A Study of Frequent Pattern and Association Rule Mining - with Applications in Inventory Update and Marketing

WONG, Chi-Wing

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Supervised by

Prof. Ada Wai-Chee Fu

©The Chinese University of Hong Kong
June 2004

Abstract of thesis entitled:

A Study of Frequent Pattern and Association Rule Mining - with Applications in Inventory Update and Marketing

Submitted by WONG, Chi-Wing

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in June 2004

In the literature of data mining, many different algorithms for mining association rule have been proposed. However, there is relatively little study on how association rules can aid in more specific targets. In this thesis two of the applications for the concepts of association rules - (1) Maximal-Profit Item Selection with Cross-Selling Effect (MPIS) problem and (2) Item Selection for Marketing with Cross-Selling Effect (ISM) problem - are investigated.

MPIS: Problem MPIS 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 variation 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 heuristics approach and an evolutionary approach - genetic algorithm - to tackle this problem. Experiments are conducted to show that the proposed methods are highly effective and efficient.

ISM: Problem ISM is to find a subset of items as marketing items in order to boost the sales of the store. We prove that a simple version of this problem is NP-hard. The cross-selling effect in this problem is modeled by the *gain rule* - another variation of association rule. We also propose two algorithms to deal with this problem. One is a quadratic programming method while the other one is a heuristics method. Experiments are also conducted to show that the algorithms are effective and efficient.

論文題目： 頻繁樣式的研究及規則分析採集的應用——庫存更新和市場推銷

作者 ： 黃智榮

學校 ： 香港中文大學

學系 ： 計算機科學及工程學系

修讀學位： 哲學碩士

摘要 ：

在數據採集的文學研究中，有很多不同規則分析採集的算法已被建議。但是，當中只有很少特別為規則分析應用的研究。在這論文中， 兩個配合跨售行銷影響的規則分析應用會被研究 —— (1) 最大利盈的項目選擇(MPIS) 和 (2) 推銷的項目選擇(ISM)。

最大利盈的項目選擇(MPIS)：問題 MPIS 是在盡多的項目中選取某些項目，而在考慮跨售行銷影響的情況下，得到最大的利盈。我們已證明這簡化的問題為 NP-hard。我們建議了一個新的方法去模擬跨售行銷的影響。這方法名為損失規則分析，我們也證明了這問題可以轉化為二次規劃法。當二次規劃法不能應用時，我們也提議了一種試探性方法。此外，我們也提議一種演化式演算法——遺傳程式方法——去解決這問題。我們所做的實驗能顯示了我們所建議的方法為有效的。

推銷的項目選擇(ISM)：問題 ISM 是在盡多的項目中選取某些項目為推銷項目，從而增加商鋪的銷售額。我們證明了這簡化的問題為 NP-hard。在這問題的跨售行銷影響被一種名為增進規則分析所模擬。我們也提議了兩種方法去解決這問題。方法一是二次規劃法，方法二是試探性方法。我們所做的實驗能夠顯示了這些方法為有效的。

Acknowledgement

I would like to thank my supervisor, Prof. Ada Wai-Chee Fu, for her patient guidance, ceaseless support, insightful ideas and generous encouragement in all aspects throughout the two years of my MPhil study. I am very grateful to have hers as my supervisor. She introduced me to research, polished my writing and presentation skills, and gave me opportunities to attend international conferences, which benefit me for the rest of life. Without her effcort, I will not be able to strengthen and improve my research and papers (published in ICDM 2003 and PAKDD 2004). I am proud of being her student, now and forever.

Thanks go to Prof. Ke Wang, Prof. Jeffrey Yu and Prof. Ho-Fung Leung for their invaluable comments, and to Miss Mingfei Jiang and Mr. Yung-Hang Lau for sharing their knowledge and insightful discussions in my research. In particular, I would like to thank my dear parent and my dear brother Chi-Wah Wong, for his support, encouragement and understanding.

I would also like to give my thanks to my fellow colleagues, Che-Yin Ip, Ying-Kin Yu, Yee Lam, Chi-Hang Chan, Wan-Yeung Wong, Lok-Hang Lee, Siu-Fung Wong, Nga-Sin Lau, Jill Law, Pik-Wah Chan, Chi-Hung Law, Wah Hung, Chi-Wai Leung, Edith Ngai, Vesta Lee, Denny Zheng, Paul Pun, Walty Yeung, Alex Fok and Shirley Ng. They have given me a joyful and wonderful time in my research. These two years should not have been that fruitful without them.

Last but not least, I would like to share some words here to respect all the scientists and engineers for their contributions which we can ground on when developing our future.

This work is dedicated to my family for the support and patience.

Contents

Abstract i

Acknowledgement iv

1 Introduction 1

1.1 MPIS 3

1.2 ISM 5

1.3 MPIS and ISM 5

1.4 Thesis Organization 6

2 MPIS 7

2.1 Introduction 7

2.2 Related Work 10

2.2.1 Item Selection Related Work 11

2.3 Problem Definition 22

2.3.1 NP-hardness 25

2.4 Cross Selling Effect by Association Rules 28

2.5 Quadratic Programming Method 32

2.6 Algorithm MPIS_Alg 41

2.6.1 Overall Framework 43

2.6.2 Enhancement Step 47

2.6.3 Implementation Details 48

2.7 Genetic Algorithm 60

2.7.1 Crossover 62

2.7.2 Mutation 64

2.8 Performance Analysis 64

2.8.1 Preparation Phase 65

2.8.2 Main Phase 69

2.9 Experimental Result 77

2.9.1 Tools for Quadratic Programming 77

2.9.2 Partition Matrix Technique 78

2.9.3 Data Sets 81

2.9.4 Empirical Study for GA 84

2.9.5 Experimental Results 92

2.9.6 Scalability 102

2.10 Conclusion 106

3 ISM **107**

3.1 Introduction 107

3.2 Related Work 108

3.2.1 Network Model 108

3.3 Problem Definition 112

3.4 Association Based Cross-Selling Effect 117

3.5 Quadratic Programming 118

3.5.1 Quadratic Form 119

3.5.2 Algorithm 128

3.5.3 Example 129

3.6 Hill-Climbing Approach 134

3.6.1 Efficient Calculation of Formula of Profit Gain 134

3.6.2 FP-tree Implementation 135

3.7 Empirical Study 136

3.7.1 Data Set 137

3.7.2 Experimental Results 138

3.8 Conclusion 141

4 Conclusion **147**

Bibliography **153**

List of Figures

2.1 An example of spider traps that there is a loop ($I_i \rightarrow I_j$ and $I_j \rightarrow)I_i$, leadings to all accumulated weights of items I_i and I_j and low weights (even zero weights) of other items I_a, I_b and I_c 19

2.2 Items with High Authority 21

2.3 An Example of a CLIQUE of size 3 27

2.4 An example of an FP-tree 50

2.5 An example of an FP-MPIS-tree (The set of selected items is $A = \{I_3, I_5, I_6\}$ and the set of unselected items is $I - A = \{I_1, I_2, I_4\}$, where $I = \{I_1, I_2, ..., I_6\}$) 51

2.6 Simple Steady-State Genetic Algorithm 61

2.7 Pseudo-code for FP-growth 67

2.8 Illustration: Hap Worst-Case dataset 1 82

2.9 Graph of Average Profitability of GAs against Population Size, where $J = 500$ and no. of generations=20 87

2.10 Graph of Average Execution Time of GAs against Population Size, where $J = 500$ and no. of generations=20 87

2.11 Graph of Average Profitability of GAs against No. of Generations, where $J = 500$ and population size=100 89

2.12 Graph of Execution Time of GAs against No. of Generations, where $J = 500$ and population size=100 89

2.13 Graph of Profitability against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (With Pruning Step) 94

2.14 Graph of Execution Time against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (With Pruning Step) 94

2.15 Graph of Profitability against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (Without Pruning Step) 94

2.16 Graph of Execution Time against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (Without Pruning Step) 94

2.17 Graph of No. of Remaining Items after Pruning Step where $n = 1000$ and the profit follows log-normal distribution 95

2.18 Graph of Profitability against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 1 96

2.19 Graph of Execution Time against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 1 96

2.20 Graph of Profitability against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 2 96

2.21 Graph of Execution Time against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 2 96

2.22 Graph of No. of Remaining Items after Pruning Step where $n = 1000$ for the data set which HAP performs worst 97

2.23 Graph of Execution Time against No. of Items Selected for the drug store data set when $J < 1000$ 103

2.24 Profitability and Execution Time against No. of Items Selected The drug store data set 103
2.25 Graph of Profitability and Execution Time against No. of Items Selected The drug store data set when $J > 19000$ 104
2.26 Graphs of Execution Time against total number of items n and against total number of transactions m, the profit follows log-normal distribution (NOTE:Execution Time lines for HAP and naive are overlapping) 104

3.1 A solution of MAXCUT problem 117
3.2 Graph of Profit Gains against α for Synthetic Data Set 1 (Discount) 141
3.3 Graph of Execution Time against α for Synthetic Data Set 1 (Discount) 141
3.4 Graph of Profit Gains against α for Synthetic Data Set 1 (Free) 141
3.5 Graph of Execution Time against α for Synthetic Data Set 1 (Free) 141
3.6 Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 1.5$ 142
3.7 Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 2$ 142
3.8 Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 2.5$ 142
3.9 Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 3$ 142
3.10 Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 1.5$ 143

3.11 Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 2$ 143
3.12 Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 2.5$ 143
3.13 Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 3$ 143
3.14 Graph of Profit Gains against α for Synthetic Data Set 2 (Discount) 144
3.15 Graph of Execution Time against α for Synthetic Data Set 2 (Discount) 144
3.16 Graph of Profit Gains against α for Synthetic Data Set 2 (Free) 144
3.17 Graph of Execution Time against α for Synthetic Data Set 2 (Free) 144
3.18 Graph of Profit Gains against α for Real Data Set (Discount) 145
3.19 Graph of Execution Time against α for Real Data Set (Discount) 145
3.20 Graph of Profit Gains against α for Real Data Set (Free) . . . 145
3.21 Graph of Execution Time against α for Real Data Set (Free) . 145

List of Tables

2.1 An Example of Transactions (Binary Representation) 17

2.2 An Example of Transactions (Item Representation) 18

2.3 Average Profit of each item 18

2.4 Count of co-occurrence of items I_i and I_j, n_{ij} 56

2.5 Benefit of items calculated after pruning in Algorithm MPIS_Alg 59

Chapter 1

Introduction

Industry players agree that data mining technology, as one of the strong pillars in CRM (Customer Relationship Management), plays a vital role in business expansion. By knowing their customer behavior based on past records, increased profits from cross-selling and other business strategies would be achieved. The behaviour in terms of sales transactions is considered significant [16]. 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 [24] 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 [32] 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:

$$\max_{x \in \mathcal{D}} \sum_{i \in \mathcal{C}} \mathcal{G}(x, y_i) \tag{1.1}$$

where $\mathcal{D}$ is the set of all possible decisions in the domain problem (e.g. inventory control and marketing), $\mathcal{C}$ 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 deeper investigation of such decision problems, we notice that we are in fact dealing with a maximization problem of the following form:

$$\max_{x \in \mathcal{D}} \mathcal{G}(x, Y) \tag{1.2}$$

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 thesis is about such a problem: optimal product selection [18, 17, 49, 52] (in SIGKDD 1999, 2000, 2002). The problem is that in a typical retail store, the types of products should be refreshed regularly in order that losing products are discarded and new products are introduced. As pointed out in [16], a major task of talented merchants is to pick the profit-generating items and discard the losing items. Thus, we are interested in finding 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 (1.2). [1]

Association rule mining [11] 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 [18] 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 two problems of the application of association rules - (1) Maximal-Profit Item Selection with cross-selling considerations (MPIS) [52] problem and (2) Item Selection for Marketing with Cross-Selling Effect (ISM) problem [51].

1.1 MPIS

MPIS [52] 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

[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 [46].

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 [52]. 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 [49], is 6.5 times slower than MPIS_Alg. These 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.

1.2 ISM

ISM [51] is to find a subset of items as marketing items so that, after a marketing campaign on these selected marketing items is promoted, the whole sales of the store will be increased and the store will earn much money.

In this thesis, the problem of item marketing is studied. Besides, we prove that a simple version of this problem is NP-hard. The cross-selling factor is modeled by the *gain rule*. The rule is of the form $\diamond t' \to I$, where I is a non-marketing item and t' is a set of marketing items. This gain rule helps us to estimate the gain in profit of item I if all items in t' are chosen as marketing items. The rule corresponds to the cross-selling effect between I and t'.

We also proposed two algorithms to solve this problem. One is a hill-climbing approach while the other is a classical optimization approach. In our experiment, both approaches are quite effective and efficient. Our proposed approaches give a greater profit gain compared with direct marketing.

1.3 MPIS and ISM

At first glance, problem MPIS and problem ISM are quite similar. Both problems consider the significant factor - cross-selling factor. But, after taking a closer look, we can find that there are some differences between these two problems.

Problem MPIS is typically helpful when the retail store wants to pick the profit generating items and discard the losing items. Problem MPIS is one which selects a set of J items with the consideration of the cross-selling effect so that the total profit of the item selection is maximized by minimizing the profit loss of some items due to the selection. Problem MPIS is defined based on the assumption that the profit of selected items may be affected if some items are not included in the final selection. This assumption is modeled by a special kind of association rule called *loss rule*. This rule is interpreted as the loss in profit of the selected items if some other items are not selected.

Problem ISM is useful when the store manager wants to decide which items should be marketed in order to boost the sales in the store. Problem ISM is one which finds a set of marketing items with the consideration of the cross-selling effect in order that the total profit of the item selection is maximized by maximizing the profit gain of both marketing items and non-marketing items. Problem ISM is based on the assumption that the marketing items can be affecting the sales of the non-marketing items. Such kind of assumption is modeled by another kind of rule, called *gain rule*. This rule represents how the marketing items affect the sales of non-marketing items. In other words, this rule can help to estimate the profit gain of item selection.

1.4 Thesis Organization

In this thesis, we first describe problem MPIS and the proposed algorithms in Chapter 2. Then, we describe problem ISM and the proposed approaches in Chapter 3. Finally, we conclude our thesis in Chapter 4.

Chapter 2

MPIS

2.1 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 [16]. 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 [24] 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 [11] 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 [16], 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 profits 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 [32] 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 thesis, the problem of Maximal-Profit Item Selection with Cross-Selling Considerations (MPIS) [52] 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 variation of association rule called *loss rule*. The rule is of the form $I_a \rightarrow \diamond d$, 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), a heuristics method called MPIS_Alg and a genetic algorithm (GA). 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. GA can be viewed as an optimization method which encodes a possible solution of a problem on a chromosome-like data structure, creates a number of chromosomes in a population and finds a solution by recombining the chromosomes in the population.

From our experiment, the profitabilities of our three proposed algorithms are greater than that of a naive approach for all data sets. On average, the profitability of both QP, GA 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 [49] 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.

The rest of this chapter is organized as follows. Section 2.2 introduces the related work of our problem. Section 2.3 gives a formal definition of our problem. Section 2.4 presents the details of the cross selling effect by association rule - loss rule. Section 2.5 and Section 2.6 gives our proposed algorithms - QP and MPIS_Alg. We will analyze our algorithm MPIS_Alg in Section 2.8. Section 2.9 presents our performance study. Section 2.10 summarizes our study.

2.2 Related Work

Recently the problem of association rule mining has received much attention. We assume a database for basket data with a set of transactions. We are given a set $I = \{I_1, I_2, ..., I_n\}$ of binary attributes called *items* where n is the total number of items. For example, $I = \{Carrot, Orange, Knife, Potato, Plate\}$. The database contains m transactions. For $i = 1, 2, ..., m$, transaction t_i is a binary vector with $t_i[k] = 1$ if item I_k is in transaction t_i. An association rule has the form $X \rightarrow I_j$, where $X \subseteq I$ and I_j is one of the elements in I. For instance, $\{Orange, Potato\} \rightarrow Carrot$.

There are two major elements associated with the association rule, *support* and *confidence factor*. Support for a rule $X \rightarrow I_j$ is equal to the fraction of the number of transactions containing all items in X and item I_j, over the number of all transactions. The confidence factor for the rule $X \rightarrow I_j$ is equal to the fraction of the transactions containing all items in set X that also contain item I_j. The problem of mining association rules is to find all rules which have the support and confidence greater than the corresponding user-defined threshold value called *minimum support (minsup)* and *minimum confidence (minconf)*. Some of the earlier work include [40, 12, 39].

2.2.1 Item Selection Related Work

There are some recent works on the maximal-profit item selection problem - PROFSET and HAP. In the following subsections, these two methods will be described in some details since they are closely related to our work.

PROFSET

PROFSET ([18] and [17]) is an algorithm based on size-constrained programming. PROFSET makes use of the disjoint maximal frequent itemset to represent the "purchase interest" so as to find the selection of items by using 0-1 programming with the constraint of a specified size.

A *frequent itemset* is a set which contains items co-occurring frequently. A *maximal frequent itemset* is a frequent itemset which does not have a frequent item superset. For example, if there are three items $\{I_a, I_b, I_c\}$, and all frequent itemsets found are [1] $\{I_a\}$, $\{I_b\}$, $\{I_c\}$ and $\{I_a, I_b\}$ $\{I_b, I_c\}$. Frequent itemsets $\{I_a\}$, $\{I_b\}$ and $\{I_c\}$ are not maximal as they have frequent item superset $\{I_a, I_b\}$ and $\{I_b, I_c\}$. But, frequent itemsets $\{I_a, I_b\}$ and $\{I_b, I_c\}$ are maximal as they have no frequent item superset.

PROFSET formulates the problem as a binary linear programming. It is assumed that we are given a set of transactions of profit margin of each item in each transaction. Before describing such programming, we first describe some notations used in PROFSET. Let $M(X)$ be the profit margin of the frequent itemset X for all transactions and $m(X)$ be the margin of the frequent itemset X for a transaction. The term $M(X)$ is obtained by summing up a number of $m(X)$ terms in some transactions containing frequent itemset X, where $m(X)$ is the margin obtained by summing up all margins of each item

[1] I_a is the notation of the item with item number a

in frequent itemset X in a transaction. The algorithm of the calculation of $M(X)$ is shown as follows.

```
for every transaction $t_j$ do
begin
    while ($t_j$ contains any frequent itemsets) do
    begin
        Draw $X$ from the set containing all possible maximal
            frequent subsets in transaction $t_j$ using probability
            distribution $\ominus_{t_j}$
        $M(X) \leftarrow M(X) + m(X)$ where $m(X)$ is the profit
            margin of $X$ in $t_j$
        $t_j \leftarrow t_j - X$
    end
end
return all $M(X)$
```

The probability distribution $\ominus_{t_j}$ is defined with:

$$\ominus_{t_j}(X) = \frac{support(X)}{\sum\limits_{Y \in t_j} support(Y)}$$

where X and Y are the maximal frequent itemsets in transaction t_j.

Now, we are going to describe the binary programming in PROFSET. Let the set of categories $\{C_1, ...C_n\}$ be the set of items and L be the set of frequent itemsets. For example, item *cheese* belongs to category *food* while item *pencil* belongs to category *stationery*. Let P_X and Q_i be binary decision variable used in the programming. $P_X = 1$ if a maximal frequent itemset X is involved in the contribution to the objective function. Otherwise, $P_X = 0$. Q_i is set to be 1 if item I_i is included in the maximal frequent itemset X and $P_X = 1$. Otherwise, Q_i is set to be 0. Let $Cost_i$ be the inventory and handling cost of item I_i. The objective is to maximize all profits from cross-selling effects between items:

Maximize

$$\sum_{X \in L} M(X)P_X - \sum_{c=1}^{n} \sum_{i \in C_c} Cost_i Q_i$$

Subject to

$$\sum_{c=1}^{n} \sum_{i \in C_c} Q_i = J \tag{2.1}$$

$$\forall X \in L, \forall i \in X : Q_i \geq P_X \tag{2.2}$$

$$\forall C_c : \sum_{i \in C_c} Q_i \geq ItemMin_{C_C} \tag{2.3}$$

Constraint 2.1 is used to determine J items to be selected in the problem, where J is a user parameter. This value is dependent on the retail environment (e.g. the number of eye-catchers and the number of facings in a promotion leaflet). Constraint 2.2 is used to specify all items in a selected maximal frequent itemset should also be selected. Constraint 2.3 restricts that the number of items in each category should be greater than or equal to a user-defined values (i.e. minimum number of items in each category).

However, PROFSET has some weaknesses as pointed out in [49].

1. PROFSET is independent of the strength of the relationship between items (i.e. the level of confidence).

2. The purchase intention is not just shown in the maximal frequent itemsets because subsets of maximal frequent itemsets also show the purchase intention, which are not considered in PROFSET.

3. PROFSET does not give the relative ranking of the selected items.

HAP

HAP [49] is a solution of a similar problem. Before describing the details of HAP, we first give an overview of HAP. It applies the "hub-authority" profit ranking approach [43] to solve the maximal profit item-selection problem. Items are considered as vertices in a graph. A link $I_i \rightarrow I_j$ represents the cross-selling effect from I_i to I_j. The strength of the link is given by the confidence of the association rule $I_i \rightarrow I_j$ and the *individual profit* of item I_i, where the individual profit of item I_i represents the recorded profit of item I_i in all transactions . 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. A good *hub* I_i is an item which leads to many other items I_j (with link $I_i \rightarrow I_j$) with a strong strength of the link. The *hub weight* and the *authority weight* of a vertice/item indicate the level of a good hub and a good authority, respectively. The greater the weight is, the better hub/authority the item is. Then the HITS algorithm [33] 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.

Now, we are going to describe HAP in details. As we mentioned before, each item is associated with two values - hub weight and authority weight. HAP defines two n-dimensional vectors, h and a, to represent the hub weight and authority weight, respectively. Entry $h(i)$ denotes the hub weight item I_i while entry $a(i)$ denotes the authority weight of item I_i. HAP models the update of each authority weight and hub weight as follows.

$$a(i) = \sum_{I_j \rightarrow I_i} prof(I_j) \times conf(I_j \rightarrow I_i) \times h(j)$$

$$h(i) = \sum_{I_i \to I_j} prof(I_i) \times conf(I_i \to I_j) \times a(j)$$

where $prof(I_i)$ is the total profit of item I_j in all transactions.

From the above modeling, we can derive the following formula:

$$h = B \cdot a$$

$$a = B^T \cdot h$$

where B is called a cross-selling matrix and the entry $B(i,j)$ is shown as follows.

$$B(i,j) = \begin{cases} prof(I_i) \times conf(I_i \to I_j) & \text{if } i \neq j \text{ and there is} \\ & \text{an association } I_i \to I_j \\ 0 & \text{if } i \neq j \text{ and there is} \\ & \text{no association } I_i \to I_j \\ prof(I_i) & \text{if } i = j \end{cases}$$

From the above two formula, we can also deduce the following:

$$h = BB^T h = (BB^T)h \tag{2.4}$$

$$a = B^T Ba = (B^T B)a \tag{2.5}$$

The hub weight and the authority weight can be updated with the following two methods.

1. Iteration Approach
2. Principal eigenvectors Approach

1. **Iterative Approach**

   We can implement the update of the weights with k iterations.

   In the implementation of this approach, we first assign the authority weight and hub weight with an initial value (e.g. 1). It is noted that the converged weights of these vectors is independent of the initial values of the vectors [35]. Then, the weights are updated k times with the formula (2.4) and (2.5). [49] and [33] pointed out that the weight vector a and v converges rapidly, usually after 20 iterations.

