene, a selected item is represented by 1 and an unselected item is represented by 0. Then, we set item I_u to be selected and set item I_s to be unselected. That means, in the gene representation, we change the bit at the position of I_u from 0 to 1 and the bit at the position of I_s from 1 to 0. For example, if the gene is 1010, and I_1 and I_2 are selected as I_s and I_u respectively, then the mutated gene is 0110.

8. Empirical study

In this section, we report on the performance study of the three proposed algorithms over two kinds of data sets—synthetic data set and real data sets. We compare the efficiency of our proposed algorithms—QP, MPIS_Alg and GA—with the previous best method HAP and the naive approach. The naive approach simply calculates the profits generated by each item I for all transactions ($\mathit{profit}(I) = \sum_{j=1}^{m} \mathit{prof}(I, t_j)$) and selects the J items with the greatest profits.

We compare our proposed algorithms with HAP. Note that the problem definition of HAP is a little different from ours. However, as mentioned in Section 3, the algorithm of HAP does not correspond exactly to the problem definition. Instead, HAP relies on a simpler rule involving only two items. The reason why we compare with HAP is that it is the state of the art in the consideration of item selection with cross-selling effect in the data mining literature.

We use the Pentium IV 1.5 GHz PC to conduct all our experiments. 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 have conducted separate experiments for GA in order to study how the parameters of GA affects the profitabilities and execution time and determine a set of GA parameters which can give good quality results without excessive overhead: two children will be generated from each pair of parents, population size is 25 and the number of generations is 20.

For the experiments with quadratic programming, there are some additional considerations. 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) (http://www.solver.com/) because it performs the best out of these solvers.

Frontline System Solver is built on top of the Microsoft Excel. As Excel has a limitation of at most 256 columns, we need to partition the matrices in order to calculate the desired quadratic objective function. We use some properties of *partitioned matrices*. For instance, if the number of items is 1000, the matrix Q used in our quadratic programming method is of order 1000 by 1000. We can partition the matrix into 16 submatrices. We utilize the following lemma (Leon, 1998) in our matrix computation.

Lemma 2 (*Block multiplication*). *If* $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ *where* A_{11}, A_{12}, A_{21} *and* A_{22} *are matrices of order* $n_1 \times m_1$, $n_1 \times m_2$, $n_2 \times m_1$ *and* $n_2 \times m_2$ *respectively, and* $B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$ *where* B_1 *and* B_2 *are matrices of order* $m_1 \times 1$ *and* $m_2 \times 1$ *respectively, then* $AB = \begin{pmatrix} A_{11}B_1 + A_{12}B_2 \\ A_{21}B_2 + A_{21}B_2 \end{pmatrix}$

8.1. Data sets

Synthetic data sets. In our experiment, we use the IBM synthetic data generator in Agrawal (2004) to generate the data set with the following parameters (same as the parameters of Wang and Su (2002)): 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 Hull (1997). We use the same settings as Wang and Su (2002). That is, 10% of items have the low profit range between \$0.1 and \$1.0, 80% of items have the medium profit range between \$1.0 and \$5.0, and 10% of items have the high profit range between \$5.0 and \$10.0.

We also generate two data sets to illustrate some weaknesses of HAP, as described in Section 3. They are called *HAP worst-case data set 1* and *HAP worst-case data set 2*. For HAP worst-case data set 1, we try to illustrate problem (2) as described in Section III. If there are 10,000 transactions and 1000 items, the first 20 transactions contain only two items, say I_1 and I_2. In this way a loop containing item I_1 and I_2 can be generated. Then, only one item randomly selected from the remaining items (i.e. $I_2, I_3, \ldots, I_{1000}$) is included in each of the remaining transactions ($t_{21}, t_{22} \ldots, t_{10000}$). The profit distribution is generated similarly as the IBM synthetic data set.

We now illustrate how to generate HAP worst-case data set 2. The idea is illustrated in figure 1. Items are divided into two layers: upper layer and lower layer. Items at each layer do not have any cross-selling effect with each other. We form pairs of items, one from each layer. For example, $\{I_A, I_B\}$, $\{I_C, I_D\}$ are such pairs. In each pair, such as $\{I_A, I_B\}$, transactions with I_A very likely also contain I_B, but transactions with I_B do not have a high chance to contain I_A. The association rule $I_A \rightarrow I_B$ has very high confidence, but $I_B \rightarrow I_A$ has low confidence.

We divide the transactions evenly for each item pair. Suppose there are 10,000 transactions and 1,000 items, hence 500 disjoint *item pairs*. A set of 20 transactions are related to each such item pair $\{I_A, I_B\}$. Consider a particular pair $\{I_A, I_B\}$. The first half (10 transactions) contains both item I_A and I_B. The second half (also 10 transactions) contains only item I_B but not item I_A. Similarly for each of the remaining 499 item pairs, 20 transactions are assigned and divided into the first half and the second half. Let us call the union of the transactions in the 499 first halves as the *first-half group*. We randomly insert item I_B into 80 transactions in the first-half group. With this insertion, we create some weak link from the items in the top layer to the items in the bottom layer. Other elements in the lower layer are treated in a similar manner.

In figure 1, we observe that the authorities of the items at the lower layer are accumulated. Thus, HAP will choose most of the items at the lower layer. There is little/no correlation

Table 2. Profit distribution of items in real data set.

Profit range	Proportion	Profit range	Proportion
\$0.0-\$0.1	2.03%	\$5–\$10	10.43%
\$0.1-\$1.0	25.05%	\$10–\$100	7.75%
\$1.0-\$5.0	54.59%	\$100–\$400	0.15%

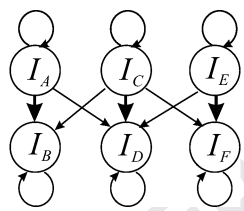

Figure 1. Illustration: Hap worst-case dataset 2.

among them, making the total profit of the selection small. If we assign high profit values to the items at the upper layer (e.g. I_A) for all transactions but low profit values to the items (e.g. I_B) at the lower layer, HAP would return poor results. In this data set, the items at the upper layer are assigned profits in the high profit range between \$5 and \$10 while the items at the lower layer are assigned profits in the low profit range between \$0.1 and \$1.0.

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 Table 2.

8.2. *Experimental results*

8.2.1. Results for synthetic data. We first experiment on the synthetic data sets. The results are shown in figures 2–4.[8]

For profitability, we observe that, for all data sets, 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.

For the HAP worst-case data set 1, the profitabilities of MPIS_Alg, QP, GA and Naive are similar. HAP gives a much lower profitability. For the HAP worst-case data set 2, GA and MPIS_Alg give the highest profitability and the second highest profitability among all algorithms, respectively. The QP and HAP approach gives the medium results in profitability, and Naive has the lowest results. Note that the QP tool does not guarantee the optimal

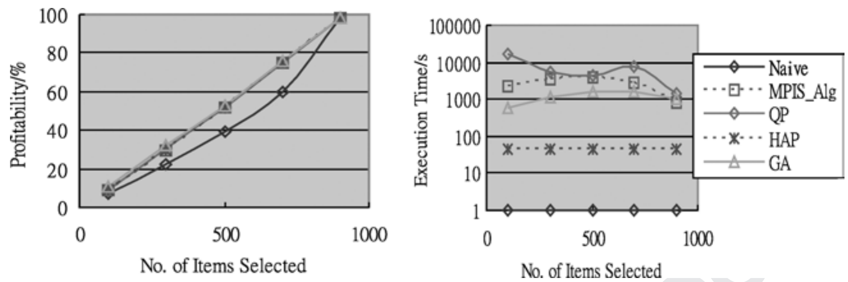

Figure 2. $n = 1000$, the profit follows log-normal distribution (NOTE: Profitability lines for MPIS_Alg, QP, GA and HAP are overlapping and execution-time line for HAP is slightly greater than that for naive).

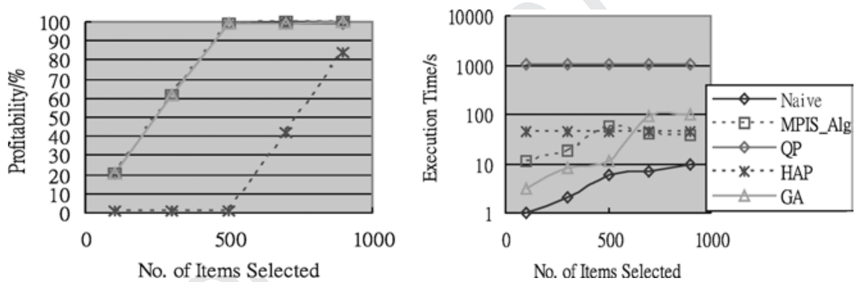

Figure 3. $n = 1000$, HAP-worst case data set 1.