2. **Principal Eigenvectors Approach**

   From the previous iterative approach, we can derive the following:

   For the first iteration,

$$\begin{aligned} h &= (BB^T)h & (From(2.4)) \\ a &= (B^TB)a & (From(2.5)) \end{aligned}$$

   For the second iteration,

$$\begin{aligned} h &= (BB^T)(BB^T)h & (From(2.4)) \\ &= (BB^T)^2h & \\ a &= (B^TB)(B^TB)a & (From(2.5)) \\ &= (B^TB)^2a & \end{aligned}$$

   For the k-th iteration,

$$\begin{aligned} h &= (BB^T)^kh \\ a &= (B^TB)^ka \end{aligned}$$

   As k approaches infinity, we can find the principal eigenvector of the matrix BB^T and B^TB as the hub vector h and authority weight a.

Example 1 (HAP Example) *We illustrate HAP with the following example.*

There are six items, namely I_1, I_2, I_3, I_4, I_5 and I_6. There are 10 transactions. Table 2.1 show an example of transactions in binary representation.

Table 2.1: An Example of Transactions (Binary Representation)

TID	I_1	I_2	I_3	I_4	I_5	I_6
1	0	0	0	0	0	1
2	0	0	0	1	0	0
3	0	1	0	0	0	0
4	0	0	0	1	1	0
5	0	0	1	1	0	0
6	1	1	0	0	0	0
7	0	0	0	1	0	0
8	0	1	0	0	1	0
9	0	1	1	1	0	0
10	1	1	0	0	0	0

However, the example can be expressed in item representation (shown in the Table 2.2). The average profit p_i of item I_i is shown in Table 2.3. Assume the number of items to be selected, J, is 3.

The cross-selling matrix is:

$$B = \begin{pmatrix} 20 & 40 & 0 & 0 & 0 & 0 \\ 14 & 7 & 7 & 7 & 7 & 0 \\ 0 & 20 & 20 & 40 & 0 & 0 \\ 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 19 & 0 & 19 & 19 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Let us explain the entry b_{54}. The formula is $b_{54} = prof(5) \times conf(I_5 \rightarrow I_4$, where $prof(5)$ is the individual profit of item I_5 (i.e. $19 \times 2 = 38$). Then, $b_{54} = 38 \times \frac{1}{2} = 19$.

Authority vector a and hub vector h are initialized to $(1\ 1\ 1\ 1\ 1\ 1)^T$. Then,

Table 2.2: An Example of Transactions (Item Representation)

TID	Items Bought	(Ordered) Frequent Items
1	I_6	I_6
2	I_4	I_4
3	I_2	I_2
4	I_4, I_5	I_4, I_5
5	I_3, I_4	I_4, I_3
6	I_1, I_2	I_2, I_1
7	I_4	I_4
8	I_2, I_5	I_2, I_5
9	I_2, I_3, I_4	I_2, I_4, I_3
10	I_1, I_2	I_2, I_1

Table 2.3: Average Profit of each item

i	p_i
1	20
2	7
3	20
4	1
5	19
6	2

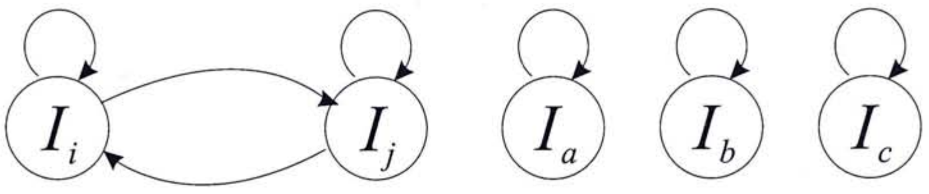

Figure 2.1: An example of spider traps that there is a loop ($I_i \to I_j$ and $I_j \to$)I_i, leadings to all accumulated weights of items I_i and I_j and low weights (even zero weights) of other items I_a, I_b and I_c

the following process iterates until these two vectors converge.

$$h = Ba$$

$$a = B^T h$$

The following authority vector is obtained.

$$a = \begin{pmatrix} 0.2200262093 \\ 0.7127000750 \\ 0.2456995424 \\ 0.5976062503 \\ 0.1616959330 \\ 0.0000000000 \end{pmatrix}$$

If $J = 3$, the items with the highest authority weights are selected. That is, items I_2, I_3 and I_4 are selected. The total profit of the item selection[2] *is \$72.8.* □

However, HAP also has some weaknesses.

1. Problems of dead ends or spider traps as illustrated in [48] can arise. For example, if there is an isolated subgraph with a cycle while other

[2]The method of the calculation of the total profit of the item selection will be described in Section 2.4.

items are not connected, then the authority weight and hub weight of all items in the cycle are accumulated and are increased to extremely high values, giving an over-estimating ranking for these items.

2. Some items may be given redundant or additional hub/authority weight in HAP. Let us consider the following example.

I_i	I_j	I_k
1	0	0
1	1	1

   From the above transactions, several rules are generated. We just consider these two rules $I_j \rightarrow I_i$ and $I_k \rightarrow I_i$. The consequent items of these two rules are I_i. According to HAP, item I_i will gain in the calculation of the profit twice. One is from item I_j while the other is from item I_k. However, there is only one real transaction, actually, the profits induced by item I_j and item I_k should be counted once only.

3. HAP may perform poorly in certain cases. For instance, for some fixed i, there are a lot of different values of x such that different rules $I_x \rightarrow I_i$ occurs (see Figure 2.2). Suppose this situation occurs also at item I_j and item I_k. Thus, items I_i, I_j and I_k will probably obtain high values for their authority weights. If I_i, I_j and I_k are chosen, the profit of the selection may not be high. This is because these items may have no associations among themselves and may have low individual profits, giving a low total profit of the selection.

4. In HAP, the authority weight of an item I_j is independent of the individual profit of item I_j. It only depends on the individual profit of any other item I_i and the confidence of the association rule $I_i \rightarrow I_j$. There

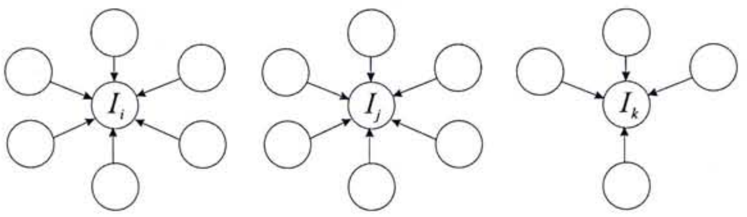

Figure 2.2: Items with High Authority

is a phenomenon that some items with low/zero individual profit have high authority weights. Let us consider an example. In this example, there are three items and their individual profits are shown below. HAP generates the following authority weights.

Item	Authority Weight	Individual Profit
I_i	0.8	\$0
I_j	0.5	\$0
I_k	0.3	\$7

Our question is: if we need to choose TWO items out of three items, which items should be selected?

HAP chooses items I_i and I_j which have the highest authority weights. The estimated profit of item selection $\{I_i, I_j\}$ is \$0+\$0=\$0. However, if we naively choose items I_i and I_k (or I_j and I_k) which have the highest individual profits, the estimated profit of the item selection $\{I_i, I_k\}$ should be equal to or smaller than \$7 [3]. In general, not all profit of item I_k is lost. Thus, the estimated profit of item selection is greater than 0, which gives a greater estimated profit than HAP.

[3]Informally speaking, the total individual profit of items I_i and I_j is equal to \$0+\$7=\$7. However, [49] points out that some individual profit of item I_i and I_k may be lost provided that item I_j is not selected. So, the estimated profit of the item selection is equal to or smaller than the total individual profit of the selected items. Reader can find the details in Section 2.4.

5. Heavy computation and massive storage is required by the cross-selling matrix when the number of items is large.

6. It is not easy to determine a good min-support threshold, and slightly different choices of min-support can greatly affect the result.

2.3 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 [49]. 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 $prof(I_a, t_i)$. [4] 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.

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

[4] 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.

Symbol	**Description**
$t_1, t_2, ..., t_m$	given transactions
$I_1, I_2, ..., I_n$	given items
$prof(I_a, t_i)$	profit of item I_a in transaction t_i
$I = \{I_1, I_2, ..., I_n\}$	Itemset of all items
J	number of selected items
S	set of the selected items
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.*

Example: Suppose a shop carries office equipments of monitors, keyboards, and telephones, with profits of \$1000K, \$100K, \$300K, respectively. 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 \$1300K. 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 \$300K from telephones. If we choose to keep both monitors and keyboards, then the profit can be expected to be \$1100K which is higher. MPIS with the profit as defined in Definition 1 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.

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

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.

2.3.1 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: We shall transform the problem of CLIQUE to the MPIS problem. CLIQUE [22] 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.

1. $J = K$
2. $B = K(K-1)$
3. For each vertex $v \in V$, construct an item.
4. For each edge $e \in E$, where $e = (v_1, v_2)$, create a transaction with 2 items $\{v_1, v_2\}$.
5. 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.

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 are the only transaction 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 csfactor 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 the CLIQUE problem.

Since CLIQUE is a NP-complete problem, the MPIS problem is NP-hard.

$\square$

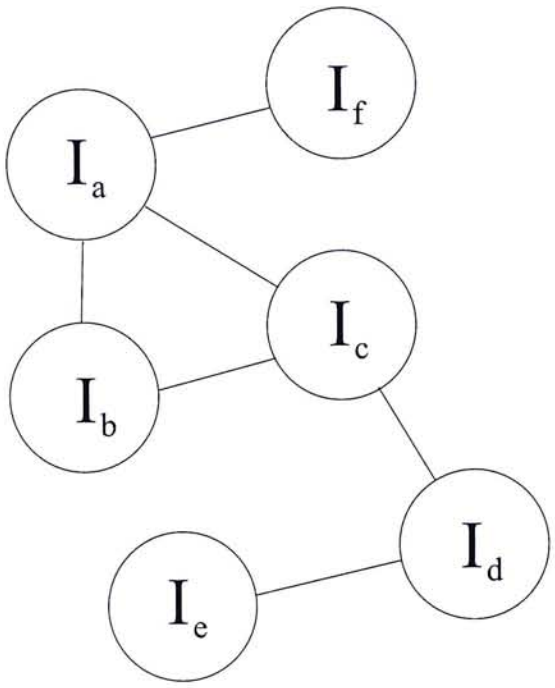

Figure 2.3: An Example of a CLIQUE of size 3

Let us illustrate the above proof with an example. If $n = 6$ and $J = 3$, suppose there is a graph as shown in Figure 2.3 and $prof(I_j, t_i) = 1$.

By the above theorem, we can construct the following transactions:

No.	I_a	I_b	I_c	I_d	I_e	I_f
t_1	1	1	0	0	0	0
t_2	1	0	1	0	0	0
t_3	0	1	1	0	0	0
t_4	1	0	0	0	0	1
t_5	0	1	0	1	0	0
t_6	0	0	0	1	1	0

From the graph, we can find a CLIQUE of size 3 containing I_a, I_b and I_c. $T_c = \{t_1, t_2, t_3\}$. It is easy to see that all profits of items in the transactions from the set T_c contribute to the total profit. The profit from the set T_c is 6. However, for transactions not in set T_c (i.e. t_4, t_5 and t_6), all profits should not be counted. The reason is described as follows. If all transactions t_i contains a selected item and an unselected item, $d_i \neq \phi$ and thus $csfactor(d_i, I_a) = 1$, where I_a is a selected item. For example, as the selection set $S = \{I_a, I_b, I_c\}$, and t_4 contains a selected item I_a and un selected item I_f, $t'_4 = \{I_a\}$ and $d_4 = \{I_f\}$. So, d_4 is non-empty. When $csfactor(d_i, I_a)$ is equal to 1, the profit of the selected item I_a will be removed (See Definition 1). So, all transactions containing one selected item and one unselected item yield zero profit. In other words, any selection of other items will give a lower profit than 6.

2.4 Cross Selling Effect by Association Rules

In Section 2.3, we did not specify how to determine the cross-selling effect $csfactor$ of some items for other items. In previous work [49], 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 where 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, 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*

$$\frac{\textit{no. of transactions containing } I_a \textit{ and any element in } d_i}{\textit{no. of transactions containing } I_a}$$

The numerator in Definition 4 is equal to $g(I_a, d_i)$ defined as follows. This function will be used in quadratic programming for the approximation of the confidence $conf(I_a \rightarrow \diamond d_i)$.

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 $i = 1, 2, .., q$.*

$$g(I_x, D) = \sum_{1 \le i \le q} |I_x Y_i| - \sum_{1 \le i < j \le q} |I_x Y_i Y_j| + \sum_{1 \le i < j < k \le q} |I_x Y_i Y_j Y_k|$$
$$- ... + (-1)^{n+1} |I_x Y_1 Y_2 ... Y_q|$$

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

For instance, there are the following transactions:

Transaction No.	I_a	I_b	I_c
1	1	0	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	0	1

Suppose $S = \{I_a\}$. In transaction t_4, $d_4 = \{I_b, I_c\}$. Let $i = 4$. According to our definition, $conf(I_a \rightarrow \diamond d_i) = \frac{3}{4}$ and $1 - conf(I_a \rightarrow \diamond d_i) = \frac{1}{4}$. In Definition 3, there is a term $1 - conf(I_a \rightarrow \diamond d_i)$. We are interested to know that the ratio of $\frac{1}{4}$ is the chance that item I_a will still be purchased even if items I_b and I_c are both absent.

In [49], the definition of csfactor is determined by $conf(\neg d_i \rightarrow \neg I_a)$ (shown as follows).

Definition 6 Previous Definition of Confidence of Loss Rule: *$conf(\neg d_i \rightarrow \neg I_a)$ is computed as*

$$\frac{\textit{no of trans. not containing } I_a \textit{ and not containing all items in } d_i}{\textit{no of trans. not containing all items in } d_i} \tag{2.6}$$

$$= \frac{\textit{total no of trans. - no of trans. containing item } I_a \textit{ or all items in } d_i}{\textit{total no of trans. - no of trans. all items in } d_i} \tag{2.7}$$

□

Our new Definition 4 is more appropriate and meaningful than the previous Definition 6 used in [49]. In Definition 6, there is a negation term $\neg d_i$, which is not easily understandable. There are some possible meanings of the term $\neg d_i$. For instance, if $d_i = \{I_1, I_2, ...I_q\}$, one of the meanings of $\neg d_i$ is $\neg I_1 \wedge \neg I_2 ... \wedge \neg I_q$ while the other meaning is $\neg I_1 \vee \neg I_2 ... \vee \neg I_q$. The term $\neg I_1 \wedge \neg I_2 ... \wedge \neg I_q$ means that all items should not exist simultaneously. This meaning is just the same interpretation as our definition but this notation is too complicated to present our idea and we choose the more straightforward approach (Definition 4). On the other hand, the term $\neg I_1 \vee \neg I_2 ... \vee \neg I_q$ means at least one item should not exist. The idea comes from the negation of the set d_i, where set d_i can be interpreted as $I_1 \wedge I_2 ... \wedge I_q$. Then, a negation of d_i will be equal to $\neg I_1 \vee \neg I_2 ... \vee \neg I_q$. In this thesis, we adopt the former (i.e. $\neg I_1 \wedge \neg I_2 ... \wedge \neg I_q$). This is because it is more reasonable to interpret all unselected items are missing at the same time and this should be represented by $\neg I_1 \wedge \neg I_2 ... \wedge \neg I_q$. However, the meaning of $\neg I_1 \vee \neg I_2 ... \vee \neg I_q$ is adopted is not straightforward as the phrase contains the concept that at least one item is missing. So, in this thesis, the former with the more straightforward meaning is adapted.

The computation time by using Definition 6 is much greater compared with that by using our new Definition 4 if an FP-tree is used (which will be described in Section 2.6.3).

2.5 Quadratic Programming Method

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 [25]. 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. First we shall show how the problem can be translated to a quadratic programming problem. Let $s = (s_1, s_2, ..., s_n)$ 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 written in 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 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.

Before presenting the transformation of the problem to the quadratic form, we are going to give the following lemmas which will be used in Theorem 2. Details of these lemmas can be found in [35].

Lemma 1 *Let v_i and x be vectors of the same dimension. If m is a positive integer, then*

$$\sum_{i=1}^{m}(v_i^T x) = (\sum_{i=1}^{m} v_i^T)x$$

$\square$

Lemma 2 *Let M_i be an n by n matrix and x be a vector of length n. If m is a positive integer, then*

$$\sum_{i=1}^{m}(x^T M_i x) = x^T(\sum_{i=1}^{m} M_i)x$$

$\square$

The following lemma says that a matrix can be transformed to a symmetric matrix when the matrix M is in the form $x^T M x$.

Lemma 3 *Let M be an n by n matrix and x be a vector of length n. There exists an n by n symmetric matrix Q such that*

$$x^T M x = x^T Q x$$

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

Now, we are going to describe some notations used in Theorem 2.

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

$$d_{ij} = t_{ij} - t'_{ij}.$$

Let n_i be the total number of transactions containing item I_i. Let n_{ij} be the total number of transactions containing items I_i and I_j. Note that $n_{ij} = n_{ji}$.

Symbol	**Description**
s	Binary vector representing which items are selected in the set S
t_i	Binary vector representing which items are in the transaction t_i
t'_i	Binary vector representing which items are selected in the set S in the transaction t_i
d_i	Binary vector representing which items are not selected in the set S in the transaction t_i
n_i	Total number of transactions containing item I_i
n_{ij}	Total number of transactions containing items I_i and I_j
$g(I_x, D)$	the number of transactions containing the item I_x and at least one item in the set D

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

The above observation is based on the principle of inclusion-exclusion in set theory. We define $conf(I_j \to \diamond d_i)$ as

$$\frac{\text{no. of transactions containing } I_j \text{ and at least one item in } d_i}{\text{no. of transactions containing item } I_j}$$

.

$$
\begin{aligned}
& conf(I_j \rightarrow \diamond d_i) \\
= & \frac{\text{no. of transactions containing } I_j \text{ and at least one item in } d_i}{\text{no. of transactions containing item } I_j} \\
= & \frac{g(I_j, D)}{\text{no. of transactions containing item } I_j} \\
& \qquad\qquad \text{where } D \text{ is the set containing all elements in } d_i \\
= & \sum_{1 \le k \le n} |I_j I_k| \times d_{ik} - \sum_{1 \le k < k' \le n} |I_j I_k I_{k'}| \times d_{ik} d_{ik'} \\
& + \sum_{1 \le k < k' < k'' \le n} |I_j I_k I_{k'} I_{k''}| \times d_{ik} d_{ik'} d_{ik''} - \ldots \\
& + (-1)^{n+1} |I_j I_1 I_2 \ldots I_n| \times d_{i1} d_{i2} \ldots d_{in} \times \\
& \frac{1}{\text{no. of transactions containing item } I_j} \quad \text{(by Def. 5)} \\
\approx & \min\left(\frac{\sum_{1 \le k \le n} |I_j I_k| \times d_{ik}}{\text{no. of transactions containing item } I_j}, 1 \right) \\
= & \min\left(\frac{\sum_{k=1}^{n} d_{ik} n_{jk}}{n_j}, 1 \right)
\end{aligned}
$$

The reason why the above approximation is acceptable is that the number of transactions containing a set of items $\mathcal{J}$ is 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.

Theorem 2 *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.

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

and

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

Proof:

$$\begin{aligned}
P &= \sum_{i=1}^{m} \sum_{I_j \in t'_i} prof(I_j, t_i)(1 - conf(I_j \to \diamond d_i)) \qquad \text{(by Def. 1)} \\
&\approx \sum_{i=1}^{m} \sum_{j=1}^{n} t'_{ij} prof(I_j, t_i) \left(1 - \frac{\sum_{k=1}^{n} d_{ik} n_{jk}}{n_j}\right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} s_j prof(I_j, t_i) \left(1 - \frac{\sum_{k=1}^{n} (t_{ik} - t'_{ik}) n_{jk}}{n_j}\right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} s_j prof(I_j, t_i) \left(1 - \frac{\sum_{k=1}^{n} (t_{ik} - t_{ik} s_k) n_{jk}}{n_j}\right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} s_j prof(I_j, t_i) \left(1 - \frac{1}{n_j} \sum_{k=1}^{n} (t_{ik} n_{jk} - t_{ik} n_{jk} s_k)\right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} s_j prof(I_j, t_i) \left(1 - \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk} + \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk} s_k\right) \\
&= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} t_{ij} s_j prof(I_j, t_i) \left[1 - \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk}\right]\right. \\
&\qquad \left. + \sum_{j=1}^{n} t_{ij} s_j prof(I_j, t_i) \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk} s_k\right) \\
&= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} t_{ij} prof(I_j, t_i) \left[1 - \frac{1}{n_j} \sum_{k=1}^{n} t_{ik} n_{jk}\right] s_j\right. \\
&\qquad \left. + \sum_{j=1}^{n} s_j \sum_{k=1}^{m} \frac{t_{ij} prof(I_j, t_i) t_{ik} n_{jk}}{n_j} s_k\right) \\
&= \sum_{i=1}^{m} \left(a_i^T s + \frac{1}{2} s^T H_i s\right)
\end{aligned}$$

where

$$\begin{aligned}
a_i &= g = (g_j | g_j = t_{ij} prof(I_j, t_i)(1 - \frac{1}{n_j}\sum_{k=1}^{n} t_{ik} n_{jk}) \\
&\qquad \text{for } j = 1, 2, ..., n)^T \\
H_i &= A = (a_{jk} | a_{jk} = \frac{2 t_{ij} prof(I_j, t_i) t_{ik} n_{jk}}{n_j} \text{ for } j, k = 1, 2, ..., n)
\end{aligned}$$

$$\begin{aligned}
P &= \sum_{i=1}^{m} (a_i^T s + \frac{1}{2} s^T H_i s) \\
&= \sum_{i=1}^{m} (a_i^T s) + \sum_{i=1}^{m} (\frac{1}{2} s^T H_i s) \\
&= (\sum_{i=1}^{m} a_i^T) s + \frac{1}{2} s^T (\sum_{i=1}^{m} H_i) s \quad \text{(by Lemma 1 and Lemma 2)} \\
&= f^T s + \frac{1}{2} s^T H s
\end{aligned}$$

where

$$\begin{aligned}
f &= \sum_{i=1}^{m} a_i = (f_j | f_j = \sum_{i=1}^{m} t_{ij} prof(I_j, t_i)(1 - \frac{1}{n_j}\sum_{k=1}^{n} t_{ik} n_{jk}) \\
&\qquad \text{for } j = 1, 2, ..., n)^T \\
H &= \sum_{i=1}^{m} H_i = (h_{jk} | h_{jk} = \sum_{i=1}^{m} \frac{2 t_{ij} prof(I_j, t_i) t_{ik} n_{jk}}{n_j} \\
&\qquad \text{for } j, k = 1, 2, ..., n)
\end{aligned}$$

□

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.*

Proof:

$$\begin{aligned}
P &= f^T s + \frac{1}{2} s^T H s \\
&= f^T s + \frac{1}{2} s^T Q s \qquad \text{(by Lemma 3)}
\end{aligned}$$

where

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

□

Since the value of s_i is either 0 or 1, from the above corollary, we have transformed the problem of MPIS to that of 0-1 quadratic programming with an equality constraint of $\sum_i s_i = J$. It is shown in [29] that such a problem is polynomially reducible to an unconstrained binary quadratic programming problem. An unconstrained binary quadratic programming problem can be transformed to a binary linear programming problem (zero-one linear programming) [14]. Zero-one linear programming and quadratic programming are known to be NP-complete [44]. [36] shows that any unconstrained binary quadratic programming problem can be converted to a positive definite unconstrained binary quadratic programming problem. It is known that non-convex quadratic programming is NP-hard [27]. However, there exist programming tools which can typically return good results within a reasonable time for moderate problem sizes.