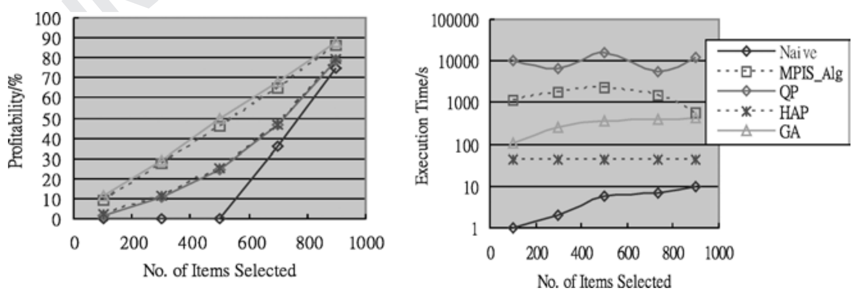

Figure 4. $n = 1000$, HAP-worst case data set 2.

solution, and also in our formulation, the objective function of QP is an approximation by the Principle of Inclusion-Exclusion (Definition 6).

When the selection size is varied, 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: (1) If there are more items to be selected, the benefit calculation in each iteration is more complex and updates to the benefit are more likely. The initial increase is related to the first factor. (2) When the number of items selected increases, the number of items to be removed in the iteration step decreases, and thus, the number of iterations decreases. The first factor is dominant when the selection is below 50% but the second factor becomes dominant in the other cases.

All algorithms involve the process of the calculation of the total profit as defined in Definition 5. The execution time of the calculation depends on two factors: (1) computation of the *csfactor*: As the number of selected items increases from 0%, there is increasing chance that both t_i' and d_i are non-empty set. That means $conf(I_j \rightarrow \Diamond d_i)$ has to be computed. The execution time will be greater. However, if the number of selected items approaches 100%, then there is higher chance that t_i' and d_i are empty sets. If t_i' is an empty set, we do not need to sum up the profit to the total profit because there are no selected items in the transaction. If d_i is an empty set, then we do not need to evaluate term $conf(I_j \rightarrow \Diamond d_i)$. Thus, the execution time will decrease. (2) computation depending on the *number of selected items*: if the number of selected items increases, we have to sum up the profit of the selected items, regardless of the term cs factor. So, the execution time will increase. In the experimental results we note that the first factor out-weighs the second factor, hence we see a increasing and then decreasing trend for both MPIS_Alg and GA.

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 Hohenbalken (1975) 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 of 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 observe that the authority weights converge rapidly.

The execution time of GAs usually increases with the number of items selected. This is because the computation time of the fitness function used in GAs increases with the number of items selected.

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 outperform HAP many folds (see the next subsection). GA is faster than MPIS_Alg as it does not involve a great number of chromosomes required for computation. Thus, it runs faster than MPIS_Alg.

8.2.2. Scalability. We have also studied the scalability of the algorithms Naive, MPIS_Alg, QP, HAP and GA. We have conducted two kinds of experiments. The first one is to study the variation of execution time against the number of items n. The other one is to study how the number of transactions m affects the execution time.

Similar to the previous synthetic data sets, we generated a number of data sets by using the IBM synthetic data generator (Agrawal, 2004) with the following parameters: 10 items per transaction on average, and 4 items per frequent itemset on average. The profit distribution is approximated by a lognormal distribution (Hull, 1997). 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. For the experiment of execution time with the variation of the number of items, the number of transactions is set to be 10,000 while the number of items is varied. Similarly, for the experiment examining execution time against the number of transactions, the number of items is set to be 1,000 while the number of transactions is varied. For each kind of experiment, we conducted the experiments 5 times with different data sets generated from IBM synthetic data set with different random seeds. The results are shown in figure 5 where the execution time is the average of different random sets.

We observe that the execution time of most algorithms increases with the number of transaction and the number of items because most algorithms run longer with a larger data size. However, for the execution time against the number of items n, we observe that the performance of GA decreases quickly with n. Recall that we have generated the data sets with the IBM synthetic data generator. The setting of the number of items per transaction (i.e. 10 items per transaction) is the same for different data sets with different total number of items n. So, with a smaller value of n, there is a higher chance that each item co-occurs