Example 2 (QP Example) *Let us use the transactions in Table 2.1 to illustrate the use of quadratic programming (QP) in our problem. Remember that, in quadratic programming, we are using vector representation to denote* $t_i = (t_{i1}\ t_{i2}\ t_{i3}\ t_{i4}\ t_{i5}\ t_{i6})^T$. *For instance,* $t_1 = (0\ 0\ 0\ 0\ 0\ 1)^T$. $t_{11} = t_{12} = t_{13} = t_{14} = t_{15} = 0$ *and* $t_{16} = 1$.

The average profit of each item is shown in Table 2.3.

From the transactions, we can find the number of occurrences of each item, n_i, *and the number of co-occurrences of items* I_i *and* I_j, n_{ij}.

The number of occurrences of each item, n_i, *is shown in the following table.*

i	n_i
1	2
2	5
3	2
4	5
5	2
6	1

The non-zero count of co-occurrences of items I_i and I_j, n_{ij} is shown in Table 2.4.

Vector f is:

$$f = \begin{pmatrix} 0 \\ 21 \\ 0 \\ 4 \\ 19 \\ 2 \end{pmatrix}$$

Let us use f_5 for illustration. In the formula

$$f_5 = \sum_{i=1}^{10} t_{i5}\ prof(I_5, t_i)(1 - \frac{1}{n_5}\sum_{k=1}^{6} t_{ik} n_{5k})$$

$t_{i5} = 1$ when transaction t_i contains item I_5. Otherwise, $t_{i5} = 0$. We need to consider transaction t_4 and t_8.

$$\begin{aligned} f_5 &= t_{45}\ prof(I_5, t_4)(1 - \frac{1}{n_5}\sum_{k=1}^{6} t_{4k} n_{5k}) \\ &\quad + t_{85}\ prof(I_5, t_8)(1 - \frac{1}{n_5}\sum_{k=1}^{6} t_{8k} n_{5k}) \\ &= 19 \times (1 - \frac{1}{2} \times n_{45}) + 19 \times (1 - \frac{1}{2} \times n_{25}) \\ &= 19 \times (1 - \frac{1}{2} \times 1) + 19 \times (1 - \frac{1}{2} \times 1) \\ &= 19 \end{aligned}$$

The computed matrix H is shown as follows.

$$H = \begin{pmatrix} 0 & 80 & 0 & 0 & 0 & 0 \\ 11.2 & 0 & 2.8 & 2.8 & 2.8 & 0 \\ 0 & 20 & 0 & 80 & 0 & 0 \\ 0 & 0.4 & 1.6 & 0 & 0.4 & 0 \\ 0 & 19 & 0 & 19 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In the above matrix, we use h_{54} for illustration. The formula is $h_{54} = \sum_{i=1}^{m} \frac{2t_{i5}prof(I_5,t_i)t_{i4}n_{54}}{n_5}$. We can observe that the term $\frac{2t_{i5}prof(I_5,t_i)t_{i4}n_{54}}{n_5}$ will be non-zero if t_{i4} and t_{i5} are both 1. This case occurs in transaction t_4. So, $h_{54} = \frac{2prof(I_5,t_4)n_{54}}{n_5} = \frac{2\times19\times1}{2} = 19$.

Actually, the matrix Q (shown as follows) is similar to matrix H but has a symmetric property. The entry $q_{45} = \frac{h_{45}+h_{54}}{2} = \frac{0.4+19}{2} = 9.7$

$$Q = \begin{pmatrix} 0 & 45.6 & 0 & 0 & 0 & 0 \\ 45.6 & 0 & 11.4 & 1.6 & 10.9 & 0 \\ 0 & 11.4 & 0 & 40.8 & 0 & 0 \\ 0 & 1.6 & 40.8 & 0 & 9.7 & 0 \\ 0 & 10.9 & 0 & 9.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

By using a constrained optimization tool, the maximization problem with the objective $P = f^Ts + \frac{1}{2}s^TQs$ subject to $\sum_{i=1}^{n} s_i = J$ can be solved. The solution vector s is found as follows. The objective function $f^Ts + \frac{1}{2}s^TQs$ is maximized to 96.5 as $f^Ts = 40$ and $s^TQs = 113$.

$$s = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Then, by using the exact evaluation of the objective of the total profit of item selection, the total profit is \$102.1.

□

2.6 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 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.

Definition 7 (Individual Count) *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.

In our algorithm we shall make use of the average profits of items over the transactions.

Definition 8 Average profit and Individual profit: *Let Z_k be the set of transactions that contain I_k, the average profit is given by*

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

Individual profit *of an item I_k is equal to $c_k \times p_k$.* □

We are going to introduce another definition.

Definition 9 *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.

The following table gives a summary of the symbols to be used.

Symbol	**Description**
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	The set containing the items associated with item I_i to be selected
$e_{i,j}$	Estimated value of item I_j from item I_i; $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support_{(I_i,I_j)}$
F	Set of frequent itemsets
Z	A set of the items to be selected in Phase 1
$\widehat{P}(A)$	Estimated profit of item selection assuming that the selection set $= A$

2.6.1 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.

In the Main Phase, we shall try to rank the items with consideration of cross-selling.

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” (denoted by set S_i). Set S_i is called the **estimation set** for I_i.

In order to determine set S_i, we calculate the the following values for all j: $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$. $e_{i,j}$ 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 a high individual profit, this value will be higher. 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: $b_i = \widehat{P}(S_i \cup \{I_i\})$.

$$b_i = \sum_{j=1}^{m} \sum_{I_a \in t'_j} prof(I_a, t_j)(1 - conf(I_a \to \diamond d_j))$$

assuming the output selection set $S = S_i \cup \{I_i\}$, and d_j is the set of items in t_j that are not selected in S.

The item benefit of an item I_i models the significance of I_i in terms of profits, with the consideration of the cross-selling effect. Suppose the output set $S = S_i \cup \{I_i\}$. That is, the output set contains item I_i and all items in S_i which are estimated to have the greatest contributions to the profits when these items co-occur with item I_i.

After we obtain the item benefits, items can be ranked by their item benefits. Then 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.

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. **Item Occurrence Counting -** count the number of occurrences of each item, $n_1, n_2, ..., n_n$. We can simply keep a variable n_i to store the number of occurrence of each item I_i for $i = 1, 2, ..., n$ in the data set.

2. **Individual Count Derivation -** obtain the individual count for each item, $c_1, c_2, ..., c_n$

   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.

3. **Size 2 Itemset Generation -** generate all size 2 itemsets, with their counts.

Main Phase

In the Main Phase, we shall try to rank the items with consideration of cross-selling. For each item, we find $J-1$ "best" items in its neighborhood (denoted by set S_i) and assume that these $J-1$ "best" items are selected, then we can estimate the resulting profit assuming that only these items are selected. Items can therefore be ranked and selected accordingly.

1. **Estimation Set Creation -** In order to estimate the profit generated from an item, the cross-selling factor should be considered, which depends on a chosen set S. Since S does not exist initially, we shall estimate a chosen item set for each item. For item I_i the estimated set of chosen items is given by S_i. S_i contains $J-1$ items in the neighbourhood of I_i.

   In this step, the sets, $S_1, S_2, ..., S_n$ are computed. For each item I_i we do the following: 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 associated I_j items, choose $J-1$ items with the highest estimated values. Put these items into the set S_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 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.

2.6.2 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.

Essentially H_i is the total profit difference caused by the addition of the item I_i to the selection set of S_i, subtracting the profit of I_i when only I_i is selected. For all items in S_i, this difference is always an increase greater than or equal to zero. 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 4 *$H_i \geq L_i$.*

Item I_i is pruned if H_i is smaller than the 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 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.

2.6.3 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.

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 $n_1, ..., n_n$.

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 need.

Item Benefit Calculation

The item benefit b_i is defined as: $b_i = \widehat{P}(S_i \cup \{I_i\})$.

That is,

$$b_i = \sum_{j=1}^{m} \sum_{I_a \in t'_j} prof(I_a, t_j)(1 - conf(I_a \to \diamond d_j))$$

assuming the output selection set $S = S_i \cup \{I_i\}$.

The benefit of an item I_i is to model the amount of benefit an item I_i can give, with the consideration of the cross-selling effect. Suppose the output set $S = S_i \cup \{I_i\}$. That is, the output set contains item I_i and all items in S_i which are estimated to have the greatest contributions to the profits when these items co-occur with item I_i.

In MPIS_Alg, most of the computation time is contributed from the calculation of the function $\widehat{P}(A)$ in "Item Benefit Calculation Step" and "Enhancement Step".

In our implementation, we utilize two efficient data structures called an FP-tree (see Figure 2.4) and a variation of an FP-tree called an *FP-MPIS-tree* (see Figure 2.5). (NOTE: The FP-tree described here assumes a minsupport of Ø.) Typically an FP-tree is a compressed form of the database and thus we can access an FP-tree efficiently. In MPIS_Alg, we only need to construct the FP-tree once in Preparation Phase for the access of compressed form of the database in the later steps. In the IBM synthetic data set, on average, an FP-tree is constructed in only 1.5 seconds.

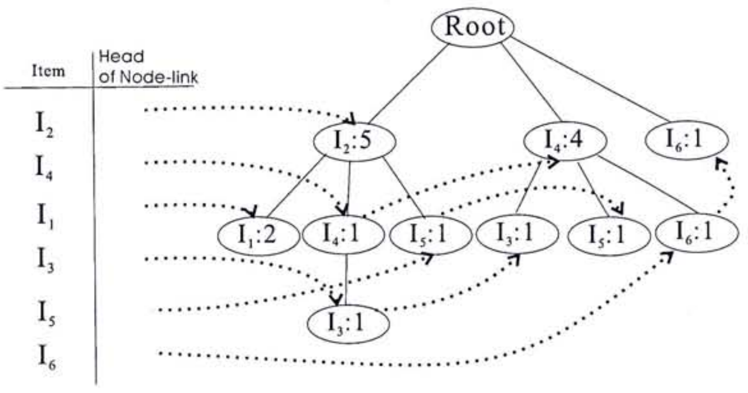

Figure 2.4: An example of an FP-tree

An FP-MPIS-tree has the similar compressed property as an FP-tree. This tree will be discussed later in this section. This tree may be constructed for the calculation of $\widehat{P}(A)$. For the IBM synthetic data set that we use (which contains 10,000 transactions), this tree can be constructed in 0.02 seconds on average. It requires less computation time compared with an FP-tree as this tree involves fewer transactions (which will be described later in this section). After the calculation of $\widehat{P}(A)$, this tree will be removed.

Before describing how to compute $\widehat{P}(A)$, we are going to describe the following first. Whenever we need to evaluate $\widehat{P}(A)$, we first read all transactions from an FP-tree and build an FP-MPIS-tree from the transactions just read.

1. How to read all transactions from an FP-tree
2. How to calculate some of the profit of the item selection in a variation of an FP-tree called an *FP-MPIS-tree*

1. **Reading Transactions from an FP-tree**

   In this section, we are going to describe how we can read all transactions in an FP-tree T. Starting from a leaf node $node_l$ of T, we can read

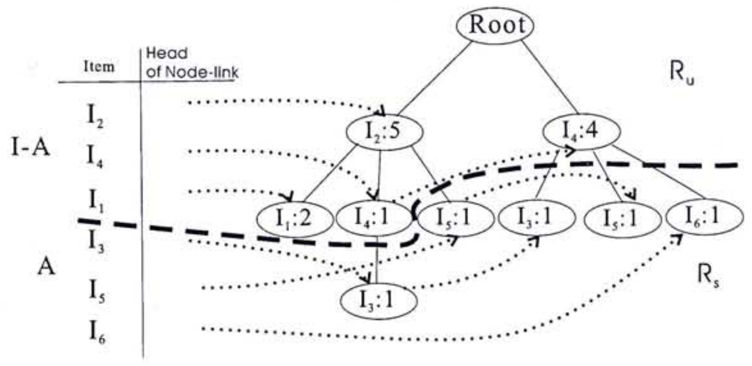

Figure 2.5: An example of an FP-MPIS-tree (The set of selected items is $A = \{I_3, I_5, I_6\}$ and the set of unselected items is $I - A = \{I_1, I_2, I_4\}$, where $I = \{I_1, I_2, ..., I_6\}$)

transaction t containing all items associated with nodes M, where M is the set of all nodes visited when we traverse from node $node_l$ to the root in T. Suppose the node count (or occurrence) stored at nodes is $count_l$. Such transaction occurs $count_l$ times in database. Let us illustrate it with a transaction in Figure 2.4. Consider the leftmost node containing I_1 in the figure. We first visit the leaf node of the tree, says the leftmost node containing I_1. We can obtain a transaction $\{I_1, I_2\}$ with count 2. We traverse the node from that node containing I_1 to the root. As we visit two nodes with items I_1 and I_2, we know that there is a transaction $\{I_1, I_2\}$. Besides, as the count stored in the node is 2, the nodes in the path traversed should co-exist two times. In short, a transaction $\{I_1, I_2\}$ occurs two times.

After reading transaction t, we will decrement the node counts of all nodes from the node $node_l$ to the root by $count_l$. If the node counts of any nodes are decremented to zero, this means that the nodes are logically removed. In our implementation, we do not need to remove the nodes physically. We just need to check the node count to determine whether the node is removed logically. [5] For instance, in Figure 2.4, after fetching the transaction $\{I_1, I_2\}$, we decrement all counts of the nodes in the path traversed. The count of the node I_1 is decremented by 2 and becomes 0. This node becomes logically removed. Besides, the count of the parent node is decremented by 2 and becomes 3.

We can read all other transactions t' from the resulting FP-tree by traversing all nodes with non-zero node count in post-order ordering. The reason for the post-order traversal is to make sure that (logical) leaf nodes are processed before internal nodes. Now, we know how to read all transactions from an FP-tree.

2. **Calculating Profit in an FP-MPIS-tree**

   We are going to describe an *FP-MPIS-tree*, a variation of an FP-tree. This tree may be constructed for the calculation of $\widehat{P}(A)$.

   As stated in [23], the FP-tree construction requires an ordering of items according to the frequency of items. That means more frequent items are inserted into the FP-tree near to the root. In the FP-MPIS-tree, we need to divide the items into two sets, $I - A$ and A. The items in set $I - A$ are inserted into the FP-MPIS-tree near to the root, compared with items in set A. Similar to the FP-tree, the ordering of items in each set in an FP-MPIS-tree is based on the frequency of items.

   Until now, we have not mentioned when this FP-MPIS-tree is con-

[5] As we have such implementations for the avoidance of the removal of nodes, each node will contain an additional variable to store the node count duplicately, which is equal to the original node count of the node. This extra variable is used for the further access of an FP-tree such that the FP-tree can be re-used.

structed during the computation of $\widehat{P}(A)$. But, we can say that the FP-MPIS-tree is constructed for all transactions containing both selected items (i.e. items in set A) and unselected items (i.e. items in set $I-A$), which will be described later in this section. At this moment, we assume that the FP-MPIS-tree contains only all such transactions. With this assumption, we can get the information $conf(I_a \rightarrow \diamond d_j)$ efficiently from such tree.

In the FP-MPIS-tree, we traverse all nodes corresponding to selected items (i.e. items in set A). At each node $\overline{N}$ with node count $count_n$, we can deduce that there are $count_n$ transactions containing all items associated with nodes $\overline{M}$, where $\overline{M}$ is the set of the nodes visited by traversing from node $\overline{N}$ to the root. With our assumption, $\overline{M}$ should contain some selected items t' and some unselected items d.

Let $\overline{P}$ be a variable storing the accumulated increment in the following calculation. For each such node $\overline{N}$ corresponding to the selected item I_a with node count $count_n$, we can increment $\overline{P}$ by $p_a \times (1 - \frac{g(I_a,d)}{n_a}) \times count_n$, where p_a is the average profit of item I_a and n_a is the frequency of item I_a. The reasons of such increment is explained as follows.

The increment corresponds to the profit of item I_a in $count_n$ transactions T. For each transaction of T, we can calculate $p_a \times (1 - conf(I_a \rightarrow \diamond d))$ by $p_a \times (1 - \frac{g(I_a,d)}{n_a})$ (See Definition 4 and 5). As there are $count_n$ such transactions T, multiplying this term by $count_n$ gives the increase.

We can do the above increment for all nodes associated with selected items by using post-order traversal, Such traversal ordering makes our algorithm efficient. As the transactions obtained by traversing from any node $\overline{N}'$ in the subtree under node $\overline{N}$ is associated with the same unselected set d, we do not need to find the set d again for all nodes $\overline{N}'$.

After processing all increments, $\overline{P}$ is the profit of all selected items in all transactions containing both selected items and unselected items.

We have not yet described how to compute the number of transactions containing item I_a and any items in set d, $g(I_a, d)$. Now we are going to describe how we compute this value in the FP-MPIS-tree.

To compute $g(I_a, d)$, we first look up the horizontal linked list (dotted links in Figure 2.5) of each item I_a in the FP-MPIS-tree. For each node Q in the linked list, call the function parseFPTree(Q, 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 the 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. This count is the number of transactions containing item I_a and at least one item in set d.

It is noted that 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.

After describing how to compute $g(I_a, d)$, we can compute the profit of item I_a in transaction t.

Now, we know how to read all transactions from an FP-tree and how to calculate $\overline{P}$ in an FP-MPIS-tree. We are going to describe how to compute $\widehat{P}(A)$ by using the above two methods just described.

We divide the transactions into 3 groups. The first group is those containing only selected items. The second group is those containing only unselected items. The third group is those containing both selected items and unselected items. Let $\widetilde{P}$ be a variable storing the final output of $\widehat{P}(A)$.

Every time we read transaction t in the FP-tree which occurs $count_n$ times, we may increment $\widetilde{P}$ according to the group that the transaction t belongs to. If t belongs to the first group, $\widetilde{P}$ is incremented by $(\sum_{I_a \in t} p_a) \times count_n$ as all items are selected in the transaction. If t belongs to the second group , we do not need to update $\widetilde{P}$ as no items are selected in this transaction. If t belongs to the third group, we will insert t with node count $count_n$ into the FP-MPIS-tree.

After reading all transactions from an FP-tree, we have updated $\widetilde{P}$ and the FP-MPIS-tree. $\widetilde{P}$ is now equal to the total profit of selected items in all transaction containing only selected items. An FP-MPIS-tree is constructed from all transactions containing some selected items and some unselected items. As we mentioned before, we can calculated $\overline{P}$ from the FP-MPIS-tree. Thus, we can compute $\widehat{P}(A)$ by adding $\widetilde{P}$ and $\overline{P}$ together.

Item Benefit Update

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 benefits b_i also have to be updated.

The update is computed as follows. Let S_i' be the selection containing the items in both the selection before item I_x is removed and the selection after item I_x is removed. Thus, $S_i' \cup \{I_x\}$ is the selection before we remove item

Table 2.4: Count of co-occurrence of items I_i and I_j, n_{ij}

i	j	$n_{ij}(=n_{ji})$
1	2	2
2	3	1
2	4	1
2	5	1
3	4	2
4	5	1

I_x while $S'_i \cup \{I_k\}$ is the selection after we remove item I_x and add item I_k in the selection S_i. We just need to do the item benefit update by scanning only those transactions $\mathcal{T}$ containing item I_x or item I_k. Let $\widehat{P}'(A, \mathcal{T})$ be the profit of the item selection in transactions $\mathcal{T}$. Thus, the item benefit is updated as follows.

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

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 with the whole database, the update can be done very efficiently.

Example 3 (MPIS_Alg Example) *We are going to illustrate algorithm MPIS_Alg by using the transactions and profits in Table 2.1 and Table 2.3.*

The non-zero count of co-occurrence of items I_i and I_j, n_{ij} is shown in Table 2.4. Other values not shown in the table are zero.

The individual count of each item is shown as follows. If the min-support is 0, the individual count, c_i can be regarded as the number of transactions *containing only item I_i. For instance, $c_4 = 2$ as there are two transactions (t_2 and t_7) containing only item I_4.*

i	c_i
1	*0*
2	*1*
3	*0*
4	*2*
5	*0*
7	*1*

The estimated value and the estimation set of each item are shown as follows. Let us take $e_{1,2}$ for illustration.

$$\begin{aligned} e_{1,2} &= p_2 \times c_2 + (p_1 + p_2) \times support_{(I_1,I_2)} \\ &= p_2 \times c_2 + (p_1 + p_2) \times n_{12} \\ &= 7 \times 1 + (20 + 7) \times 2 \\ &= 61 \end{aligned}$$

$e_{i,j}$ $i\backslash j$	*1*	*2*	*3*	*4*	*5*	*6*	S_i
1	*0*	*61*	*0*	*2*	*0*	*2*	I_2, I_4
2	*54*	*7*	*27*	*10*	*26*	*2*	I_1, I_3
3	*0*	*34*	*0*	*44*	*0*	*2*	I_2, I_4
4	*0*	*15*	*42*	*2*	*20*	*2*	I_3, I_5
5	*0*	*33*	*0*	*22*	*0*	*2*	I_2, I_4
6	*0*	*7*	*0*	*2*	*0*	*2*	I_2, I_4

The values of L_i and H_i are shown in the following table. We use L_5 and H_5 to illustrate how to calculate the following table. There are only transactions t_4 and t_8 containing item I_5. Calculating L_5 involves the assumption of the set $S = \{I_5\}$. In transaction t_4, d_4 contains I_5. $conf(I_5 \rightarrow \diamond d_4) = \frac{1}{2}$. The *term* $prof(I_5, t_4)(1 - conf(I_5 \rightarrow \diamond d_4)) = 19 \times (1 - \frac{1}{2}) = 9.5$. *In transaction* t_8*, similarly, the term* $prof(I_5, t_8)(1 - conf(I_5 \rightarrow \diamond d_8)) = 19 \times (1 - \frac{1}{2}) = 9.5$. *Thus,* $L_5 = 9.5 + 9.5 = 19$. *Consider* H_5. *The output set* $S = \{I_2, I_4, I_5\}$ *is assumed. In transaction* t_4 *and* t_8, d_4 *and* d_8 *do not contain items. The terms* $conf(I_5 \rightarrow \diamond d_4)$ *and* $conf(I_5 \rightarrow \diamond d_8)$ *are equal to 0. So, the terms* $prof(I_5, t_4)(1 - conf(I_5 \rightarrow \diamond d_4)) = prof(I_5, t_8)(1 - conf(I_5 \rightarrow \diamond d_8)) = 19$. *Thus,* $H_5 = 19 + 19 = 38$.

i	L_i	H_i
1	0	40
2	26.60	32.2
3	0	40
4	4	4.8
5	19	38
6	2	2

The J-th (i.e. 3rd) largest values is 4. The item I_6 *should be pruned in the pruning step as* $H_6 < 4$. *So, there are five remaining items,* I_1, I_2, I_3, I_4 *and* I_5. *As all estimation sets* S_i *do not contain item* I_6*, there is no need to update the estimation sets. So, this time,* $b_i = H_i$. *The result can be found in Table 2.5.*

Item I_4 *is removed as* $b_4 = 4.8$ *is the smallest* b_i *value. Then, there are four remaining items* I_1, I_2, I_3 *and* I_5. *As* S_1, S_3 *and* S_5 *contain item* I_4*, item* I_4 *has to be replaced with the item which has not been selected with the greatest estimate value. Consider* S_1*, the item (not including item* I_1 *itself) which has not been selected with the greatest estimate value (i.e.* $e_{1,3} = e_{1,5} = 0$*) is* I_3 *or* I_5. *We arbitrarily choose item* I_3 *to replace item* I_4. *After the estimate set* S_i *is updated, the item benefit* b_i *is updated. The result is shown as follows.*

Table 2.5: Benefit of items calculated after pruning in Algorithm MPIS_Alg

i	S_i	b_i
1	I_2, I_4	40
2	I_1, I_3	32.2
3	I_2, I_4	40
4	I_3, I_5	4.8
5	I_2, I_4	38
6	/	/

i	S_i	b_i
1	I_2, I_3	*40*
2	I_1, I_3	*32.2*
3	I_1, I_2	*0*
4	/	/
5	I_1, I_2	*28.5*
6	/	/

Item I_3 is removed as $b_3 = 0$ is the smallest value. There are three remaining items I_1, I_2, and I_5. Algorithm MPIS_Alg stops here and items I_1, I_2, and I_5 are chosen in the output set. Item benefit b_1 and b_2 have to be computed again as item I_3 is in estimation sets S_1 and S_2.

i	S_i	b_i
1	I_2, I_5	*40*
2	I_1, I_5	*33.6*
3	/	/
4	/	/
5	I_1, I_2	*28.5*
6	/	/

The final total profit of item selection is equal to \$40 + \$33.6 + \$28.5 =

\$102.1.

□

2.7 Genetic Algorithm

Genetic algorithms (GA) are a family of evolution-inspired computational models. The algorithms encode a possible solution of a problem on a chromosome-like data structure. The algorithms first keep track of a number of chromosomes in a population. By utilizing the reformation operators to these chromosome in the population, the algorithm can create different populations in order to preserve critical information in the previous population.

In genetic algorithm (GA), we need to represent the solution of MPIS (i.e. which items should be selected). MPIS is a subset selection problem. In our problem, there are n items in the transactions. 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.

Our genetic algorithm (GA) is an approach of a steady-state GA described in [45]. It also uses a strategy of the weaker-parent replacement described in [38] and [19]. In [38] and [19], this strategy is shown as a good choice of replacement as one could estimate that a parent would be one of the genes in the population which is closest to the child. The outline of our GA is shown in Figure 2.6. In our experiment, we use the number of generations as the stopping criteria described in Figure 2.6.

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 $i = 1, 2, ..., n$. 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).

```
Procedure GA
Input: $N$: population size
         $M$: number of children generated for for each crossover
begin
  generate population $Pop$ of size $N$
  calculate the fitness value of each gene in the population
  find and store the best of the gene according to the fitness value
  repeat
    duplicate the population $Pop$ and assign it to $Pop_{pre}$
    for each gene in the population $Pop_{pre}$
      set this gene as the first parent
      find the second parent in a roulette wheel approach in the population $Pop_{pre}$
      create $M$ children with crossover of the first parent and the second parent
      for each child created
        mutate the child with probability $p_{mutate}$
        evaluate the fitness value of the child
        if the child has a greater fitness value than the weaker parent
        (with smaller fitness value)
          update the population $Pop$ by replacing the weaker parent with the child
          (NOTE:Each successive child in the inner loop is compared with the
          updated weaker parent.)
        if the child has a greater fitness value than the best
          replace the best stored with the child
        Otherwise, the child is discarded.
  until a stopping criteria is reached
end
```

Figure 2.6: Simple Steady-State Genetic Algorithm

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 3. That is, given a set S of selected items of size J, the total profit of item selection P is:

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

We can adopt the computation method of P in Algorithm MPIS_Alg (See Section 2.6.3).

Selection for the Second Parent: In the algorithm, we need to select two parents for crossover. The first parent is selected deterministically but the second parent 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:

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

2.7.1 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.

We have a matrix L. Each entry $L_{i,j}$ in the matrix contains the profit loss of item I_i if item I_j is not selected. We model this entry by

$$profit(I_i) \times \frac{\text{no. of transactions containing } I_i \text{ and } I_j}{\text{no. of transactions containing } I_i}$$

where $profit(I_i)$ is the total profit 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.

Let us give an example for crossover. For instance, suppose $n = 6$ and $J = 3$, and the first parent is 11100 and the second parent is 10011. In Step 1, set $C = \{I_1\}$ as both parents contain item I_1. Set $D = \{I_2, I_3, I_4, I_5\}$. We need to choose two items from D to complete the generation of a child according to a loss matrix L. Assume the loss matrix L is:

$$L = \begin{pmatrix} L_{1,1} & L_{1,2} & L_{1,3} & L_{1,4} & L_{1,5} \\ L_{2,1} & L_{2,2} & L_{2,3} & L_{2,4} & L_{2,5} \\ L_{3,1} & L_{3,2} & L_{3,3} & L_{3,4} & L_{3,5} \\ L_{4,1} & L_{4,2} & L_{4,3} & L_{4,4} & L_{4,5} \\ L_{5,1} & L_{5,2} & L_{5,3} & L_{5,4} & L_{5,5} \end{pmatrix} = \begin{pmatrix} 0.0 & 2.5 & 6.5 & 5.2 & 7.2 \\ 45.2 & 0.0 & 5.9 & 7.5 & 9.5 \\ 5.2 & 5.2 & 0.0 & 2.2 & 7.8 \\ 0.0 & 0.0 & 0.0 & 0.0 & 9.1 \\ 1.0 & 2.3 & 4.5 & 9.2 & 0.0 \end{pmatrix}$$

We can calculate all L_j for $I_j \in D$. $L_2 = 2.5, L_3 = 6.5, L_4 = 5.2, L_5 = 7.2$. We can rank all I_j according to the L_j values. Item I_2, I_3, I_4 and I_5 are ranked with 1, 3, 2 and 4 and are given with probability 0.1, 0.3, 0.2 and 0.4 respectively. In Step 2, assume I_2 and I_4 are chosen randomly in this roulette wheel approach to become the remaining items in the gene of the child. The gene of the child will become 11010.

2.7.2 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.

2.8 Performance Analysis

Let us analyze the running time (RT) of MPIS_Alg. Recall that MPIS_Alg has two phases - Preparation Phase and Main Phase. We divide them into sub-sections to analyze the running time. Besides, the following table shows the description of the symbols used in our analysis.

Symbol	Description
n	number of items
m	number of transactions
J	number of items to be selected
n_{TS}	maximum number of items in the transaction
n_M	maximum number of occurrence of an item
n_{node}	maximum number of nodes representing an item in the FP-tree
n_i	the number of transactions containing item I_i
n_{ij}	the number of transactions containing item I_i and item I_j
n_R	the number of remaining items after the Enhancement Step
n_{fp}	the total number of nodes in the FP-tree

2.8.1 Preparation Phase

In this phase, we need to count the number of occurrences of each item $n_1, n_2, ..., n_n$ and the individual counts of each item $c_1, c_2, ..., c_n$ and to generate all size 2 itemsets. In this Phase, we just need to construct an FP-tree from the transactions. By setting the minsupport as 0 and the maximum itemset size as 2, we can find n_i and size 2 itemsets.

There are two steps in this phase

1. Construction of the FP-tree

2. Mining of itemsets

1. **Step of Construction of the FP-tree**

   In this step, we need to do the following:

   (a) Scan the database and find $n_1, n_2, .., n_n$ and $c_1, c_2, ..., c_n$

   (b) Sort $n_1, n_2, .., n_n$

   (c) Scan the database again and insert each transaction into the FP-tree

We first need to scan the database to obtain the number of occurrences of each item, which takes $O(mn_{TS})$ as there are m transaction and each transaction contains at most n_{TS} items. (It is noted that the maximum transaction size is n_{TS}, but as the transaction size is usually much smaller than n_{TS}, the insertion cost of each transaction requires much smaller cost than n_{TS}.) We give the ordering of the items based on the frequency of the items by sorting $n_1, n_2, ..., n_n$, which takes $O(nlog\ n)$. Then, we need to scan the database again and insert the items for each transaction into the FP-tree. As the tree depth is $O(n_{TS})$, the insertion cost for each transaction is $O(n_{TS})$. As there are m transactions, the running time of inserting the items in transactions is $O(mn_{TS})$. Thus, the overall RT for the step of the construction of the FP-tree is $O(mn_{TS} + nlog\ n + mn_{TS}) = O(nlog\ n + mn_{TS})$.

2. **Step of Mining of Itemsets**

   Before analyzing the RT of this step, we need to give the pseudo-code of procedure FP-growth used in this step in Figure 2.7.

   We start to call FP-growth($Tree, null$), as follows. This call needs $O(n_{TS}^2 + nn_{node}n_{TS})$ time. The RT of the above steps are shown as follows.

   (a) Line (1) requires $O(n_{TS})$ time

   (b) Lines (2)-(3) require $O(n_{TS})$ time

   (c) Line (4)-(8) requires $O(nn_{node}n_{TS})$ time

   In this procedure, we just need to consider two cases:

   (a) **Case 1:** α contains two single items

   (b) **Case 2:** α is a single item

   (c) **Case 3:** α is an empty set

```
Procedure FP-growth($Tree, \alpha$)
{
(1) if $Tree$ contains a single path $P$
(2) then for each combination (denoted as $\beta$)
      of the nodes in the path $P$ do
(3)     generate pattern $\beta \cup \alpha$ with $support =$
            $minimum\ support\ of\ nodes\ in\ \beta$;
(4) else for each $a_i$ in the header of $Tree$ do{
(5)     generate pattern $\beta = a_i \cup \alpha$ with
            $support = a_i.support$;
(6)     construct $\beta$'s conditional pattern based and
            then $\beta$'s conditional FP-tree $Tree_\beta$;
(7)     if $Tree_\beta \neq \emptyset$
(8)     then call FP-growth($Tree_\beta, \beta$) }
}
```

Figure 2.7: Pseudo-code for FP-growth

We just need to consider the above cases as our objective is to find 2-itemsets. Case 3 is just equal to the call of FP-growth($Tree, null$).

(a) **Case 1**

Let us consider the case where α contains two single items

In line (1), as the tree depth is $O(n_{TS})$, the check of a single path P in the tree is $O(n_{TS})$. (NOTE: In the FP-tree, usually, the length of the path is smaller than n_{TS}.) As our objective is to generate 2-itemsets and α contains two single items, lines (2)-(3) can be skipped and lines (2)-(3) require $O(1)$ time. Lines (4)-(8) are skipped as $\beta = a_i \cup \alpha$ contains three single items, which is not our desired output. So, these lines requires $O(1)$ time.

Thus, the overall RT in this case is $O(n_{TS} + 1 + 1) = O(n_{TS})$.

(b) **Case 2**

Let us consider the case where α is a single item.

In line (1), similar as before, the check is $O(n_{TS})$. As our objective is to generate 2-itemsets and there are only $O(n_{TS})$ combinations of the 2-itemsets containing α and β, lines (2)-(3) need $O(n_{TS})$ time.

In line (4), as there are n items, there are $O(n)$ possible headers of $Tree$. For each header, we will do the code in lines (5)-(8). Line (5) needs $O(n_{node})$ time because we need to parse the horizontal link of a_i so as to count the $a_i.support$. (NOTE: In the FP-tree, the number of nodes for a frequent item is much smaller than n_{node} as the FP-tree has the compressed property of the database for much frequent item.)

In line (6), the construction of β's conditional pattern base and then β's conditional FP-tree $Tree_\beta$ requires $O(n_{node}n_{TS})$ time. In this operation, we need to parse the horizontal link of a_i to visit each node. Then, for each node visited, we need to parse vertical links. As there are $O(n_{node})$ nodes and the tree depth is $O(n_{TS})$, the RT for line (6) is $O(n_{node}n_{TS})$.

In lines (7)-(8), suppose $Tree_\beta \neq \emptyset$ (i.e. the worst case for these lines). We need to call FP-growth($Tree_\beta, \beta$), where β contains two items. This is the case mentioned before (i.e. **Case 1**). The RT for this case is $O(n_{TS})$.

The code for lines (4)-(8) requires $O(n \times (n_{node} + n_{node}n_{TS} + n_{TS})) = O(n \times n_{node}n_{TS}) = O(nn_{node}n_{TS})$.

Therefore, the overall RT in this case is $O(n_{TS} + n_{TS} + nn_{node}n_{TS}) = O(nn_{node}n_{TS})$.

(c) **Case 3**

Now, we consider the case where α is an empty set.

In line (1), the check is similar as before and the time complexity is equal to $O(n_{TS})$.

For lines (2)-(3), for this case, we just need to find 2-itemsets. As the single path P has the length $O(n_{TS})$, there are $O(C_2^{n_{TS}}) = O(n_{TS}^2)$ combinations. (NOTE: As we discussed before, the length of the path is usually smaller than n_{TS}, the number of combinations is also smaller than $C_2^{n_{TS}}$.) As line (3) takes $O(1)$ time, the RT for lines (2)-(3) is $O(n_{TS}^2)$.

In line (4), similarly, there are $O(n)$ possible headers of $Tree$.

In a similar way, line (5) and line (6) require $O(n_{node})$ time and $O(n_{node}n_{TS})$, respectively.

Suppose we encounter the worst-case in lines (7)-(8). As β contains one item, we need to call FP-growth($Tree_\beta, \beta$) recursively. This is just equal to Case 2. The RT for this case is $O(nn_{node}n_{TS})$.

The code for lines (4)-(8) is $O(n \times (n_{node} + n_{node}n_{TS} + nn_{node}n_{TS})) = O(n^2 n_{node}n_{TS})$.

Thus, the overall RT is $O(n_{TS}+n_{TS}^2+n^2n_{node}n_{TS}) = O(n_{TS}^2+n^2n_{node}n_{TS})$.

As a result, the RT of the step of the construction of the FP-tree and the step of mining of itemsets are $O(nlog\ n + mn_{TS})$ and $O(n_{TS}^2 + n^2n_{node}n_{TS})$ respectively.

In short, the RT of these two steps is:

(a) Construction of the FP-tree (which requires $O(nlog\ n + mn_{TS})$ time)

(b) Mining of itemsets (which requires $O(n_{TS}^2 + n^2n_{node}n_{TS})$ time)

Thus, the overall RT for the preparation phase is $O(nlog\ n + mn_{TS} + n_{TS}^2 + n^2n_{node}n_{TS})$. Typically, $n_{TS} << m$ and thus $n_{TS}^2 = O(mn_{TS})$, the overall RT is $O(nlog\ n + mn_{TS} + n^2n_{node}n_{TS})$.

2.8.2 Main Phase

In this phase, we have the following steps:

1. Estimation Set Creation
2. Item Pruning
3. Item Benefit Calculation
4. Item Selection and Item Benefit Update & Iteration

1. **Estimation Set Creation**

   There are the following sub-steps:

   (a) Calculation of Estimated Value

   (b) Selection of $J-1$ items with the Highest Estimated Value for set S_i

   (a) **Calculation of Estimated Value**

   For each item, n estimated values have to be calculated. For each estimated value $e_{i,j} = p_j \times c_j + (p_j + p_i) \times support(I_i, I_j)$, the calculation needs $O(1)$ time as we have variables to store p_i, p_j, c_j and $support(I_i, I_j)(i.e.n_{ij})$. As there are n items, the RT is $O(n^2)$.

   (b) **Selection of $J-1$ items with the Highest Estimated Value for set S_i**

   In this sub-step, there are two procedures for each item. For each item, there are n estimated values. For each item, we will do the following:

   i. sorting the first J estimated values

   ii. comparing the remaining $n-J$ estimated values with the J sorted values

   i. For each item I_i, initially, J estimated values from item I_i are sorted. These sorted J values are put in a sorted list L. This procedure takes $O(Jlog\ J)$ time on average by using quick sort. Let $e_{i,x}$ be the smallest estimated value in the sorted list L.

The RT of quick sort on an array containing n elements is dependent a lot on where the split it is. For the sake of the simplicity of analysis, assume the split is at the midpoint along the array. Let $T(n)$ be the time of quick sort on an array containing n elements. It is easy to see that we can get a recursive formula $T(n) = n + 2T(n/2)$. By solving this formula, we can get $T(n) = nlog\ n + n$ if $T(1) = 1$.

In this step, if we need to sort J items, there are $(nlog\ n + n)$ operations in quick sort.

ii. Then, $e_{i,y}$, where $y = J + 1$, is read and compared with $e_{i,x}$. If it is larger than $e_{i,x}$, then discard $e_{i,x}$ in the list L and insert $e_{i,y}$ into the sorted list L. During the insertion into the sorted list, we need to find a correct position to be inserted such that all elements in the list L are still sorted. This step requires $log\ J$ time on average. The process continues for all remaining $n - J - 1$ estimated values (i.e. $y = J+2, J+3, ..., n$). As there are $n - J$ remaining estimated values and each insertion takes $log\ J$ time, the second procedure takes $(n - J)log\ J$ operations, which takes $O((n - J)log\ J)$.

Thus, the two procedures needs $Jlog\ J + (n - J)log\ J = nlog\ J$ operations, which takes $O(nlog\ J)$ time.

As there are n items, the overall RT of this sub-step is $O(n^2 log\ J)$.

The overall RT in this step is $O(n^2 + n^2 log\ J) = O(n^2 log\ J + nJ)$.

2. **Enhancement Step:Item Pruning**

If we apply the enhancement step, a smaller number of remaining items n_R are processed in the remaining iteration steps, which makes MPIS_Alg more efficient. There are two sub-steps:

(a) Calculation of L_i and H_i

(b) Item pruning

Before the above two sub-steps are to be analyzed, the RT of the calculation of benefit is first discussed.

(a) **Calculation of Benefit**

The benefit of each item, b_x is defined as follows:

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

if the output selection set $S = S_x \cup \{I_x\}$

In this step, we need to calculate $\widehat{P}(A)$, where $A = S_x \cup \{I_x\}$.

In Section 2.6.3, we can see how we use two efficient data structures called an *FP-tree* and an *FP-MPIS-tree* in order to compute $\widehat{P}(A)$.

In item benefit calculation, as we discussed in Section 2.6.3, we need to

i. read transactions from the FP-tree, which takes $O(n_{fp}n_{TS})$ time.

ii. calculate some of the profit of the item selection in the FP-MPIS-tree, which takes $O(n_{fp}n_{node})$ time.

i. **Reading transactions from the FP-tree**

Now, we will analyze the running time of reading transactions in the FP-tree. For each node with non-zero node count in the FP-tree, we need to traverse from that node upwards in order to generate a transaction. For each such node, the RT of the upwards traversal is $O(n_{TS})$ as the depth of the tree is $O(n_{TS})$. The worst-case in the Step of reading transactions from the FP-tree is that all nodes has the non-zero node count when we traverse the node. That means we need to traverse all nodes to generate a transaction. So, as the number of nodes in the FP-tree is n_{fp}, the RT of reading all transactions in the FP-tree is $O(n_{fp}n_{TS})$.

ii. **Calculating Profit in the FP-MPIS-tree**

We are going to analyze the running time of calculating profit in the FP-MPIS-tree. For each node we visited which is associated with a selected item, we need to do an increment by $p_a \times (1 - \frac{g(I_a,d)}{n_a}) \times count_n$. As there are variables storing p_a, n_a and $count_n$, the RT of the access of these variables is $O(1)$. We are interested to analyze the RT of the computation of $g(I_a, d)$.

In the FP-MPIS-tree implementation, it is easy to verify that we need to horizontally access $O(n_{node})$ nodes for item I_a in the FP-MPIS-tree. (NOTE: As we described before, the number of nodes is usually smaller than n_{node} due to the compressed property of the FP-MPIS-tree.) Then, for each node for item I_a, we may need to vertically access other nodes upwards and downwards. This path length is $O(n_{TS})$. So, the function $g(I_a, d)$ can be computed in $O(n_{node}n_{TS})$ time.

Thus, the RT for the term $prof(I_a, t_i)(1 - \frac{g(I_a,d)}{n_a})$ is $O(n_{node}n_{TS})$. However, in the FP-MPIS-tree, for the worst-case, we need to traverse all nodes. As there are $O(n_{fp})$ nodes in the FP-MPIS-tree, the RT of the calculation of some profit in the FP-MPIS-tree is $O(n_{fp}n_{node}n_{TS})$. As the maximum transaction size is small and constant, the RT of the calculation of profit in this tree is $O(n_{fp}n_{node})$.

Finally, we need to compute $\widehat{P}(A)$. In this step, we need to read transactions from the FP-tree, which takes $O(n_{fp}n_{TS})$ time. For each transaction, we need to check which group the transaction belongs to. The RT of this sub-step is $O(n_{TS})$. For the transactions containing selected items, we need to update the benefit, which takes $O(1)$ time. For the transactions containing unselected items, we do not need to update the benefit, which also takes $O(1)$ time. For the transactions containing both selected items and unselected items, we need to insert the transactions into the FP-MPIS-tree. Each insertion needs $O(n_{TS})$ time. Thus, for reading a transaction in the FP-tree, we requires $O(n_{TS}+1+1+n_{TS}) = O(n_{TS})$ time. As there are $O(n_{fp}n_{TS})$ transactions read from the FP-tree, the RT of the computation of $\widehat{P}(A)$ (without the computation of the profit in the FP-MPIS-tree) is $O(n_{fp}n_{TS}^2)$.

Now, by including the RT of the calculation of profit in the FP-MPIS-tree, the RT of calculation of $\widehat{P}(A)$ is $(n_{fp}n_{TS}^2+n_{fp}n_{node}n_{TS})$. Again, as the maximum transaction size is small and can be considered a constant, the RT of the calculation of benefit is $O(n_{fp}n_{node})$.

(b) **Calculation of L_i and H_i**

For each item I_i, L_i and H_i are calculated with the same formula of the benefit but with different output selection sets. The RT for the calculation of a pair of L_i and R_i is $O(n_{fp}n_{node} + n_{fp}n_{node}) = O(n_{fp}n_{node})$. The overall RT is $O(nn_{fp}n_{node})$ as there are n items.

(c) **Item Pruning**

After L_i and H_i are calculated, we need to find the J-th largest value of L_j for pruning. The implementation is just similar to the step Sorting Estimated Value. As there are n items and we are interested in the item, I_j with J-th highest value of L_j for pruning, the RT is $O(nlog\ J)$. Then, each H_i is scanned to check whether $H_i < L_j$ for pruning the item I_i. There are n items and the RT is (n). So, the RT for this pruning is $O(nlog\ J + n) = O(nlog\ J)$.

The overall RT for this step is $O(nn_{fp}n_{node} + nlog\ J)$.

3. **Item Benefit Calculation**

In this step, the benefits of the remaining items have to be updated. Let the number of remaining items be n_R. As discussed in the previous step, the RT of the item benefit calculation is $O(n_{fp} n_{node} n_R)$. As $n_R = O(n)$, the RT is $O(n_{fp} n_{node} n)$.

4. **Item Selection and Item Benefit Update & Iteration**

There are a number of iterations for item selection and item benefit update. For each iteration, the item with the smallest benefit is removed and the benefits of other related items are updated.

For the first iteration, finding the item I_x with the smallest benefit among n_R remaining items requires the scanning of n_R numbers. After the selection of the item I_x with the smallest benefit, in the worst-case where the sets S_i of $n_R - 1$ remaining items all contain item I_x, $n_R - 1$ benefits have to be updated as the sets S_i are updated.

For the second iteration, similarly, finding the item with the smallest benefit requires the scanning of $n_R - 1$ numbers. For the worst-case, after $n_R - 2$ benefits have to be updated.

For the k-th iteration, scanning $n_R - (k-1)$ numbers is required. $n_R - k$ number of benefits have to be updated for the worst-case.

For the last iteration, $J+1$ items are left and $J+1$ scannings are required. Let k-th iteration be the last iteration by equating the remaining items: $J + 1 = n_R - (k-1)$. That means $k = n_R - J$, which is the number of iterations.