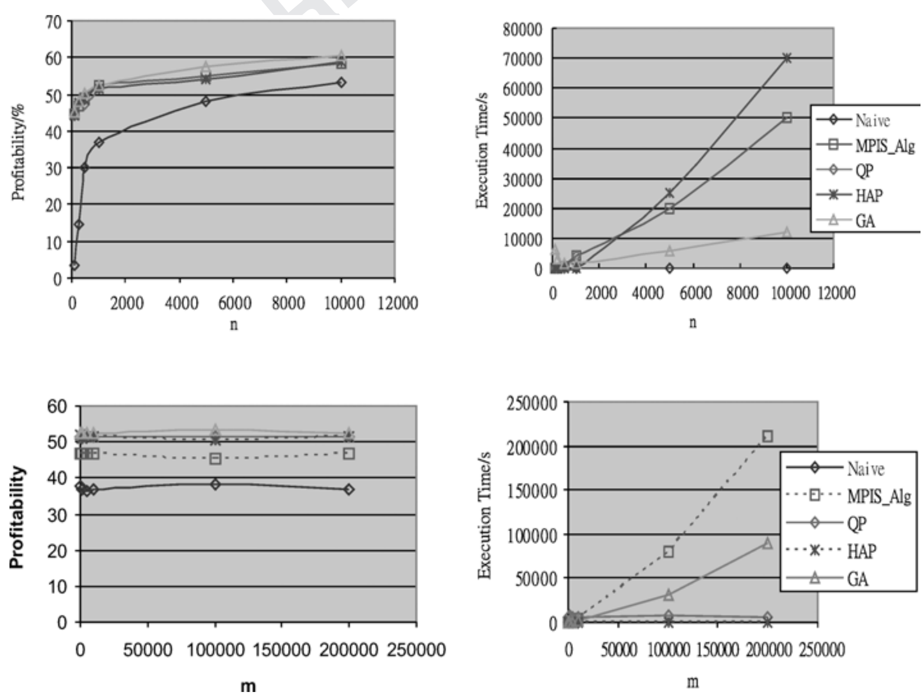

Figure 5. Graphs of Execution Time/Profitability against (1) total number of items n with $m = 10000$ and (2) total number of transactions m with $n = 1000$; the profit follows log-normal distribution.

P1: NVI
Data Mining and Knowledge Discovery SJNW243-05-NO00001359 May 9, 2005 21:6

with other items, or the number of items which co-occurs with an item is greater. This will lead to a greater number of branches from each node in FP-tree and FP-MPIS-tree (See Appendix A for the description of FP-tree and FP-MPIS-tree). As our implementation of computation of the confidence in the objective function is heavily dependent on FP-tree and FP-MPIS-tree, more branches are traversed for the computation of the objective function, which makes GA slower. Therefore the execution time of GA decreases with n initially.

For large n, the execution time of GA increases because the total profit requires more computation time. This is due to the fact that compared with other algorithms, GA requires heavier computations for the objective functions.

There is another algorithm which does not follow the general trend. For the graph of execution time against the number of transactions m, the line of QP remains nearly the same. We should note that the fundamental execution time of QP depends on the dimensionality of the quadratic objective function, which depends on n and not on m. No matter how the number of transactions changes, QP runs for nearly the same execution time.

The profitabilities of all algorithms increases with the number of items. As described above, the chance that two items co-occur is higher with smaller n. Thus, there is a stronger cross-selling effect among items. Once some items are missing, the effect of profit loss in other selected items is higher. For the same reason, the naive algorithm has a greater difference in profitability compared with other algorithms initially.

8.2.3. Results for the 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 (Frontline Systems Solver, http://www.solver.com/) 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 figures 6 and 7. 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 $I_j \rightarrow I_i$ is quite high. This creates a high authority weight for item I_i. Items like I_i would lead to smaller profitability for HAP.

The profitabilities/execution time of MPIS_Alg and GA are quite similar, hence in the following we only compare MPIS_Alg with the naive approach and HAP.

MPIS_Alg gives a greater profitability than the 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.

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

P1: NVI
Data Mining and Knowledge Discovery SJNW243-05-NO00001359 May 9, 2005 21:6

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 5.08 GB. If float data type (4 bytes) is required, then about 2.5 GB 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 found that less than 5 MB memory is needed. Our results in figures 6 and 7 are based on this enhanced hashing approach. However, the computation with this reduced size is still very massive.

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.

To summarize, the profit gain and efficiency considerations would make MPIS_Alg and GA the better choices for an application.