Thus,

$$\begin{aligned} & \text{total number of scannings} \\ = \; & n_R + (n_R - 1) + ... + (J+1) \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2}(n_R + J + 1)(n_R - J) \\
&= \frac{1}{2}((n_R + J)(n_R - J) + n_R - J) \\
&= \frac{1}{2}(n_R^2 - J^2 + n_R - J) \\
&= O(n_R^2 - J^2)
\end{aligned}$$

Similarly, the total number of items to be updated with benefit is $O(n_R^2 - J^2)$. It is noted that each update of benefit takes $O(n_{fp}n_{node})$ time. Thus, the worst-case RT for the update of all benefits is $O(n_{fp}n_{node}n_R^2 - n_{fp}n_{node}J^2)$. Thus, the overall RT in this iterative step is $O(n_R^2 - J^2 + n_{fp}n_{node}n_R^2 - n_{fp}n_{node}J^2) = O(n_{fp}n_{node}n_R^2 - n_{fp}n_{node}J^2)$. As $n_R = O(n)$, the RT is $O(n_{fp}n_{node}n^2 - n_{fp}n_{node}J^2)$.

In summary, the RT of each step is shown as follows:

1. Estimation Set Creation (which takes $O(n^2 log\ J + nJ)$ time)
2. Item Pruning (which takes $O(nn_{fp}n_{node} + nlog\ J)$ time)
3. Item Benefit Calculation (which takes $O(n_{fp}n_{node}n)$ time)
4. Item Selection and Item Benefit Update & Iteration (which takes $O(n_{fp}n_{node}n^2 - n_{fp}n_{node}J^2)$ time)

The overall RT of Main Phase is $O((n^2log\ J + nJ) + (nn_{fp}n_{node} + nlog\ J) + n_{fp}n_{node}n + (n_{fp}n_{node}n^2 - n_{fp}n_{node}J^2)) = O(n^2log\ J + nJ + n^2n_{fp}n_{node} - n_{fp}n_{node}J^2)$. As $J = O(n)$, the RT is $O(n^2log\ n + n^2n_{fp}n_{node})$ (NOTE: The coefficient/constant term of the term $n_{fp}n_{node}J^2$ (about 0.5) is smaller than that of the term $n^2n_{fp}n_{node}$ (about 1).)

We summarize the RTs of the two phases as follows:

1. Preparation Phase (which takes $O(nlog\ n + mn_{TS} + n^2n_{node}n_{TS})$ time)

2. Main Phase (which takes $O(n^2 log\ J + nJ + n^2 n_{fp} n_{node} - n_{fp} n_{node} J^2)$ time)

The overall RT of MPIS_Alg is $O((nlog\ n + mn_{TS} + n^2 n_{node} n_{TS}) + (n^2 log\ J + nJ + n^2 n_{fp} n_{node} - n_{fp} n_{node} J^2)) = O(nlog\ n + mn_{TS} + n^2 n_{node} n_{TS} + n^2 log\ J + nJ + n^2 n_{fp} n_{node} - n_{fp} n_{node} J^2)$. As n_{TS} is constant and small, the RT is $O(nlog\ n + m + n^2 log\ J + nJ + n^2 n_{fp} n_{node} - n_{fp} n_{node} J^2)$. As $J = O(n)$, the RT is $O(m + n^2 log\ n + n^2 n_{fp} n_{node})$. (NOTE: The coefficient/constant term of the term $n_{fp} n_{node} J^2$ (about 0.5) is smaller than that of the term $n^2 n_{fp} n_{node}$ (about 1).)

2.9 Experimental Result

2.9.1 Tools for Quadratic Programming

We have searched on the web to find suitable tools to solve the quadratic programming. We have examined LINDO [4], TOMLAB[8], GAMS[3], BARON[1], OPTRIS[7], WSAT[9], Frontline System Solver[2], MOSEK[5] and OPBDP[6]. We choose Frontline System Solver (Premium Solver - Premium Solver Platform) as a tool of solving quadratic programming because it perform the best out of these solvers.

We have sent an email to each Solver Company to ask whether the product can solve the non-convex binary quadratic programming. The companies for LINDO, WSAT and MOSEK replied that the product cannot support such a problem. TOMLAB replied that its product can handle such kind of problem but this product has not been released yet. Frontline System Solver and BARON (based on the platform GAMS) replied that their product can solve such kind of problem. But, others have not replied yet.

So, in order to make our decision between Frontline System Solver and BARON, we have done some experiments to compare the output of the programming from two solvers. We used the maximization quadratic problem to find the optimal solution where J = n/2.

n	Optimal Value Found		Execution Time/min	
	Frontline	BARON	Frontline	BARON
50	8691.25915	7439.46665	4	16
100	57801.93125	50638.133	10	16
200	421746.8683	364420	42	12

From the experiment we conducted, we observed that Frontline Solver gives a greater value of the maximization problem than BARON. Besides, Frontline Solver gives a faster execution time to solve the problem for most cases. So, we choose Frontline System Solver as a tool to solve our quadratic program.

MatLab 6.1 can support quadratic programming with continuous variables. Besides, [15] developed a matlab function for solving mixed integer quadratic programs with convex property. But, our quadratic problem is a concave type. After conducting the above experiments, we decided to use Frontline Solver.

2.9.2 Partition Matrix Technique

We use the techniques of *partitioned matrices* when using the software Frontline Systems Solver (Premimum Solve Platform)[2] which is built on the Microsoft Excel. As excel has a limitation of at most 256 columns, we need to partition the matrixes in order to calculate the desired quadratic objective function.

For instance, if the number of items is 1000, the matrix Q used in our quadratic programming method is of the order 1000 by 1000. We partition the matrix into 16 submatrices. That is,

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} \end{pmatrix}$$

where Q_{ij} is a 250 by 250 matrix for $i, j = 1, 2, 3$ and 4. Then, we organize this matrix in sequence. That is, we define a matrix in the Microsoft Excel as

$$A = \begin{pmatrix} Q_{11} \\ Q_{12} \\ Q_{13} \\ Q_{14} \\ Q_{21} \\ Q_{22} \\ Q_{23} \\ Q_{24} \\ Q_{31} \\ Q_{32} \\ Q_{33} \\ Q_{34} \\ Q_{41} \\ Q_{42} \\ Q_{43} \\ Q_{44} \end{pmatrix}.$$

Thus, we can put the above matrix A in only a sheet in Microsoft Excel because there is no limitation of the number of rows in Microsoft Excel. Then, the multiplication between partitioned matrices/vectors is based on Theorem 3.

The following describes the partitioned matrices. The reader can find more details in [35].

Definition 10 Partitioned Matrices: *Any matrix can be decomposed into a number of submatrices. For example,*

$$\begin{aligned}
A &= \begin{pmatrix} 1 & -2 & 4 & 1 & 3 \\ 2 & 1 & 1 & 1 & 1 \\ 3 & 3 & 2 & -1 & 2 \\ 4 & 6 & 2 & 2 & 4 \end{pmatrix} \\
&= \left(\begin{array}{ccc|cc} 1 & -2 & 4 & 1 & 3 \\ 2 & 1 & 1 & 1 & 1 \\ \hline 3 & 3 & 2 & -1 & 2 \\ 4 & 6 & 2 & 2 & 4 \end{array}\right) \\
&= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}
\end{aligned}$$

where

$$\begin{aligned}
A_{11} &= \begin{pmatrix} 1 & -2 & 4 \\ 2 & 1 & 1 \end{pmatrix} \\
A_{12} &= \begin{pmatrix} 1 & 3 \\ 1 & 1 \end{pmatrix} \\
A_{21} &= \begin{pmatrix} 3 & 3 & 2 \\ 4 & 6 & 2 \end{pmatrix} \\
A_{22} &= \begin{pmatrix} -1 & 2 \\ 2 & 4 \end{pmatrix}
\end{aligned}$$

As we know that the matrix can be partitioned, the matrix multiplication is given in the following theorem[35].

Theorem 3 (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* $AB = \begin{pmatrix} A_{11}B_1 + A_{12}B_2 \\ A_{21}B_2 + A_{21}B_2 \end{pmatrix}$

2.9.3 Data Sets

In this thesis, several synthetic data sets and a real data set are to be tested in our experiments.

Synthetic Data Set

In our experiment, we use IBM synthetic data generator [10] to generate the data set with the following parameters (same as the parameters of [49]): 1,000 items, 10,000 transactions, 10 items per transaction on average, and 4 items per frequent itemset on average. We generated the profits of items with a lognormal distribution. The price distribution can be approximated by a lognormal distribution, as pointed out in [28]. That is the logarithm of the profit follows the same distribution as the data set in [49]. We use the same settings as [49]. 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.

We also generate two data sets to illustrate some weaknesses of HAP, as described in Section 2.2. They are called *HAP worst-case data set 1* and *HAP worst-case data set 2*. For HAP worst-case data set 1, we generate a loop as in Figure 2.1. If there are 10,000 transactions and 1000 items, we generate the first 20 transactions containing only two items, says I_1 and I_2. A loop containing item I_1 and I_2 can be generated. Then, only one item from the remaining items (i.e. $I_2, I_3, ..., I_{1000}$) is generated in each of the remaining transactions ($t_{21}, t_{22}..., t_{10000}$). This means that each remaining transaction contains only one item other than the items involved in the loop. The profit distribution is generated similarly as IBM synthetic data set.

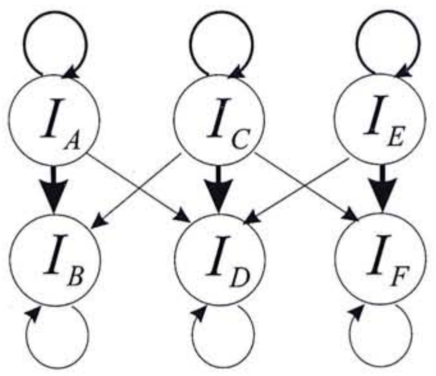

Figure 2.8: Illustration: Hap Worst-Case dataset 1

We now illustrate how to generate HAP worst-case data set 2. The idea is illustrated in Figure 2.8. 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 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 $\{I_A, I_B\}$, 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.

In Figure 2.8, 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 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. The reason why we assign the total profit to the item at the upper layer is that we observe that, in each association rule $I_A \to I_B$, the authority of an item I_B only depends only two factors, the total profit of item I_A and the confidence factor of the association rule $I_A \to I_B$, but is independent of the total profit of the item itself (i.e. I_B), which is one of the disadvantages of HAP. In this data set, the items at the upper layer are assigned with the high profit range between \$5 and \$10 while the items at the lower layer are assigned with the low profit range between \$0.1 and \$1. The worst-case data sets are used to illustrate the weaknesses of HAP, as described in Section 2.2.

Real Data Set

The real data set is obtained from a large drug store in Canada over a period of 3 month [49]. 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 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 of Items
\$0-\$0.1	2.03%
\$0.1-\$1	25.05%
\$1-\$5	54.59%
\$5-\$10	10.43%
\$10-\$100	7.75%
\$100-\$400	0.15%

2.9.4 Empirical Study for GA

In this section, we will describe what experiments related to GA have been conducted. All experiments were done with the IBM synthetic data set described in Section 2.9.3. The following experiments have been conducted:

1. **GA Parameter Experiment** (Section 2.9.4)

   The experiment has been conducted to study two important parameters used in GA - Population Size and No. of Generations. In this experiment, we also study several additional factors to be described in this section - *mutation effect*, *child effect* and *strategy effect.*

2. **MPIS Experiment** (Section 2.9.4)

   The experiment has been conducted to study MPIS problem with the variation of the number of items selected.

GA Parameter Experiment

In this subsection, we are going to study the effect of the GA parameter on the profitability and execution time of our GAs. We choose $J = 500$ (i.e. no. of items selected) for these GA parameter studies [6].

We carried out GA experiment 20 times [7] , each with different random starts. We will call the number of times the experiments are carried out as the number of runs (which is a common term used in GA) later. Based on 20 experiments, we got

[6]We choose $J = 500$ because this is the half item selection, which corresponds to the largest solution space C^n_J, where $n = 1000$ and $J = 500$. We would like to find the effectiveness of our GA in a large solution space. If GA can yield a good result in a large solution space, it is expected that GA can give a good result in a smaller solution space. Besides, in MPIS experiment (which is a study of the effect of the variation of selection on the profitability), we found that the difference in profitability between our all proposed algorithms is the greatest when the selection is half.

[7]Usually, in the literature of genetic algorithm, the number of times (or runs) is set to be 10 or above. One of the examples can be found in [41].

two major measurements - average profitability, and the average execution time.

We have conducted the experiments with the variation of the following factors: variation of population Size and variation of number of Generations Different experiments are conducted with mutation and without mutation. Besides, we will study the effect of the number of children generated during crossover.

In our experiments, we describe our GAs in the following table:

Notation	**Description**
BIN-C1 with mutation	each crossover generates only one child, with mutation
BIN-C1 without mutation	each crossover generates only one child without mutation
BIN-C2 with mutation	each crossover generates two children, with mutation
BIN-C2 without mutation	each crossover generates two children, without mutation

BIN stands for GA in binary representation.

For BIN, in addition to the factor whether the mutation takes place, we are going to investigate the number of children generated in each crossover. In Section 2.7.1, the crossover operator in GAs only generates one child. BIN-C1 is the GAs generates only one child. BIN-C2 is the GAs generates two children. In BIN-C2, for each generation of children, we apply the crossover operator in Subsection 2.7.1 twice, in order to generate two children.

In this section, we are going to study the profitability and execution time of GAs described in the table on two factors - (1) the variation of population size and (2) the number of generations. Besides, for each factor, we are going to investigate the following:

- **Mutation Effect**: the effect of mutation for each kind of GAs
- **No. of Children Effect**: the effect of number of children generated during crossover

1. **Variation of Population Size**

In this part, we are going to investigate the effect of profitability and execution time on the variation of population size. In this experiment, the parameter is set as follows. We conducted our experiment 20 times. The number of runs is 20. The number of generations is 20. In BIN , we set $p_{mutate} = 0.5$ (i.e. the probability of mutation). [8]

(a) **Mutation Effect**

From the graph in Figure 2.9, GAs with mutation usually gives a greater average profitability than GAs without mutation. This shows the ability to increase the diversity in the population by mutation, which can give a better result.

In Figure 2.10, the execution time of GAs with mutation and without mutation are nearly the same as the mutation operations does not require much computation time.

(b) **No. of Children Effect**

Profitability: In BIN, we found that, during a crossover, generating two children is better than generating only one child as the average profitability of BIN-C2 is greater than that of BIN-C1. This is because BIN-C2 (i.e. generation of two children) can give a higher probability of the generation of the "better" child compared with BIN-C1 (i.e. generation of one child). Although the average profitabilities of BIN-C3 and BIN-C4 is greater than that of BIN-C2, the difference between them is small.

Execution Time: In Figure 2.10, the execution time of BIN-C2 is

[8]We have conducted the experiment on the variation of p_{mutate} from 0 to 1. We found that the setting $p_{mutate} = 0.5$ gives the greatest average profitabilities. The reasons is described as follows. If $p_{mutate} = 0$ (i.e. no mutation), the gene cannot be jumped out of a local optimal. If $p_{mutate} = 1$ (i.e. the gene always mutates), the goodness of the gene generated from the parent may be deteriorated.

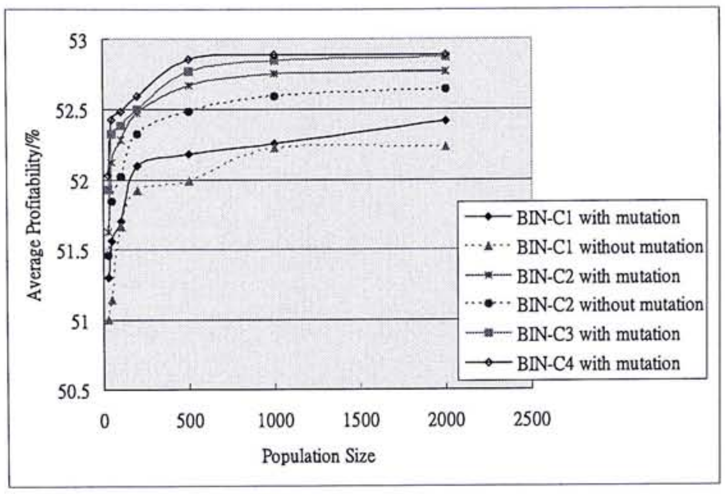

Figure 2.9: Graph of Average Profitability of GAs against Population Size, where $J = 500$ and no. of generations=20

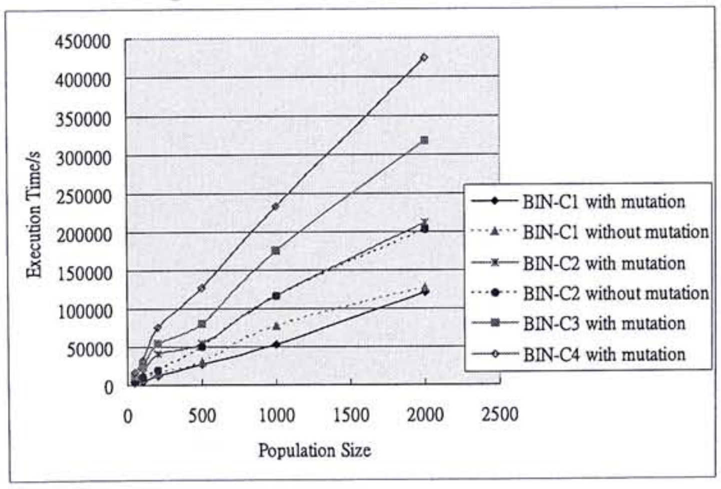

Figure 2.10: Graph of Average Execution Time of GAs against Population Size, where $J = 500$ and no. of generations=20

approximately twice that of BIN-C1. It is obvious that we need to evaluate one more child by fitness function for each crossover in BIN-C2. Similar arguments can be made in BIN-C3 and BIN-C4.

(c) **Population Size Effect**

In Figure 2.9, we observe that nearly all average profitabilities of all GAs (BIN-C1, BIN-C2, BIN-C3 and BIN-C4) increase with population size from 50 to 1000. Then, the profitabilities level off after population size of 1000.

From Figure 2.10, the execution time of all GAs increase with population size. It is because we need to calculate more candidates in the population when the population size increase.

2. **Variation of Number of Generations**

In this part, how the factor of number of generations affects the profitability of execution time is studied. In this experiment, the parameter is set as follows. The number of runs is 20. The population size is 100. In BIN, we set $p_{mutate} = 0.5$ (i.e. the probability of mutation).

We are going to analyze the following effects:

(a) **Mutation Effect, Number of Children Effect and Strategy Effect**

The effect of mutation, no. of children and strategy in this experiment with the variation of the number of generations is just similar to that in the experiment with the variation of population size, described in the above part. That is, it is preferable to use mutation for diversity. During crossover, BIN-C2 (i.e. generation of two children during each crossover) gives a greater profitability than BIN-C1 (i.e. generation of

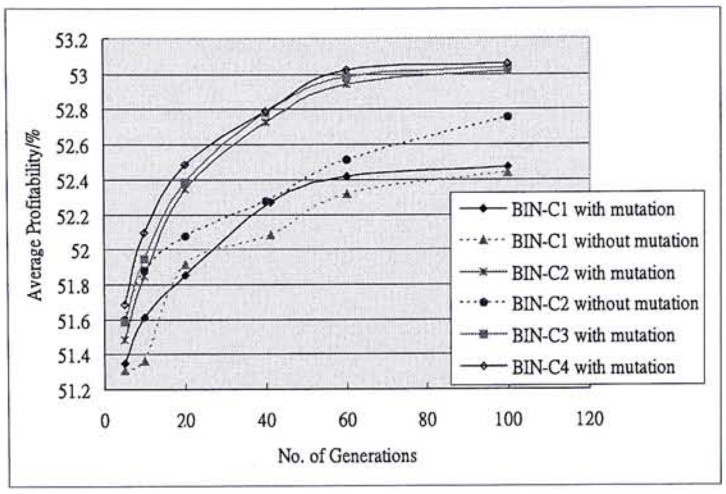

Figure 2.11: Graph of Average Profitability of GAs against No. of Generations, where $J = 500$ and population size=100

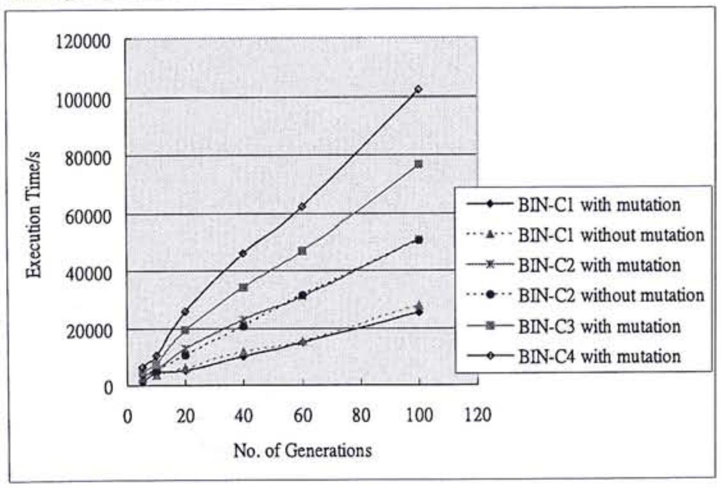

Figure 2.12: Graph of Execution Time of GAs against No. of Generations, where $J = 500$ and population size=100

one child during each crossover). In this experiment, we found that the difference between BIN-C3/BIN-C4 and BIN-C2 is quite small. We will adopt the crossover which generates two children in MPIS experiment, because not only the execution time of BIN-C2 is twice faster than that of BIN-C4 but also the difference in profitabilities between these GAs is small.

(b) **Effect of the Number of Generations**

Profitability: In Figure 2.11, the profitabilities of nearly all GAs (BIN-C1, BIN-C1, BIN-C3 and BIN-C4) increase with no. of generations from 5 to 60. The lines for the profitabilities of those GAs after 60 generations are still increasing but the increase is not quite sharp. So, as we observed, in the later experiment - MPIS experiment, we choose 60 as the number of generations for one of the experiment sets. **Execution Time:** The execution time of all GAs increases with the number of generations (as shown in Figure 2.12). It is trivial as GAs require more computations of the fitness function of each children when the number of generations increases.

MPIS Experiment

In this subsection, we are going to study our problem MPIS. We will study how the factor of number of items selected, J, affects the profitabilities and execution time. As we observed from the above experiments, we would choose BIN-C2 with mutation in the following experiments as BIN-C2 gives good results in a reasonable time among all GAs. We have tried to conduct two kinds of experiments with two different settings, *Setting 1* and *Setting 2*.

1. **Setting 1:** The number of runs is 20. The number of generations is 60. The population size is 100.

The reason we choose the parameter setting for the number of generation is our results shown in the last section (Section 2.9.4) - GA Parameter Experiment. We choose the setting which GA gives the nearly the best profitability and not too large execution time. In GA Parameter Experiment (with the study of the factor of no. of generations) , we observed that, if we use this setting, 52% average profitability is obtained.

2. **Setting 2:** The number of runs is 20. The number of generations is 20. The population size is 25.
   Although this experiment set with this setting may not give a greater profitability than the experiment set with setting 1, this experiment set can give fast execution time and reasonable large profitability. In GA Parameter Experiment (with the study of the factor of population size) , we observed that this setting can yield about 51% profitability. Compared with Setting 1, we have similar profitability but the execution time is 12 times faster.

For both settings, the other parameters are set as follows. In BIN, we set $p_{mutate} = 0.5$ (i.e. the probability of mutation).

From this experiment, we conclude that we prefer choosing Setting 2 as the parameter setup. This is because the experiment set with Setting 2 can give a reasonably great profitability with fast execution time. For instance, if $J = 900$, then GA with Setting 1 requires about 1 day but GA with Setting 2 requires within 1 hour. On average, GA with Setting 2 runs 24 times faster than that with Setting 1. Besides, on average, the difference in profitabilities between GAs with Setting 1 and Setting 2 is at most 0.4%.

In the following experiments, we will choose GA in binary representation with Setting 2 to conduct our experiments.

2.9.5 Experimental Results

We have done the experiment of the data set described above. We use a Pentium IV 1.5 GHz PC to conduct our experiment. We use Frontline System Solver to solve QP method. 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.

Example 4 (Naive Example) *We use the same scenario in Table 2.1 and Table 2.3 to illustrate the naive approach. The individual profit of each item is shown as follows.*

i	*Individual Profit of Item I_i*
1	*40*
2	*35*
3	*40*
4	*5*
5	*38*
6	*1*

If $J = 3$, the items with the greatest individual profits are selected. That means items I_1, I_3 and I_5 are selected. The benefits b_1, b_3 and b_5 are \$0, \$0 and \$19 respectively. The total profit of item selection is \$19.

□

Synthetic Data

In our experiments with synthetic data, the min-support used is 0.1%. The min-support is used in the step individual count derivation in algorithm MPIS_Alg.

First of all, we have done an experiment on the IBM synthetic data with the variation of number of items selected when the profit follows log-normal distribution [28] and the total number of items is equal to 1000. The profitability and execution time of algorithm QP, MPIS_Alg, HAP and naive are shown in Figure 2.13 and Figure 2.14. The graph of number of remaining items after pruning Step is shown in Figure 2.17. Without the pruning step in Algorithm MPIS_Alg, the profitability and execution time are shown in Figure 2.15 and Figure 2.16.

Secondly, we conducted an experiment for the HAP worst-case data set 1, where $n = 1000$. The profitability and the execution time against the number of items selected are shown in Figure 2.18 and Figure 2.19. For the HAP worst-case data set 2, similar graphs are shown in Figure 2.20 and Figure 2.21. The number of remaining items against J after the pruning step in MPIS_Alg can be found in Figure 2.22

We are going to analyze and describe the execution time and profitability of the algorithms.

1. **Execution Time**

   In the experiment of all data sets, QP approach gives fluctuation of execution time with the number of items. The execution time MPIS_Alg increases to a maximum value when the percentage of the number of items selected is approximately equal to 50% and decreases after that. Besides, QP approach gives the greatest execution time. Naive approach remains constant at the fastest execution time. HAP approach also remains unchanged at the second fastest execution time.

   (a) **MPIS_Alg**

   From the graph of the execution time against the selection, the execution time of MPIS_Alg increases from 0% selection to half of the selection

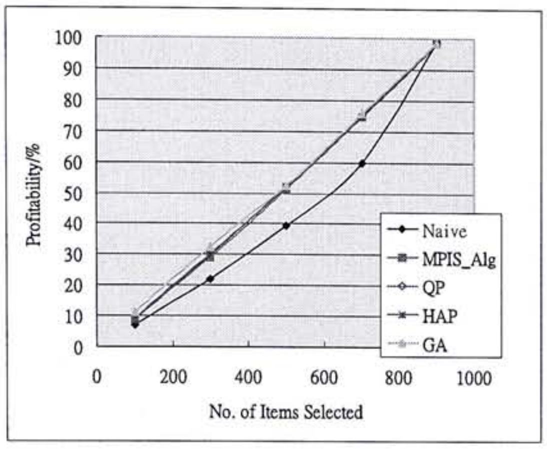

Figure 2.13: Graph of Profitability against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (With Pruning Step)

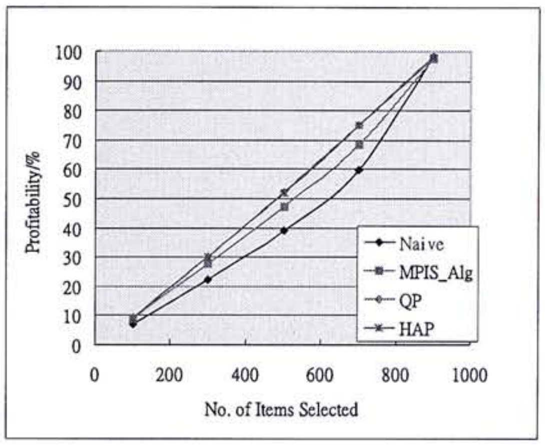

Figure 2.15: Graph of Profitability against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (Without Pruning Step)

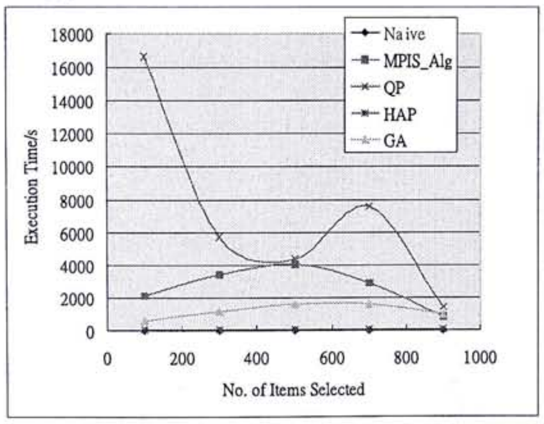

Figure 2.14: Graph of Execution Time against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (With Pruning Step)

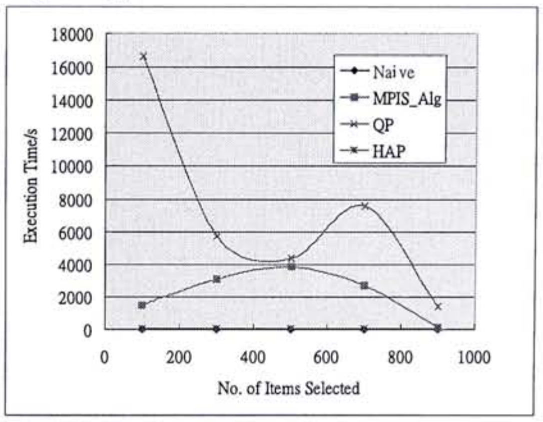

Figure 2.16: Graph of Execution Time against No. of Items Selected where $n = 1000$ and the profit follows log-normal distribution (Without Pruning Step)

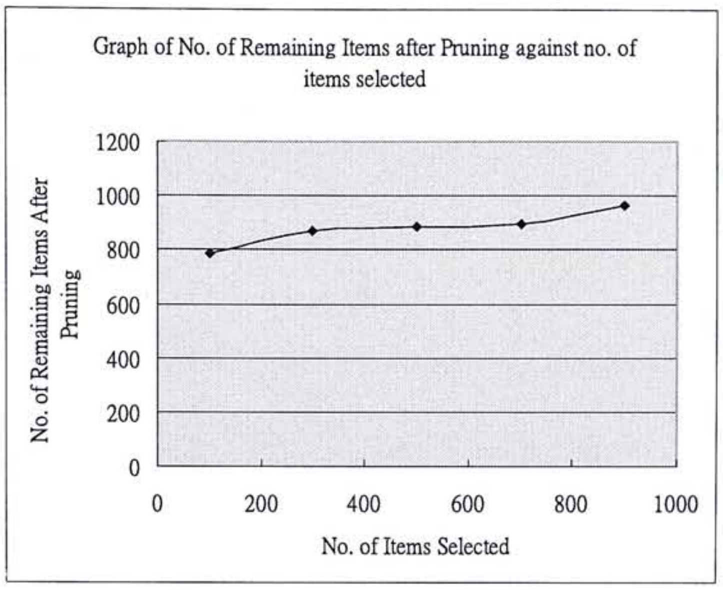

Figure 2.17: Graph of No. of Remaining Items after Pruning Step where $n = 1000$ and the profit follows log-normal distribution

and then decreases afterwards. Actually, 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 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 slection is larger than 50%.

(b) **Quadratic Programming**

The quadratic programming approach(QP) used in the Solver uses a variant of the Simplex method to determine feasible region and then

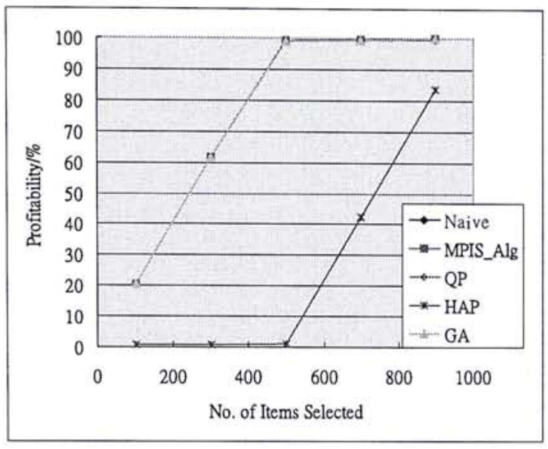

Figure 2.18: Graph of Profitability against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 1

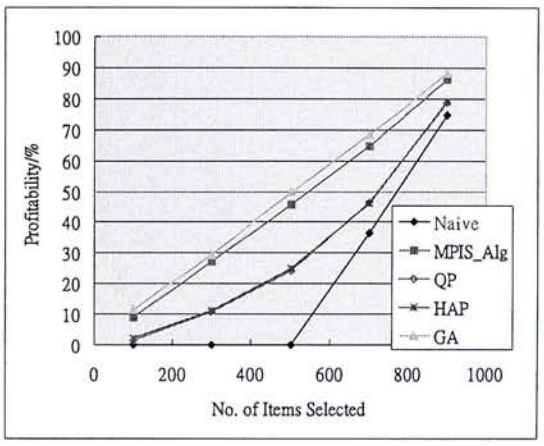

Figure 2.20: Graph of Profitability against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 2

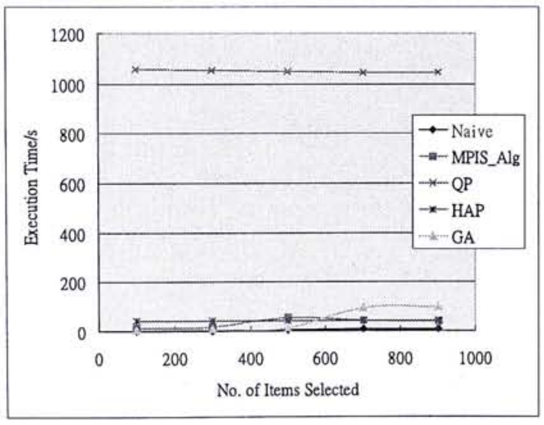

Figure 2.19: Graph of Execution Time against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 1

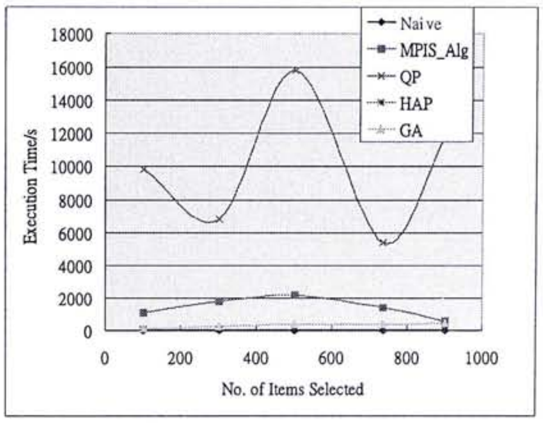

Figure 2.21: Graph of Execution Time against No. of Items Selected where $n = 1000$ for the HAP-worst case data set 2

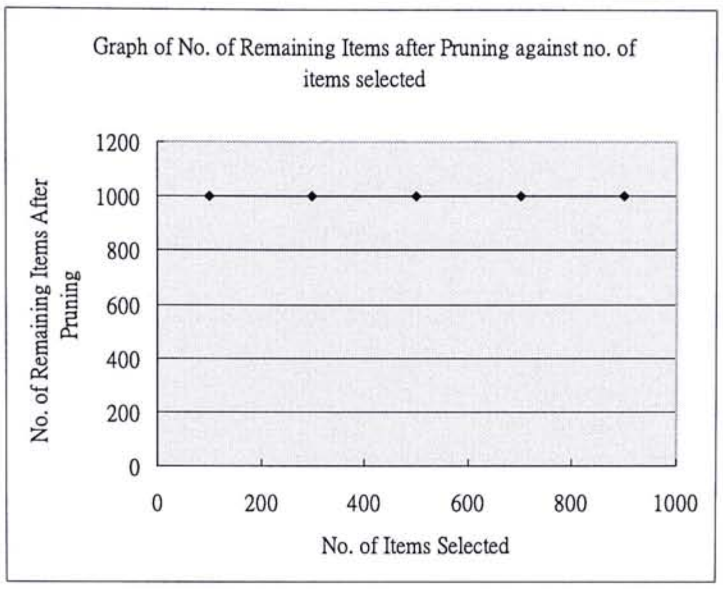

Figure 2.22: Graph of No. of Remaining Items after Pruning Step where $n = 1000$ for the data set which HAP performs worst

uses the methods described in [26] to find the solution based on the property of the quadratic property.

Simplex method is a systematic approach to find the optimal solution at the corner/extreme point of the solution set [20]. It first starts with a basic feasible solution. Then, it does an iterative step to improve the objective function by moving other basic feasible solutions. The approach used in the Solver only uses the step of finding the feasible region (or feasible sets) of Simplex method.

[26] is the algorithm called *A Finite Algorithm to Maximize Certain Pseudoconcave Functions on Polytopes.* This algorithm is to find the optimum of certain nonlinear functions (quadratic functions in our case). It selectively decomposes the feasible sets into simplices of varying dimensions. Then, by using linear programming technique and a gradient criterion, a sequence of these simplices are selected based on calculus.

From this sequence, the interior maxima can be found.

Thus, as the approach used in the Solver 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).

(c) **GA**

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.

(d) **HAP**

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 converges rapidly. Besides, [31] also stated that the convergence is quite rapid (usually about 20 iterations).

(e) **Naive**

The naive approach performs the fastest. It simply calculates the profits generated of each item for all transactions and select the J items with the greatest profits.

It is observed the execution time of this approach remains unchanged because the change is insignificant as the execution time is quite small.

(f) **Comparison**

Moreover, QP performs the greatest execution time compared with other algorithms. This is because this approach does not use any heuristics for this problem but other algorithms use some heuristics of the problem that will cause the execution time shorter. Naive gives the shortest execution time as there are only simple fast operations. HAP gives the second shortest execution time for this small synthetic data set. (But, it gives the greatest execution time for a large data set (shown in the real data set).) As mentioned before, the number of iterations involved are quite small. MPIS_Alg is ranked with the second greatest execution time, which uses the heuristics approach and greedy approach to solve this problem. GA is faster than MPIS_Alg as, with Setting 2, it does not involve a great number of chromosomes required for computation. Thus, it runs faster than MPIS_Alg.

2. **Profitability**

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 which is involved in the calculation of the profitability.

In the experiment with the variation of the number of items selected, it is quite trivial that 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 the greatest. But, HAP performs the worst compared with all algorithms. However, for the HAP worst-case data set 1, MPIS_Alg gives the greatest profitability among all algorithms. In addition to naive approach, HAP approach gives the second worst profitability.

Real Data Set

In this drug store data set, we have conducted a similar experiment with synthetic data set. [9]

With the consideration of all items, the profitability and the execution time with the variation of number of items selected are shown in Figure 2.23, Figure 2.24 and Figure 2.25.

We can see that HAP really gives the lowest profitability. However, MPIS_Alg and GA give a greater profitability than naive approach because naive approach does not consider any cross-selling effect.

In the drug store data set, HAP gives the worst profitability. The reason is described as follows.

In the dataset, there are some items with zero-profit and high authority weight (described in Section 2.2), 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. The above situations occur quite naturally when the frequency of 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 and GA give a greater profitability than naive approach in 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.

[9] Quadratic programming usually does not work well for the problems of reasonable size (which especially occur in the real life data). For instance, quadratic programming cannot solve the problem with reasonable time. The current Quadratic Programming (QP) Solver [2] can only solve not more than 2000 variables. In the real data set, there are 26,128 variables (i.e. items). This solver cannot solve such large problem.

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 choice for an application.

The profitabilities/execution time of MPIS_Alg and GA are quite similar.

It is noted that, if the total number of items is 1000, the execution time of HAP is smaller than that of MPIS_Alg. However, when the problem size is large and the total number of items is 26,128, the execution time of HAP is much slower than MPIS_Alg. This means that the execution time of HAP increases significantly with the problem size increases compared with MPIS_Alg. In HAP, there is a *cross-selling matrix* B, to be updated. The matrix is of the order $n \times n$. 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 $M = B^T B$. We can see that there is matrix multiplication of matrix M with vector a. It is noted how much memory used for the matrix M used. If double data type (8 bytes) is used for storage of each entry, then the matrix requires the memory size of about 5.08GB. If float data type (4 bytes) is used, about 2.5GB memory size is used. This large matrix cannot fit into the physical memory, causing more disk access for virtual memory. As this matrix is sparse, a hash data structure can also be used. Only non-zero entries are stored in the hash data structure. In the real data set, less than 5MB memory size are needed if a hash structure is used.

On the other hand, algorithm MPIS_Alg contain a pruning step which remove the items that has little chance in the final selection. After this pruning step, only a few remaining items are there. So, the execution time of MPIS_Alg is much fewer. For instance, if the number of items to be selected J is 500, then there are only 538 remaining items after the step.

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 2.23. For instance, if J = 500 and min-support = 0.05%, the profitability of naive and MPIS_Alg is about 1.3%. 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.

2.9.6 Scalability

We have also studied the scalability of all algorithms (Naive, MPIS_Alg, QP, HAP and GA) we studied. We conducted two kinds of such experiment. 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 synthetic data set we used, we generated a number of data sets by using IBM synthetic data generator [10] 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 [28]. 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 of execution time against the number of transactions, the number of items is set to be 1,000 while the number of transactions

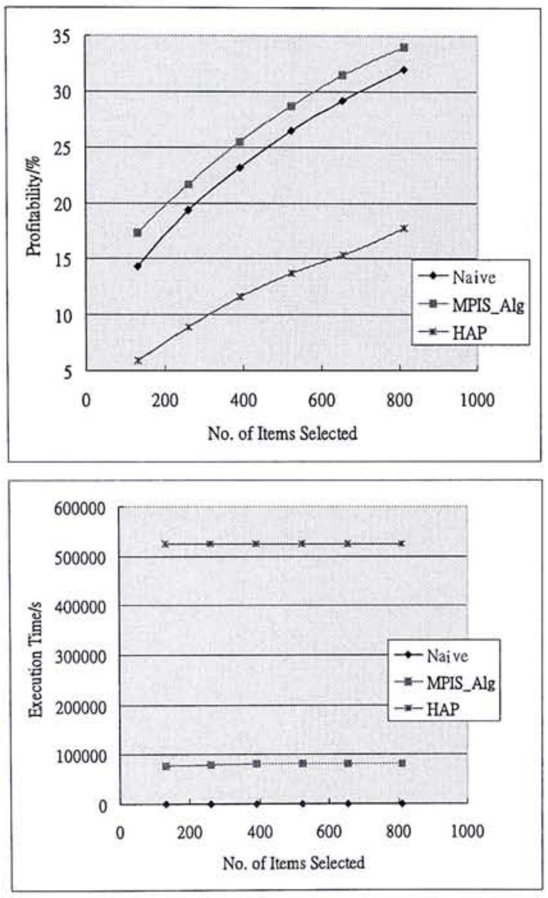

Figure 2.23: Graph of Execution Time against No. of Items Selected for the drug store data set when $J < 1000$

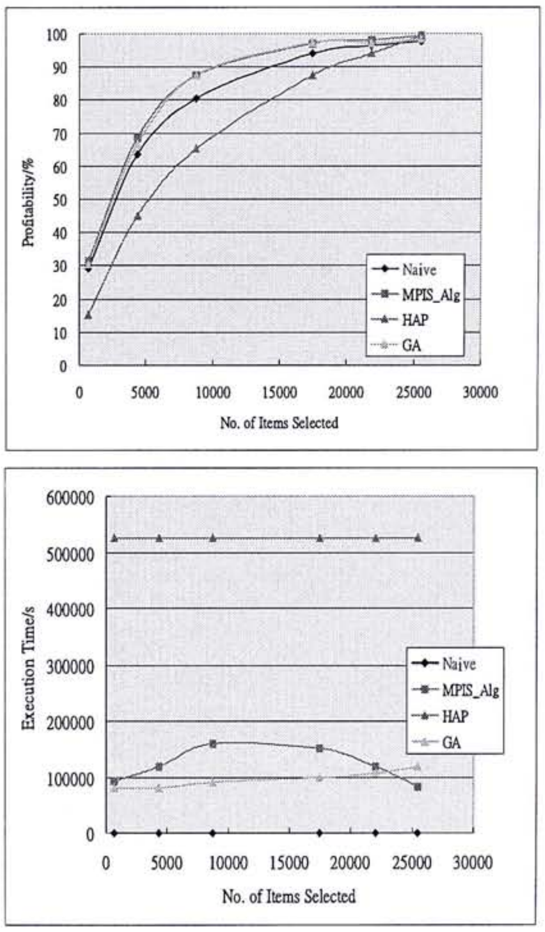

Figure 2.24: Profitability and Execution Time against No. of Items Selected The drug store data set

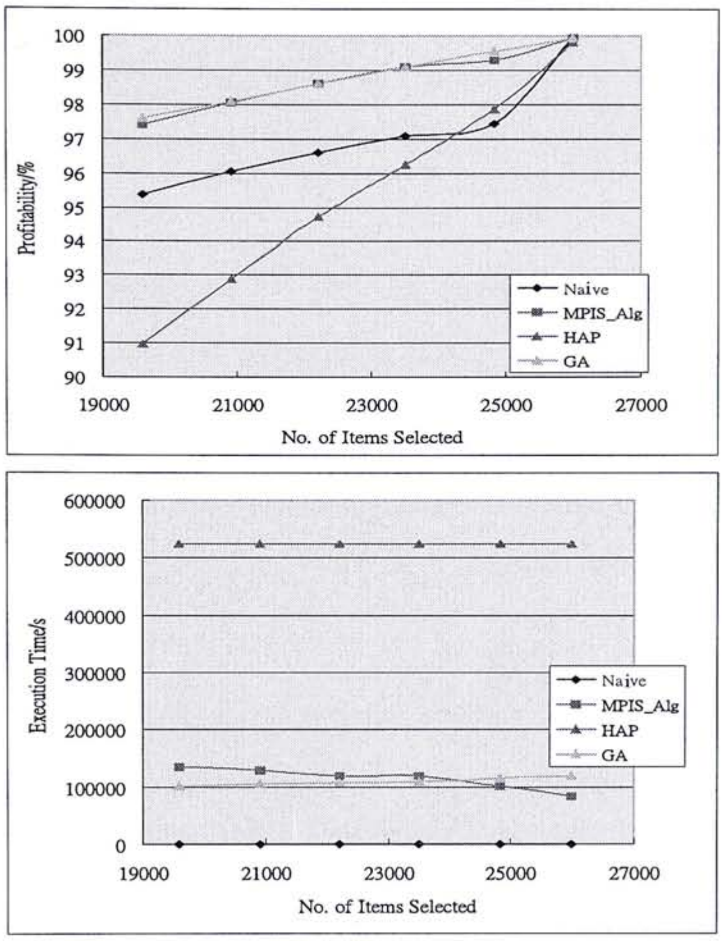

Figure 2.25: Graph of Profitability and Execution Time against No. of Items Selected The drug store data set when $J > 19000$

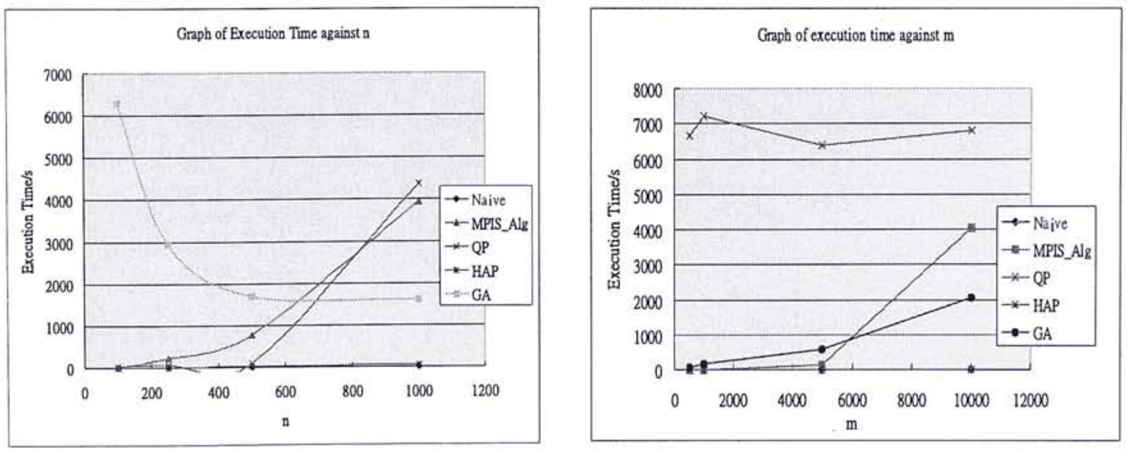

Figure 2.26: Graphs of Execution Time against total number of items n and against total number of transactions m, the profit follows log-normal distribution (NOTE:Execution Time lines for HAP and naive are overlapping)

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 2.26 where the execution time is the average of different random sets.