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,

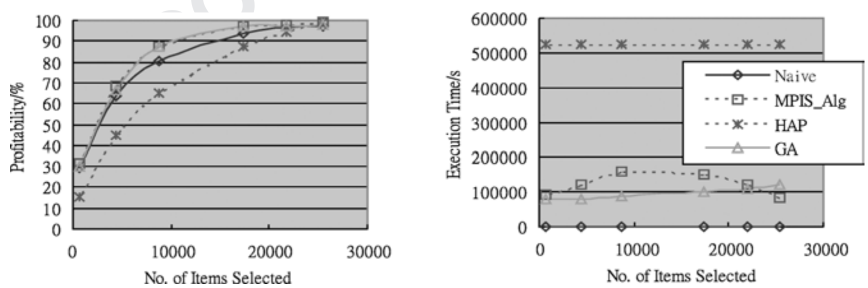

Figure 6. The drug store data set (NOTE: Profitability lines for MPIS_Alg and GA are overlapping).

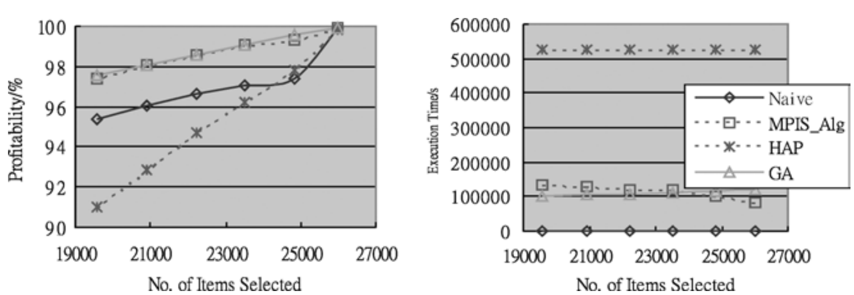

Figure 7. The drug store data set when $J > 19000$ (NOTE: Profitability lines for MPIS_Alg and GA are overlapping)).

the resulting profitabilities are much lower than those shown in figure 6. 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.

9. On the application of MPIS

We assume that patterns can be mined from the history of transactions. However, in the history we should be cautious of the following possible disturbing events:

1. *Out-of-stock situations:* Most supermarkets regularly face periods of temporal unavailability of particular products. As a result, some customers may purchase a *substitute product*, whilst others may not purchase the product at all.
2. *Promotional actions:* Promotional campaigns and product mix of a competing store influence purchase behavior of customers in the current store. Promotional effects may bias the purchases in a particular product category (say soft drink) towards the most advertised products (say Coca-Cola and Pepsi). This is called 'market concentration' in marketing. The historical transactional records of the store will be affected by this competitive environment.

The above problems can be avoided if we keep track of the out-of-stock events and special promotions so as to avoid the polluted data. Some kinds of store such as DIY (do-it-yourself) stores (i.e. stores where one can purchase screws, bolts, hammers, etc.) would be suitable for the product selection framework. DIY stores typically have a limited product assortment, with much less alternatives for product substitution, and the purchases are much less driven by promotional effects.

10. 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 a quadratic programming approach, a heuristical approach and an evolutionary approach to solve the MPIS problem. We show by experiments that these methods are efficient and effective.

We believe that there are many opportunities to extend the current work. 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. For example, item selection can be considered with an additional taxonomy of categories. Each item may belong to a specific category. For example, item *cheese* belongs to category *dairy products* which belongs to *food* while item *pencil* belongs to category *stationery*. The problem can be formulated as item selection such that the number of selected items in each category should be greater than or equal to a user-defined number, or as a problem of category selection instead of item selection.

Another investigation is possible if we have the customer identity in each transaction. We can study the behaviour that a customer may switch to another store altogether when some items are missing. In this case, not only the transactions of a customer with some missing items are interesting, but the other transactions by the same customer will also be of interest.

Appendix A: Proof of theorem

Theorem 2. *Given Observation* 1*, the total profit of item selection can be approximated by the following quadratic form.* $P = f^T s + \frac{1}{2} s^T H s$, *where* f *is a vector of size* n *and* H *is an* n *by* n *matrix.* $f = (f_j | f_j = \sum_{i=1}^{m} t_{ij}\, prof(I_j, t_i)(1 - \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk}) n)^T$ *for* $j = 1, 2, \ldots, n$ *and* $H = (h_{jk} | h_{jk} = \frac{2n_{jk}}{n_j} \sum_{i=1}^{m} t_{ij}\, prof(I_j, t_i) t_{ik}$ *for* $j, k = 1, 2, \ldots, n$).