We observed that the execution time of most algorithms increases with the number of transactions and the number of items. It is quite trivial because most algorithms run longer with a larger data size. However, we will describe one special line for each graph as it does not follow a general trend of other algorithms. For the graph of execution time against the number of items n, we observed that the line of GA decrease exponentially with n. As we remember, we 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 with other items. In other words, the number of items which co-occurs with an item is greater. This will leads to a greater number of branches of each node in 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, it is required to traverse more branches for the computation of the objective function, which makes algorithm GA slower. Compared with other algorithms, GA requires to have a huge amount of computations of the objective functions. So, the execution time of GA decreases with n.

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 is based on the dimensionality of the quadratic objective function. No matter how the number of transactions changes, QP runs for nearly the same execution time.

2.10 Conclusion

One of the applications of the concepts of association rule - the maximal-profit item selection problem with cross-selling effect (MPIS) is discussed in this thesis. We propose a modelling 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 works 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. We also plan to analyze the performance of our algorithm in term of running time in order to compare with our empirical results.

We may study a variation of the problem with the space constraints. As some managers of a store may need to decide to select a subset of items in order to save spaces in the store, the space constraint can be added into our problem. In the new problem, the size of each item and the space of the store are some of the inputs. Then, the problem is to find a set of items and the number of storage units such that the profit of item selection is maximized and the space at the store is fully utilized.

Moreover, item selection can be involved with an additional constraint of categories. That means each item belongs to a specific category. For example, item *cheese* belongs to category *food* while item *pencil* belongs to category *stationery.* The problem can be formulated as an original item selection such that the number of selected items in each category should be greater than or equal to a user-defined number.

Chapter 3

ISM

3.1 Introduction

Recently, there have been some researchers [21, 42, 30] who proposed the application of selection problem for marketing. Their problem is to find a subset of customers which is valuable to be marketed in order that those customers to be marketed can persuade their friends to purchase the items in the store (Such phenomenon is called "word-of-mouth" effect). Such marketing can boost the sales of the store. However, those proposed approaches are dealing with the customer selection. Item selection for marketing is still an important problem in a typical store. Manager in a store may want to choose a subset of items in the store for marketing in order to boost the sales of other items which are not being marketed with the cross-selling effect that customers may purchase more non-marketing items if they purchase marketing items. However, the models of previous work [21, 42, 30] cannot deal with our item selection problem, which will be discussed in this thesis.

The problem Item Selection for Marketing (ISM) with cross-selling effect is addressed in this thesis. ISM is to find a subset of items as marketing items so that, after a marketing campaign on these selected marketing items is promoted, the whole sales of the store will be increased and the store will earn much money.

In this chapter, the problem of item marketing is studied. Besides, we prove that a simple version of this problem is NP-hard. We also proposed two algorithms to solve this problem. One is a hill-climbing approach while the other is a classical optimization approach. In our experiment, both approaches are quite effective and efficient.

The remaining in this chapter is organized as follows. Section 3.2 introduces the related work of our problem. Section 3.3 gives a formal definition of our problem. Section 3.4 presents the details of the cross selling effect by association rule - gain rule. Section 3.5 and Section 3.6 gives our proposed algorithms - QP and Hill Climbing. Section 3.7 presents our performance study. Section 3.8 summarizes our study.

3.2 Related Work

In our modeling, we adopt the association rules to model the marketing of items. Besides, we also utilize the network effect concerned in the papers addressed in the field of network model.

3.2.1 Network Model

In the traditional research of social networks, [13] proposed a diffusion model for the "word-of-mouth" effect. [37] summaries the trend of the social networks in the literature. Besides, there are some recent related works ([21], [42] and [30]) on the selection problem on viral marketing.

Social Network Model

In the literature of social networks, there are some related works (e.g. [13] and [37]). The researchers in the literature investigated the *diffusion processes* for "word-of-mouth" effect in the success of the new products.

Besides, in the social network model described in [50], the selection of a subset of nodes in the network is based on the *degree centrality* and *distance centrality.*

There are some weaknesses.

1. The approach based on degree centrality and distance centrality cannot handle well the case that most of the highly central nodes (i.e. highest-degree nodes) are in a cluster so that an selection of those nodes in the cluster is unnecessary.

Domingos's and Richardson's Model

[21] and [42] introduced to employ network effects (or network influence) in the application of marketing. The determination of marketing strategy with the use of network effects is called *viral marketing.*

Domingos's and Richardson proposes two models ([21] and [42]).

In the first model, [21] proposed a probabilistic model to select a subset of customers in order to maximize the total profit with the the considerations of the "word-of-mouth" effects. [21] claimed that this problem was intractable and proposed three approximate procedures to tackle this problem. The approximate procedures are single pass, greedy search and hill-climbing search. In [21]'s experiments, among three approximate procedures, hill-climbing search gives the greatest profit.

The second model proposed by [42] was just similar to [21]. However, [21] modeled this network effect with a non-linear function but [42] adopted a simple linear model to approximate this effect. As a linear approximation is used in [42], an optimal solution of customer selection could be obtained.

However, there is a weakness. The probability function that a customer purchases a product in terms of the marketing action in the network should be differentiable.

Kempe's, Kleinberg's and Tardos's Model

Kempe's, Kleinberg's and Tardos's Model [30] was an operational model from mathematical sociology and interaction particle systems. They described two models - *Linear Threshold Model* and *Independent Cascade Model.* They also developed a general framework based on *submodular functions* in order to employ the a performance guarantee of such functions proved by some previous work. Their proposed approach has a proved guarantee with some properties of the functions. In their experiment, the approach outperformed the traditional approaches in social network (e.g. degree centrality and distance centrality).

But, the model has some weaknesses.

1. As their model depends heavily on *submodular functions* and the property of the function. It reduces the flexibility of solving some kinds of problems. As stated in [30], it is too not trivial to provide a guarantee in a general threshold and cascade models as those models are so broad that some models cannot utilize the submodular functions.

2. Let $h_v(x)$ be the probability that the node v will become active if the marketing strategy x is used. In [30], the function $h_v(x)$ should be twice differentiable, non-decreasing and concave so that their proposed approach have the proved guarantee. However, [30] did not show that it was easy to define some functions with such properties. Besides, we also believe that it is a difficult task to define a continuous non-decreasing function with twice-differentiable and concave property. Thus, their proposed general marketing strategies cannot solve the daily life problem with proved guarantee easily.

Common Weakness

In such kinds of network model, there are the following common weaknesses.

1. There is no knowledge of the function used in their models. That means the function may be independent of the customer behavior and the function is constructed to estimate the customer behavior. That is, the models make up some functions which may not be a correct model for the real world.

2. The functions used in their models are so simple that it cannot be applicable in the real world. For instance, some of the models are modeled by adding the weightings of all neighbors of a node. Such models cannot estimate accurately the influence of the neighbors. The influence value should be equal to a value smaller than the sum of the weightings of all neighbors, because there exist some neighbors which contribute the same effect on a node, which makes the total net effect on the node with a smaller value.

3. Their models lack sufficient data to support. Their customer selection model require the information between customers. However, it is difficult to obtain such kinds of information in the data set. [42]uses the estimate of equal influence weighting of neighbors, which loses the real world's unequal influence weightings between people. Besides, some researchers [30] propose to utilize a collection of the research papers to obtain the information between customers. However, as pointed out in [30], there are still some researchers who have not published papers together but are still good friends (or neigbhors) with each other, which introduces a high weighting of network influence. Moreover, such data sets only obtain the information of customers in research domain. Real-world enterprizes need the customer behavior not only from researchers but also the normal residents in the world.

3.3 Problem Definition

Item Selection for Marketing (ISM) is a problem to select a set of items for marketing, called *marketing item*, so as to maximize the total profit of marketing items and non-marketing items among all choices. 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 before marketing is given by $prof(I_a, t_i)$. Let $S \subset I$ be a set of selected items. In each transaction t_i, we define two symbols, t'_i and d_i, for the calculation of the total profit.

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

Symbol	Description
$t_1, t_2, ..., t_m$	given transactions
$I_1, I_2, ..., I_n$	given items
$prof(I_a, t_i)$	original profit of item I_a in transaction t_i
$I = \{I_1, I_2, ..., I_n\}$	Itemset of all items
S	set of the selected items
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

Before going to define the profit generated after marketing, we are going to give the original profit before marketing for all transactions.

Definition 11 (Profit Before Marketing) *The original profit $Profit_0$ before marketing for all transactions is defined as:*

$$Profit_0 = \sum_{i=1}^{m} \sum_{I_a \in t_i} prof(I_a, t_i)$$

Now, we are going to describe the phenomenon after marketing. Suppose we select a subset S of marketing items. Marketing actions can be taken for all items in the set S (e.g. discount for the items). Let us consider a transaction t_i containing the marketing items I_a and non-marketing items I_b. If we market item I_a with cost $cost(I_a, t_i)d$ (e.g. discount of each item), the profit of item I_a after marketing in transaction t_i will become $prof(I_a, t_i) - cost(I_a, t_i)$. After the marketing actions are taken, more marketing items, says I_a, will be purchased. The increase ratio of item I_a is defined by $\alpha(\{I_a\})$ [1]. If $\alpha(\{I_a\}) = 1$, then there is no increase of the sales of items I_a. If $\alpha(\{I_a\}) = 2$, then the sales of I_a is double compared with the sales before marketing.

On the other hand, without the consideration of cross-selling effect due to marketing, the profit of non-marketing items I_b is still $prof(I_b, t_i)$. With the consideration of cross-selling effects, some of the non-marketing items I_b will be purchased more if there is an increase of sales of marketing items I_a. The cross-selling factor is modelled by $csfactor(T, I_b)$, where T is a set of marketing items I_a, and $0 \leq csfactor(T, I_b) \leq 1$. That is, more customers may come to buy item I_b if some other items in T are being marketed. $csfactor(T, I_b)$ is the fraction of profit of item I_b that will be boosted in a transaction if the items in set T is being marketed. The increase of the sale of item I_b is modelled by $(\alpha(T) - 1)csfactor(T, I_b)$, where $\alpha(T)$ [2] is the increase ratio of the purchase of items in T. If $\alpha(T) = 1$, then there is no increase of sales of marketing items in set T. So, there is no increase of sales of non-marketing item I_b. The term $(\alpha(T) - 1)csfactor(T, I_b)$ becomes zero. Similarly, if $\alpha(T) = 2$, the sales of items in set T is doubled. Thus, the increase of sales is modelled by $csfactor(T, I_b)$.

Definition 12 (Profit After Marketing) *The profit after marketing $Profit_1$ is*

[1] $alpha(I_i)$ generalizes the case when the $\alpha(I_i)$ is fixed for all items I_i. We note that different items may have their different increase ratio of the sales (i.e. $\alpha(I_i)$). However, it is difficult to predict this parameter $\alpha(I_i)$ for each item I_i. For simplicity, we set all $\alpha(I_i)$ to be the same (e.g. α_0) in this thesis, which is the same as [21, 42].

[2] If $\alpha(I_i) = \alpha_0$ for all i, then it is easy to see that $\alpha(T) = \alpha_0$ for any T (a subset of I) .

defined as follows.

$$\begin{aligned} Profit_1 &= \sum_{i=1}^{m}\Bigg[\sum_{I_a\in t'_i}\alpha(\{I_a\})(prof(I_a,t_i)-cost(I_a,t_i)) \\ &\quad + \sum_{I_b\in d_i}(1+(\alpha(t'_i)-1)csfactor(t'_i,I_b))prof(I_b,t_i)\Bigg] \end{aligned}$$

The objective of marketing is to increase the profit gain compared with the profit before marketing. The profit gain is defined as follows.

Definition 13 (Profit Gain) *Profit gain is :*

$$\textit{Profit Gain} = Profit_1 - Profit_0$$

From the above definition, we can derive the profit gain as follows.

$$\begin{aligned} \text{Profit Gain} &= Profit_1 - Profit_0 \\ &= \sum_{i=1}^{m}\Bigg[\sum_{I_a\in t'_i}\alpha(\{I_a\})(prof(I_a,t_i)-cost(I_a,t_i)) \\ &\quad + \sum_{I_b\in d_i}(1+(\alpha(t'_i)-1)csfactor(t'_i,I_b))prof(I_b,t_i)\Bigg] \\ &\quad -\sum_{i=1}^{m}\sum_{I_a\in t_i} prof(I_a,t_i) \quad \text{(By Definition 11 and 12)} \\ &= \sum_{i=1}^{m}\Bigg[\sum_{I_a\in t'_i}\alpha(\{I_a\})(prof(I_a,t_i)-cost(I_a,t_i)) \\ &\quad + \sum_{I_b\in d_i}(1+(\alpha(t'_i)-1)csfactor(t'_i,I_b))prof(I_b,t_i)\Bigg] \\ &\quad -\sum_{i=1}^{m}\Bigg[\sum_{I_a\in t'_i} prof(I_a,t_i) + \sum_{I_b\in d_i} prof(I_b,t_i)\Bigg] \\ &= \sum_{i=1}^{m}\Bigg[\sum_{I_a\in t'_i}[(\alpha(\{I_a\})-1)prof(I_a,t_i) - \alpha(\{I_a\})cost(I_a,t_i)] \end{aligned}$$

$$+ \sum_{I_b \in d_i} (\alpha(t'_i) - 1) csfactor(t'_i, I_b) prof(I_b, t_i) \Bigg] \quad (3.1)$$

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

This problem is at least as difficult as the following decision problem, which we call the decision problem for ISM.

ISM Decision Problem: Given a set of items and a set of transactions with profits assigned to each item in each transaction, a minimum profit gain G, and cross-selling factors, $csfactor()$, can we pick a set S such that $Profit\ Gain \geq G$?

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. In our proof in the following, we consider the very simple version where $csfactor(t'_i, I_a) = 1$ for any non-empty set of t'_i. That is, any selected items in the transaction will increase the profit of the other items. This may be a much simplified version of the problem, but it is still very difficult.

Theorem 4 (NP-hardness) *The item selection for marketing (ISM) decision problem where $csfactor(t'_i, I_a) = 1$ for $d_i \neq \phi$ and $csfactor(t'_i, I_a) = 0$ for $d_i = \phi$ is NP-hard.*

Proof:

We shall transform the problem of MAX CUT to the ISM problem. MAX CUT [22] is an NP-complete problem defined as follows:

MAX CUT:

Given a graph - (V, E) with weight $w(e) = 1$ for each $e \in E$ and positive integer K, is there a partition of V into disjoint sets V_1 and V_2 such that the sum of the weights of the edges from E that have one endpoint in V_1 and one endpoint in V_2 is at least K?

The transformation from MAXCUT to ISM problem is described as follows.

1. $G = K$
2. $\alpha(\{I_a\}) = 2$
3. $\alpha(t'_i) = 2$
4. For each vertex $v \in V$, construct an item.
5. For each edge $e \in E$, where $e = (v_1, v_2)$, create a transaction with 2 items $\{v_1, v_2\}$.
6. Set $prof(I_j, t_i) = 1$ and $cost(I_a, t_i) = 0.5$, where t_i is a transaction created in the above, $i = 1, 2, ..., |E|$, and I_j is an item in t_i.

$$\begin{aligned}
\text{Profit Gain} &= \sum_{i=1}^{m} \left[\sum_{I_a \in t'_i} ((2-1) \times 1 - 2 \times 0.5) \right. \\
&\quad \left. + \sum_{I_b \in d_i} (2-1) \times csfactor(t'_i, I_b) \times 1 \right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(By (3.1))} \\
&= \sum_{i=1}^{m} \left[0 + \sum_{I_b \in d_i} csfactor(t'_i, I_b) \right] \\
&= \sum_{i=1}^{m} \sum_{I_b \in d_i} csfactor(t'_i, I_b)
\end{aligned}$$

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 ISM, the original MAX CUT problem is also solved. Since MAX CUT is an NP-complete problem, ISM problem is NP-hard.

Consider the case where the cut size of the bipartition of the graph G is K. Let the set of corresponding edges for the cut be C. That is, in the ISM problem, each corresponding transaction should contain a pair of items of different partitions. It is easy to see that if the set of items in the one of the partition are chosen in the set S then the profit gain will be greater than G. That is, the vertices in the one of the partition in MAXCUT is the solution of the marketing items in ISM. T_c are the only transaction that contribute to the profit gain. In a transaction not in T_c, if an item I_i in C exists, it co-exists with another item which is also in C, since the csfactor is 0, the profit gain from item I_i in the transaction is 0.

Let us illustrate the above proof with an example. If there is a MAX CUT with 3-edge cut shown in Figure 3.1, there will be transactions of the following form for I_a, I_b, I_c and I_d:

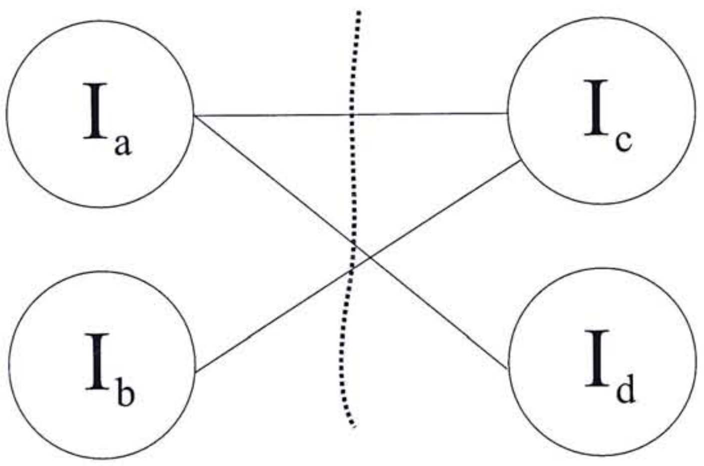

Figure 3.1: A solution of MAXCUT problem

Transaction No.	I_a	I_b	I_c	I_d
1	1	0	1	0
2	1	0	0	1
3	0	1	1	0

3.4 Association Based Cross-Selling Effect

$csfactor(t'_i, I_j)$ is modelled by $conf(\diamond t'_i \rightarrow I_j)$. The reason is described as follows. A transaction can be viewed as a customer behavior. In transaction t_i, there are the cross-selling effect between any marketing items I_a in t'_i and non-marketing items in set d_i. Let us consider some cases. If all items in t_i are being marketed, then there are non-marketing items, the profit gain is the difference between profit of marketing items before marketing and that after marketing. If all items in t_i are not marketed, as there are no marketing items, in transaction t_i, there is no cross-selling effect for marketing items in transaction t_i. Thus, the profit gain due to marketing becomes zero. Now, we are going to consider the case of transaction containing both marketing items and non-marketing items. Suppose the customer purchases any marketing items in set t'_i, always purchases non-marketing items I_b. This phenomenon is modelled by a gain rule $\diamond t'_i \to I_b$. The definition of the confidence of this rule is the same as the usual definition of association rules. That is,

$$\frac{\text{no. of transactions containing any items in set } t'_i \text{ and item } I_b}{\text{no. of transactions containing any items in set } t'_i}$$

The greater the value is, the greater the cross-selling effect is.

3.5 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 [25]. 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 ISM 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 to be marketed 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 $f^T s + \frac{1}{2} s^T Q s$.

We are first describing the approach of approximating the profit gain P in quadratic form in Section 3.5.1. Then, based on this quadratic form, we will outline and describe our algorithm in Section 3.5.2.

3.5.1 Quadratic Form

In this section, we are going to describe how to approximate the total profit gain P. We have some assumptions. For simplicity, we assume $\alpha(\{I_i\}) = \alpha_0$ for $i = 1, 2, ..., n$. If $\alpha(\{I_i\})$ are all equal to α_0 for $i = 1, 2, ..., n$, and I' is a subset of set $I = \{I_1, I_2, ... I_n\}$, then $\alpha(I') = \alpha_0$.

Now, we are going to describe how we approximate confidence $conf(\diamond t'_i \to I_j)$ in Theorem 5. Before describing it, we would like to state Lemma 5.

Lemma 5 *Suppose* $c(\mathbf{s}) = \frac{\mathbf{a^T s}}{\mathbf{b^T s + s^T M s}}$, *where* $\mathbf{s}, \mathbf{a}$ *and* $\mathbf{b}$ *are* n*-dimension vectors, and* $\mathbf{M}$ *is an* n *by* n *matrix.* $c(\mathbf{s})$ *is approximated and linearized by* $c(\mathbf{s_0}) + |\frac{dc(\mathbf{s})}{d\mathbf{s}}|^T_{\mathbf{s}=\mathbf{s_0}}(\mathbf{s} - \mathbf{s_0})$, *where* s_0 *is a fixed* n*-dimension vector and*

$$\frac{dc(\mathbf{s})}{d\mathbf{s}} = \mathbf{e} = \left(e_t \middle| e_t = \frac{\partial c(\mathbf{s})}{\partial s_t} = \frac{(\mathbf{b^T s + s^T M s}) a_t - \mathbf{a^T s}(b_t + 2 \sum_{j=1}^{n} s_j m_{tj})}{(\mathbf{b^T s + s^T M s})^2} \right.$$

$$\left. \text{for } t = 1, 2, ..., n \right)^T$$

Proof:

By Taylor's Series [47], function $c(\mathbf{s})$ can be written as:

$$\begin{aligned} c(\mathbf{s}) &= c(\mathbf{s_0}) + \left|\frac{dc(\mathbf{s})}{d\mathbf{s}}\right|_{\mathbf{s}=\mathbf{s_0}}^{T} (\mathbf{s}-\mathbf{s_0}) + \frac{1}{2}(\mathbf{s}-\mathbf{s_0})^T \left|\frac{d^2c(\mathbf{s})}{d\mathbf{s}^2}\right|_{\mathbf{s}=\mathbf{s_0}}^{T} (\mathbf{s}-\mathbf{s_0}) + \ldots \\ &\approx c(\mathbf{s_0}) + \left|\frac{dc(\mathbf{s})}{d\mathbf{s}}\right|_{\mathbf{s}=\mathbf{s_0}}^{T} (\mathbf{s}-\mathbf{s_0}) \end{aligned}$$

$c(\mathbf{s})$ is approximated by taking the first two terms in Taylor's Series.