Proof:

$$
\begin{aligned}
P &= \sum_{i=1}^{m} \sum_{I_j \in t_i'} prof(I_j, t_i)(1 - conf(I_j \to \Diamond d_i)) \quad \text{(byDef.2)} \\
&\approx \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij}'\, prof(I_j, t_i)\left(1 - \frac{1}{n_j} \sum_{k=1}^{n} d_{ik} n_{jk}\right) \\
&= \sum_{i=1}^{m} \left(a_i^T s + \frac{1}{2} s^T H_i s\right)
\end{aligned}
$$

where

$$
a_i = (g_{i1}\ g_{i2} \ldots g_{in})^T, \quad g_{ij} = t_{ij}\, prof(I_j, t_i)\left(1 - \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk}\right) \quad \text{for } j = 1, 2, \ldots, n
$$

$$
H_i = A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \quad \text{where } a_{jk} = \frac{2 t_{ij} prof(I_j, t_i) t_{ik} n_{jk}}{n_j}
$$

$$
\begin{aligned}
P &\approx \sum_{i=1}^{m} \left(a_i^T s + \frac{1}{2} s^T H_i s\right) = \sum_{i=1}^{m} (a_i^T s) + \sum_{i=1}^{m} \left(\frac{1}{2} s^T H_i s\right) \\
&= \left(\sum_{i=1}^{m} a_i^T\right) s + \frac{1}{2} s^T \left(\sum_{i=1}^{m} H_i\right) s \\
&= f^T s + \frac{1}{2} s^T H s
\end{aligned}
$$

where

$$f = \sum_{i=1}^{m} a_i = (f_1\ f_2\ \dots\ f_n)^T$$

where

$$f_j = \sum_{i=1}^{m} t_{ij}\, prof(I_j, t_i)\left(1 - \frac{1}{n_j}\sum_{k=1}^{n} t_{ik} n_{jk}\right) \quad \text{for } j = 1, 2, \dots, n$$

$$H = \sum_{i=1}^{m} H_i = \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ \dots & \dots & \dots & \dots \\ h_{n1} & h_{n2} & \dots & h_{nn} \end{pmatrix}$$

where

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

□

A. Implementation details of MPIS_Alg

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

B.1. Preparation phase

The individual count of each item can be derived if the database is scanned. For each transaction t_i, check whether the transaction t_i contains only one item, I_k. If yes, increment the individual count c_k. This counting can take place together with the counting of the item occurrences $n_1, \dots, n_n$.

The size 2 itemsets are kept in a hash table with a hash function on the sum of the item ids of the two items in each set. The count of each itemset is kept in the table. This facilitates the computation of $e_{i,j}$ when the support of such itemsets are needed.

B.2. Reading transactions from an FP-tree

In a number of cases, computation requires that the transactions in the database are examined; 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 (Han et al., 2000). 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 number of transactions is large. These repeated patterns are usually processed only once with $\mathcal{FPT}$. From our experiments this mechanism can greatly reduce the overall running time.

B.3. Mining size 2 itemsets

The size 2 itemsets are mined by a modified FP-growth algorithm with the constraint that the maximum size of itemsets mined should be 2. This mining is similar to the mining of frequent itemsets of all lengths in FP-growth. The difference is as follows. Suppose an FP-tree T has a single path P. In normal FP-growth, the complete set of the frequent itemsets of T can be generated by enumerating all the combinations of the subpaths of P. However, in our case, we generate the combinations of itemsets of size 2 only.

B.4. Calculating profit with the FP-MPIS-tree

In the definition of the profit of an item selection $P(A)$ (see Definition 5), we need to compute the number of transactions containing some item I and any item in set d_i (the value of $g(I, d_i)$), where $I \in A$ and $d_i \subseteq \mathcal{I} - A$. This is computed for many selections for each iteration, hence the efficiency is important. For this task, we use the FP-MPIS-tree (Han et al., 2000) data structure.

In the FP-MPIS-tree, we divide the items into two sets, $\mathcal{I} - A$ and A. Set A corresponds to items selected while $\mathcal{I} - A$ contains those not selected. The items in set $\mathcal{I} - A$ 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 8.

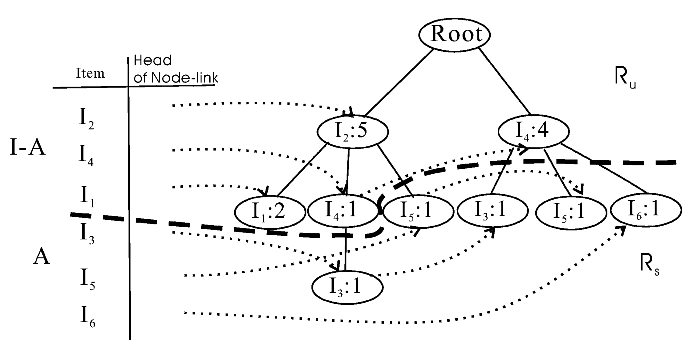

Figure 8. An example of an FP-MPIS-tree.

In this example, the set of selected items is $A = \{I_3, I_5, I_6\}$ and the set of unselected items is $\mathcal{I} - A = \{I_1, I_2, I_4\}$, where $\mathcal{I} = \{I_1, I_2, \ldots, I_6\}$.

To compute $g(I_a, d)$, we first look up the horizontal linked list (dotted links in figure 8) 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 parse FPTree(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-MPIS-tree to N. Starting from the node N, we traverse the tree upwards towards the root of the FP-MPIS-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. We adopt the refinements above in all of our experiments.

B.5. 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, and 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 item I_x or item I_k. The scanning is based on $\mathcal{FPT}$ and is highly efficient by making use of the linked lists starting from the header table. Let $P'(A, \mathcal{T})$ be the profit of the item selection A generated by transactions in $\mathcal{T}$. The item benefit is updated as follows.

$$b_i \leftarrow b_i + P'(S_i' \cup \{I_k\}, \mathcal{T}) - P'(S_i' \cup \{I_x\}, \mathcal{T})$$

The computation of $P'(A, \mathcal{T})$ can be done in a similar manner as $P(A)$ but $P'(A, \mathcal{T})$ 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.

Acknowledgments

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. We thank Y.L. Cheung for a version of the FP-tree mining program. We thank the anonymous reviewers for very helpful suggestions and comments. This research is supported by the Hong Kong RGC Earmarked Grant UGC REF.CUHK 4179/01E.

Notes

1. The problem is related to inventory management which has been studied in management science; however, previous works are mostly on the problems of when to order, where to order from, how much to order and the proper logistics (Taylor, 2001). The selection of items is also related to the problem of product assortment and shelf management (Brown and Tucker, 1961; Borin and Farris, 1995; Urban, 1998).
2. This definition is a generalization of the case where the 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 $prof(I, t_i)$ to be a constant over all i.
3. From marketing research/practice, we know that customers may purchase substitute products (e.g. different package size or different brand) when their favourite product is not available. In this case, the sales of the selected item I are not necessarily lost due to the absence of the set D because missing items in D may be substituted by other items. Some related work in this aspect are (Gruen et al., 2002; Campo et al., 2003; Kok and Fisher, 2004). It will be interesting to consider such item substitution in future work.
4. However, Brijs (2002) refines the original PROFSET and provides a decision alternative based on the strength of the dependency measured by log-linear analysis.
5. However, an indirect assessment (by excluding each item iteratively from the objective function) can be adopted in order to obtain a relative ranking of the selected items. This is because the objective function provides a quantitative assessment of the profit value of each product as a result of removal of this item from the optimal product set. From these profit values, a ranking can be derived.
6. Our model is based on the assumption that the purchase pattern will not change. For a more refined model, more elaborate before-after analysis must be undertaken but which could be too costly with the large number of products in a typical application.
7. Here we illustrate how to add the constraints on the assortment width and assortment depth. Suppose there are K categories in the store and the retailer would like to have items in at least W categories. We introduce K binary variables associated with each item I_i, say $C_{i,1}, C_{i,2}, \ldots C_{i,K}$. If I_i belongs to Category $C_{i,1}$, then $C_{i,1} = 1$, otherwise, $C_{i,1} = 0$. The binary assignments to $C_{i,2}, \ldots C_{i,K}$ are similar. Also introduce K binary variables w_j which can take on values of 0 or 1. Suppose D_j is the assortment depth requirement for category C_j (i.e. the minimum number of items in the category C_j). The assortment depth and width can be achieved by enforcing

    1. $\sum_{i=1}^{n} C_{i,j} s_i \geq D_j w_j$ for each category C_j,
    2. $\sum_{i=1}^{K} w_j \geq W$

8. In figure 3, the profitability lines for MPIS_Alg, QP, GA and naive are overlapping and execution-time lines for MPIS_Alg and HAP are slightly greater than that for naive. For figure 4, the profitability lines for QP and HAP are overlapping and execution-time line for HAP is slightly greater than that for naive.

References

Agrawal, R., 2004. IBM Synthetic Data Generator, http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html'.

Agrawal, R., Imilienski, T., and Swami. 1993. Mining association rules between sets of items in large databases In Proc. of ACM SIGMOD, pp. 207–216.

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In Proc. of 20th VLDB Conference, pp. 487–499.

Beasley, J. 1998. Heuristic algorithms for the unconstrained binary quadratic programming problem. In Technical report, the Management School, Imperial College, London.

Blischok, T. 1995. Every transaction tells a story. Chain Store Age Executive with Shopping Center Age 71(3): pp. 50–57.

Brijs, T., Goethals, B., Swinnen, G., Vanhoof, K., and Wets, G. 2000. A data mining framework for optimal product selection in retail supermarket data: The generalized PROFSET Model. Proc. of ACM SIGKDD, pp. 300–304.

Brijs, T., Swinnen, G., Vanhoof, K., and Wets, G. 1999. Using association rules for product assortment decisions: A case study. In Proc. of ACM SIGKDD, pp. 254–260.

Campo, K., Gijsbrechts, E., and Nisol, P. 2003. The impact of retailer stockouts on whether, how much, and what to buy. International Journal of Research in Marketing, 20:273–286.

Cavicchio, D.J. 1970. Adaptive search using simulated evolution. Ph. D. dissertation, Univ. Michigan, Ann Arbor, MI.

Garey, M. and Johnson, D. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. USA: Freeman.

Gruen, T., Corsten, D., and Bharadwaj, S. 2002. Retail out-of-stocks: A world- wide examination of extent, causes and consumer responses. In Grocery Manufacturers of America.

Han, J., Pei, J., and Yin, Y. 2000. Mining frequent patterns without candidate generation. In Proc. of ACM SIGMOD, pp. 1–12.

Hedberg, S. 1995. The data gold rush. In BYTE, pp. 83–99.

Hiller and Lieberman. 2001. Introduction to Operations Research, 7th ed, USA: McGraw Hill.

Hohenbalken, B.V. 1975. A Finite algorithm to maximize certain pseudoconcave functions on polytopes. In Mathematical Programming 8.

Horst, R., Pardalos, P.M., and Thoai, N.V. 2000. Introduction to Global Optimization, 2nd edn. Kluwer Academic Publishers.

Hull, J.C. 1997. Options, Futures, and Other Derivatives, 3rd edn. Prentice Hall International, Inc.

Iasemidis, L., Pardalos, P., Sackellares, J., and Shiau, D. 2001. Quadratic binary programming and dynamical system approach to determine the predictability of epileptic seizures. Journal of Combinatorial Optimization, Kluwer Academic, 5: 9–26.

Kleinberg, J., Papadimitriou, C., and Raghavan, P. 1998. A microeconomic view of data mining. Knowledge Discovery Journal, 2. 311–324.

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

Kok, A. and Fisher, M. 2004. Demand Estimation and Assortment Optimization under Substitution: Methodology and Application. In Working paper,Fuqua School of Business, Duke University.

Leon, S.J. 1998. Linear Algebra with Applications, 5th edn. Prentice Hall.

Luo, J., Pattipati, K.R., and Willett, P. 2001. A sub-optimal soft decision PDA method for binary quadratic programming. In Proc. of the IEEE Systems, Man, and Cybernetics Conference, pp. 3140–3145.

Mahfoud, S.W. 1992. Crowding and preselection revisited. Parallel Problem Solving from Nature, 2:27–36.

Mannila, H. 1997. Methods and problems in data mining. In Proc. of Int. Conf. on Database Theory, pp. 41–55.

Mannila, H., Toivonen, H., and Verkamo, A.I. 1994. Efficient algorithms for discovering association rules. In Proc. of KDD, pp. 181–192.

Safronov, V. and Parashar, M. 2002. Optimizing web servers using page rank prefetching for clustered accesses. In World Wide Web: Internet and Web Information Systems V 5(1): 165 – 176.

Sahni, S. 1974. Computationally related problems. SIAM J. Comput. 3:262–279.

Syswerda, G. 1989. Uniform crossover in genetic algorithms. In Proc. 3rd International Conference of Genetic Algorithms, pp. 2 – 9.

Taylor, B. 2001, Inventory Management. In: Introduction to Management Science, 7th Edition. Prentice Hall, Chapter 16.

Ullman, J. 2003. Lecture Notes at http://www-db.stanford.edu/∼ullman/mining/mining.html’.

Urban, T. 1998. An inventory-theoretic approach to product assortment and shelf-space allocation. Journal of Retailing, 74:15–35.

Wang, K. and Su, M. 2002. Item selection by “Hub-Authority,” Profit Ranking. In Proc. of ACM SIGKDD, pp. 652–657.

Wong, R., Fu, A., and Wang, K. 2003. MPIS: maximal-profit item selection with cross-selling considerations. In Proc. of IEEE ICDM, pp. 371–378.