By differentiating $c(s)$ with respect to s, $\frac{dc(\mathbf{s})}{d\mathbf{s}} = \mathbf{e}$ can be obtained by

$$\frac{dc(\mathbf{s})}{d\mathbf{s}} = \mathbf{e} = \left(e_t \middle| e_t = \frac{\partial c(\mathbf{s})}{\partial s_t} = \frac{(\mathbf{b^T s} + \mathbf{s^T M s})a_t - \mathbf{a^T s}(b_t + 2\sum_{j=1}^{n} s_j m_{tj})}{(\mathbf{b^T s} + \mathbf{s^T M s})^2} \right.$$
$$\left. \text{for } t = 1, 2, ..., n \right)^T$$

□

Theorem 5 *The confidence $conf(\diamond t'_i \to I_j)$ can be approximated by $c^0_{ij} + \mathbf{c^1_{ij}}^{\mathbf{T}}(\mathbf{s}-\mathbf{s_0})$, where c^0_{ij} is a constant, and $\mathbf{c^1_{ij}}$ and $\mathbf{s_0}$ is an n-dimension vector. Let $c(\mathbf{s}) = \frac{\mathbf{a^T s}}{\mathbf{b^T s} + \mathbf{s^T M s}}$, where*

$$\begin{aligned} \mathbf{a} &= (a_k | a_k = n_{kj} t_{ik} \textit{ for } k = 1, 2, ..., n)^T \\ \mathbf{b} &= (b_k | b_k = n_k t_{ik} \textit{ for } k = 1, 2, ..., n)^T \\ \mathbf{M} &= (m_{kl} | m_{kl} = -n_{kl} t_{ik} t_{il} \textit{ for } k, l = 1, 2, ..., n) \end{aligned}$$

c^0_{ij} and $\mathbf{c^1_{ij}}^{\mathbf{T}}$ have the following values and $\mathbf{s_0}$ is a fixed vector.

$$\begin{aligned} c^0_{ij} &= c(\mathbf{s_0}) \\ \mathbf{c^1_{ij}}^{\mathbf{T}} &= \left|\frac{dc(\mathbf{s})}{d\mathbf{s}}\right|_{\mathbf{s}=\mathbf{s_0}}^{T} \end{aligned}$$

Proof:

$$
\begin{aligned}
& conf(\diamond t'_i \rightarrow I_j) \\
= \;& \frac{\text{no. of transactions containing } I_j \text{ and at least one item in set } t'_i}{\text{no. of transactions containing at least one item in set } t'_i} \\
\approx \;& \frac{\sum_{k=1}^{n} n_{kj} t'_{ik}}{\sum_{k=1}^{n} n_k t'_{ik} - \sum_{k=1}^{n} t'_{ik} \sum_{l=1}^{n} n_{kl} t'_{il}} \\
& \quad \text{(by the principle of inclusion-exclusion and approximation)} \\
= \;& \frac{\sum_{k=1}^{n} n_{kj} t_{ik} s_k}{\sum_{k=1}^{n} n_k t_{ik} s_k - \sum_{k=1}^{n} t_{ik} s_k \sum_{l=1}^{n} n_{kl} t_{il} s_l} \\
= \;& \frac{\sum_{k=1}^{n} n_{kj} t_{ik} s_k}{\sum_{k=1}^{n} n_k t_{ik} s_k - \sum_{k=1}^{n} s_k \sum_{l=1}^{n} n_{kl} t_{ik} t_{il} s_l} \\
= \;& \frac{\mathbf{a^T s}}{\mathbf{b^T s + s^T M s}}
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbf{a} &= (a_k | a_k = n_{kj} t_{ik} \text{ for } k = 1, 2, ..., n)^T \\
\mathbf{b} &= (b_k | b_k = n_k t_{ik} \text{ for } k = 1, 2, ..., n)^T \\
\mathbf{M} &= (m_{kl} | m_{kl} = -n_{kl} t_{ik} t_{il} \text{ for } k, l = 1, 2, ..., n)
\end{aligned}
$$

Let $c(\mathbf{s}) = \frac{\mathbf{a^T s}}{\mathbf{b^T s + s^T M s}}$.

By Lemma 5 and choosing a fiexed vector s_0,

$$
\begin{aligned}
c(\mathbf{s}) &\approx c(\mathbf{s_0}) + \left| \frac{dc(\mathbf{s})}{d\mathbf{s}} \right|^T_{\mathbf{s}=\mathbf{s_0}} (\mathbf{s} - \mathbf{s_0}) \\
&= c^0_{ij} + \mathbf{c^1_{ij}}^{\mathbf{T}} (\mathbf{s} - \mathbf{s_0})
\end{aligned}
$$

where

$$
c^0_{ij} = c(\mathbf{s_0})
$$

$$\mathbf{c_{ij}^1}^{\mathbf{T}} = \left|\frac{dc(\mathbf{s})}{d\mathbf{s}}\right|^T_{\mathbf{s}=\mathbf{s_0}}$$

□

From the above proof, we can observe that there are two types of approximations in the calculation of the confidence. The first approximation is to remove the terms of order greater than 1 in the Taylor series used in the function $c(\mathbf{s})$. However, if the chosen initial point $\mathbf{s_0}$ is near to the optimal point, the error due to the approximation is small. In our experiment, we choose the approach of direct marketing for the initialization of $\mathbf{s_0}$. The second approximation is the approximated evaluation of the confidence by using the principle of inclusion-exclusion. As the objective function includes the evaluation of confidence with these two approximations, the objective profit gain approximated in the quadratic programming may be less accurate.

Now, we are going to describe how the profit gain P can be approximated in the quadratic form in Theorem 6.

Theorem 6 *Let* $\mathbf{c_{ij}^1} = \phi = (\phi_k)$. *The profit gain* P *can be approximated by the following form.*

$$P = L + \mathbf{f^T}\mathbf{s} + \frac{1}{2}\mathbf{s^T}\mathbf{H}\mathbf{s}$$

where

$$\begin{aligned}
L &= \sum_{i=1}^{m} L_i = \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) \\
\mathbf{f} &= \sum_{i=1}^{m} \mathbf{a_i} \\
&= \left(f_j | f_j = \sum_{i=1}^{m}\left(t_{ij}\left[(\alpha_0 - 1)prof(I_j, t_i) - \alpha_0 cost(I_j, t_i)\right]\right.\right. \\
&\quad + \sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k, t_i)
\end{aligned}$$

$$- t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c^0_{ij} - \mathbf{c^1_{ij}}^{\mathbf{T}}\mathbf{s_0})\phi_j\Big) \; for \; j = 1, 2, ..., n\Big)^T$$

$$\mathbf{H} \;=\; \sum_{i=1}^{m}\mathbf{H_i} = (h_{jk}|h_{jk} = -2\sum_{i=1}^{m} t_{ij}(\alpha_0 - 1)prof(I_j, t_i)\phi_k \; for \; j, k = 1, 2, ..., n)$$

Proof:

From (3.1), the profit gain is:

$$\begin{aligned}
Profit\ Gain \;&=\; \sum_{i=1}^{m}\left[\sum_{I_a \in t'_i}[(\alpha(\{I_a\}) - 1)prof(I_a, t_i) - \alpha(\{I_a\})cost(I_a, t_i)]\right.\\
&\quad \left. + \sum_{I_b \in d_i}(\alpha(t'_i) - 1)conf(\diamond t'_i \to I_j)prof(I_b, t_i)\right]\\
&=\; \sum_{i=1}^{m}\left[\sum_{j=1}^{n} t'_{ij}\,[(\alpha_0 - 1)prof(I_j, t_i) - \alpha_0 cost(I_j, t_i)]\right.\\
&\quad \left. + \sum_{j=1}^{n} d_{ij}(\alpha_0 - 1)conf(\diamond t'_i \to I_j)prof(I_j, t_i)\right]\\
&=\; \sum_{i=1}^{m}\left[\sum_{j=1}^{n} t_{ij}s_j\,[(\alpha_0 - 1)prof(I_j, t_i) - \alpha_0 cost(I_j, t_i)]\right.\\
&\quad \left. + \sum_{j=1}^{n}(t_{ij} - t'_{ij})(\alpha_0 - 1)conf(\diamond t'_i \to I_j)prof(I_j, t_i)\right]\\
&=\; \sum_{i=1}^{m}\left[\sum_{j=1}^{n} t_{ij}\,[(\alpha_0 - 1)prof(I_j, t_i) - \alpha_0 cost(I_j, t_i)]\,s_j\right.
\end{aligned}$$

$$\left. + \sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)(\alpha_0 - 1)conf(\diamond t'_i \to I_j)prof(I_j, t_i)\right] \qquad (3.2)$$

We now consider the term $\sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)(\alpha_0 - 1)conf(\diamond t'_i \to I_j)prof(I_j, t_i)$.

$$\begin{aligned}
&\sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)(\alpha_0 - 1)conf(\diamond t'_i \to I_j)prof(I_j, t_i)\\
=\;& (\alpha_0 - 1)\sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)conf(\diamond t'_i \to I_j)prof(I_j, t_i)
\end{aligned}$$

$$
\begin{aligned}
&\approx (\alpha_0 - 1)\sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)(c_{ij}^0 + \mathbf{c_{ij}^1}^{\mathbf{T}}(\mathbf{s} - \mathbf{s_0}))prof(I_j, t_i) \qquad \text{(By Theorem 5)} \\
&= (\alpha_0 - 1)\sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)(c_{ij}^0 + \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s} - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})prof(I_j, t_i) \\
&= (\alpha_0 - 1)\sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)((c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s})prof(I_j, t_i) \\
&= (\alpha_0 - 1)\sum_{j=1}^{n}(t_{ij}(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + t_{ij}\mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s} - t_{ij}s_j(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) \\
&\quad -t_{ij}s_j\mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s})prof(I_j, t_i) \\
&= (\alpha_0 - 1)\left(\sum_{j=1}^{n} t_{ij}(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) \times prof(I_j, t_i) + \sum_{j=1}^{n} t_{ij}\mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s} \times prof(I_j, t_i) - \right. \\
&\quad \left. \sum_{j=1}^{n} t_{ij}s_j(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) \times prof(I_j, t_i) - \sum_{j=1}^{n} t_{ij}s_j\mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s} \times prof(I_j, t_i)\right) \\
&= (\alpha_0 - 1)\left(\sum_{j=1}^{n} t_{ij}prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)\mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s} - \right. \\
&\quad \left. \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})s_j - \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)s_j\mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s}\right) \\
&= (\alpha_0 - 1)\left(\sum_{j=1}^{n} t_{ij}prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)\sum_{k=1}^{n}\phi_k s_k - \right. \\
&\quad \left. \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})s_j - \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)s_j\sum_{k=1}^{n}\phi_k s_k\right) \\
&\qquad\qquad\qquad\qquad \text{where } \mathbf{c_{ij}^1} = \phi = (\phi_k) \\
&= (\alpha_0 - 1)\left(\sum_{j=1}^{n} t_{ij}prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \sum_{k=1}^{n}(\sum_{j=1}^{n} t_{ij}prof(I_j, t_i)\phi_k)s_k - \right. \\
&\quad \left. \sum_{j=1}^{n} t_{ij}prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})s_j - \sum_{j=1}^{n} s_j\sum_{k=1}^{n} t_{ij}prof(I_j, t_i)\phi_k s_k\right) \\
&= \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \\
&\quad \sum_{k=1}^{n}(\sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j, t_i)\phi_k)s_k -
\end{aligned}
$$

$$
\begin{aligned}
& \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})s_j - \\
& \sum_{j=1}^{n} s_j \sum_{k=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k s_k \\
= \; & \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \\
& \sum_{j=1}^{n}(\sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k,t_i)\phi_j)s_j - \\
& \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})s_j - \\
& \sum_{j=1}^{n} s_j \sum_{k=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k s_k \\
= \; & \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \\
& \sum_{j=1}^{n}\left(\sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k,t_i) - t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j\right)s_j \\
& -\sum_{j=1}^{n} s_j \sum_{k=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k s_k
\end{aligned}
$$

From (3.2),

$$
\begin{aligned}
& Profit\ Gain \\
= \; & \sum_{i=1}^{m}\left[\sum_{j=1}^{n} t_{ij}\left[(\alpha_0 - 1)prof(I_j,t_i) - \alpha_0 cost(I_j,t_i)\right]s_j \right. \\
& \left. + \sum_{j=1}^{n}(t_{ij} - t_{ij}s_j)(\alpha_0 - 1)conf(\diamond t_i' \to I_j)prof(I_j,t_i)\right] \\
= \; & \sum_{i=1}^{m}\left[\sum_{j=1}^{n} t_{ij}\left[(\alpha_0 - 1)prof(I_j,t_i) - \alpha_0 cost(I_j,t_i)\right]s_j + \right. \\
& \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \\
& \sum_{j=1}^{n}\left(\sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k,t_i) - t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j\right)s_j
\end{aligned}
$$

$$
\begin{aligned}
& - \sum_{j=1}^{n} s_j \sum_{k=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k s_k \Bigg] \\
= & \sum_{i=1}^{m} \Bigg[\sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \\
& \sum_{j=1}^{n} t_{ij}\left[(\alpha_0 - 1)prof(I_j,t_i) - \alpha_0 cost(I_j,t_i)\right] s_j \\
& + \sum_{j=1}^{n} \left(\sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k,t_i) - t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j \right) s_j \\
& - \sum_{j=1}^{n} s_j \sum_{k=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k s_k + \Bigg] \\
= & \sum_{i=1}^{m} \Bigg[\sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) + \\
& \sum_{j=1}^{n} \Bigg(t_{ij}\left[(\alpha_0 - 1)prof(I_j,t_i) - \alpha_0 cost(I_j,t_i)\right] + \\
& \sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k,t_i) - t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j \Bigg) s_j + \\
& - \sum_{j=1}^{n} s_j \sum_{k=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k s_k \Bigg] \\
= & \sum_{i=1}^{m} \Big[L_i + \mathbf{a_i^T s} + \mathbf{s^T H_i s} \Big]
\end{aligned}
$$

where

$$
\begin{aligned}
L_i &= \sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) \\
\mathbf{a_i} &= \mathbf{g} \\
&= \Big(g_i | g_j = t_{ij}\left[(\alpha_0 - 1)prof(I_j,t_i) - \alpha_0 cost(I_j,t_i)\right] \\
&\qquad + \sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k,t_i) \\
&\qquad - t_{ij}(\alpha_0 - 1)prof(I_j,t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j \text{ for } j = 1,2,...,n \Big)^T \\
\mathbf{H_i} &= \mathbf{A} \\
&= \Big(a_{jk} | a_{jk} = -2t_{ij}(\alpha_0 - 1)prof(I_j,t_i)\phi_k \text{ for } j,k = 1,2,..,n \Big)
\end{aligned}
$$

$$\begin{aligned}
Profit\ Gain &= \sum_{i=1}^{m}(L_i + \mathbf{a_i^T s} + \frac{1}{2}\mathbf{s^T H_i s}) \\
&= \sum_{i=1}^{m} L_i + \sum_{i=1}^{m}(\mathbf{a_i^T s}) + \sum_{i=1}^{m}(\frac{1}{2}\mathbf{s^T H_i s}) \\
&= \sum_{i=1}^{m} L_i + (\sum_{i=1}^{m}\mathbf{a_i^T})\mathbf{s} + \frac{1}{2}\mathbf{s^T}(\sum_{i=1}^{m}\mathbf{H_i})\mathbf{s} \\
&\qquad \text{(by Lemma 1 and Lemma 2)} \\
&= L + \mathbf{f^T s} + \frac{1}{2}\mathbf{s^T H s}
\end{aligned}$$

where

$$\begin{aligned}
L &= \sum_{i=1}^{m} L_i = \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0}) \\
\mathbf{f} &= \sum_{i=1}^{m}\mathbf{a_i} \\
&= \left(f_j | f_j = \sum_{i=1}^{m}\left(t_{ij}\left[(\alpha_0 - 1)prof(I_j, t_i) - \alpha_0 cost(I_j, t_i)\right] + \right.\right. \\
&\quad \sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k, t_i) \\
&\quad \left.\left. - t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j\right) \text{ for } j = 1, 2, ..., n\right)^T \\
\mathbf{H} &= \sum_{i=1}^{m}\mathbf{H_i} = (h_{jk} | h_{jk} = -2\sum_{i=1}^{m} t_{ij}(\alpha_0 - 1)prof(I_j, t_i)\phi_k \text{ for } j, k = 1, 2, ..., n)
\end{aligned}$$

□

It is noted that the above quadratic form is non-standard as there is a constant L in the above form and the matrix H is not symmetric. By the following corollary, the above non-standard quadratic form is transformed to the standard quadratic from.

Corollary 2 *Profit Gain P can be approximated by $P = \mathbf{f'^T s'} + \frac{1}{2}\mathbf{s'^T Q s'}$ where $\mathbf{Q}$ is a symmetric $n+1$ by $n+1$ matrix, and $\mathbf{s'}$ and $\mathbf{f'}$ are $(n+1)$-dimension vectors.*

Proof:

$$\begin{aligned} P &= L + \mathbf{f^T s} + \frac{1}{2}\mathbf{s^T H s} \\ &= \mathbf{f'^T s'} + \frac{1}{2}\mathbf{s'^T Q s'} \qquad \text{(by Lemma 3)} \end{aligned}$$

where

$$\mathbf{Q} = (q_{ij}) \text{ and } q_{ij} = \begin{cases} \frac{h_{ij}+h_{ji}}{2} & \text{for all } i,j = 1,2,...,n \\ 0 & \text{for } i = n+1 \text{ and } j = 1,2,...,n+1 \\ 0 & \text{for } j = n+1 \text{ and } i = 1,2,...,n+1 \end{cases}$$

$$\mathbf{f'} = (f'_j)^T \text{ and } f'_j = \begin{cases} f_j & \text{for } j = 1,2,...,n \\ L & \text{for } j = n+1 \end{cases}$$

$$\mathbf{s'} = (s'_j)^T \text{ and } s'_j = \begin{cases} s_j & \text{for } j = 1,2,...,n \\ 1 & \text{for } j = n+1 \end{cases}$$

□

It is noted that the above quadratic form is dependent on the initial data point of n dimension binary vector, says s_0. In other words, the above quadratic form requires an input argument - s_0. After the above quadratic form takes s_0 as its argument, the objective quadratic equation is formed. We can solve this objective quadratic equation with a standard solver.

3.5.2 Algorithm

Our quadratic programming approach is an iterative algorithm. At the first iteration, a random data point s_0 of n dimension binary vector is generated. Then, the value of this data point is put into the quadratic equation described above. The objective quadratic equation just formed is solved with a standard quadratic solver. We can find a solution s_1 which can maximize the objective quadratic function. For next iteration, we take the solution just found s_1 as the initial data point of the quadratic form. We repeat the process until some stopping criteria are reached. Stopping criteria can be the number of loops and the convergence (e.g. the difference between the previous objective values and the current objective values).

The following shows the outline of our algorithm.

Algorithm

1. Randomly start at initial point $s_0^{(0)}$
2. Use quadratic programming approach with objective function
$$P = \mathbf{f}'^{\mathbf{T}}\mathbf{s}' + \frac{1}{2}\mathbf{s}'^{\mathbf{T}}\mathbf{Q}\mathbf{s}'$$
to find a solution $s_0^{(1)}$.
3. Repeat the above step with the initial point $s_0^{(1)}$ until some stopping criteria are reached.

3.5.3 Example

We are going to illustrate our QP in the following example.

The following table shows 3 transactions and 3 items. Set $\alpha = 2$.

	I_1	I_2	I_3
t_1	1	1	0
t_2	1	1	0
t_3	1	0	1

The number of occurrence n_i of a single item I_i is shown in the following table:

i	n_i
1	3
2	2
3	1

The number of co-occurrence n_{ij} of two items, item I_i and item I_j. The values are shown in the following table.

i/j	1	2	3
1	0	2	1
2	2	0	0
3	1	0	0

Now, we are going to calculate vectors **a** and **b** and matrix **M** for each i and j, where $i, j = 1, 2, 3$.

For $i = 1$ and $j = 1$,

$$\mathbf{a} = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $i = 1$ and $j = 2$,

$$\mathbf{a} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $i = 1$ and $j = 3$,

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $i = 2$ and $j = 1$,

$$\mathbf{a} = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $i = 2$ and $j = 2$,

$$\mathbf{a} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $i = 2$ and $j = 3$,

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

For $i = 3$ and $j = 1$,

$$\mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

For $i = 3$ and $j = 2$,

$$\mathbf{a} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

For $i = 3$ and $j = 3$,

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{M} = -\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Suppose the initial vector $\mathbf{s}$ is set to be $(101)^T$. We are going to calculate $c(\mathbf{s}) = \frac{\mathbf{a^T s}}{\mathbf{b^T s + s^T M s}}$ for each i and j, where $i, j = 1, 2, 3$.

We would like to show the results in the following table.

i/j	1	2	3
1	0	0.67	0.33
2	0	0.67	0133
3	0.5	1	0.5

After calculating the function $c(\mathbf{c})$, we are going to evaluate the derivative function $\frac{dc(\mathbf{s})}{d\mathbf{s}}$ for each i and j, where $i, j = 1, 2, 3$.

For $i = 1$ and $j = 1$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0 \\ 0.66 \\ 0 \end{pmatrix}$ For $i = 1$ and $j = 2$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0 \\ 0.44 \\ 0 \end{pmatrix}$

For $i = 1$ and $j = 3$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0 \\ 0.22 \\ 0 \end{pmatrix}$ For $i = 2$ and $j = 1$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0 \\ 0.67 \\ 0 \end{pmatrix}$

For $i = 2$ and $j = 2$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0 \\ 0.44 \\ 0 \end{pmatrix}$ For $i = 2$ and $j = 3$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0 \\ 0.22 \\ 0 \end{pmatrix}$

For $i = 3$ and $j = 1$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} -0.25 \\ 0 \\ 0.75 \end{pmatrix}$ For $i = 3$ and $j = 2$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0.5 \\ 0 \\ 0.5 \end{pmatrix}$

For $i = 3$ and $j = 3$, $\frac{dc(\mathbf{s})}{ds} = \begin{pmatrix} 0.25 \\ 0 \\ 0.25 \end{pmatrix}$

We are now going to calculate the value of L. In the equation of L, we need to calculate the term

$$t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c^0_{ij} - \mathbf{c^1_{ij}}^{\mathbf{T}}\mathbf{s_0})$$

for each transaction t_i and each item I_j, for $i, j = 1, 2, 3$. The following table shows the value of such term for different values of i and j.

i/j	1	2	3
1	0	3.3	0
2	0	1.98	0
3	0	0	0

By summing up the above terms, the value of L is equal to 6.67.

Then, $\mathbf{f}$ is to be evaluated.

Let us consider the first entry f_1 in $\mathbf{f}$. Similarly, for $j = 1$,we calculate each sub-term

$$t_{ij}\left[(\alpha_0 - 1)prof(I_j, t_i) - \alpha_0 cost(I_j, t_i)\right] +$$

$$\sum_{k=1}^{n} t_{ik}(\alpha_0 - 1)prof(I_k, t_i) - t_{ij}(\alpha_0 - 1)prof(I_j, t_i)(c_{ij}^0 - \mathbf{c_{ij}^1}^{\mathbf{T}}\mathbf{s_0})\phi_j$$

for different transactions (i.e. different values of i, where $j = 1, 2, 3$).

i	Value of the Term
1	5
2	5
3	1

The sum of the above values (i.e. f_1) is equal to 11.

After the calculation of f_j for other values (i.e. $j = 2, 3$), f is equal to

$$\begin{pmatrix} 11.00 \\ 1.04 \\ 13.00 \end{pmatrix}$$

H is finally computed.

For the entry H_{jk} where $j = 1$ and $k = 1$, similarly, we need to compute each term $t_{ij}(\alpha_0 - 1)prof(I_j, t_i)\phi_k$ for different transactions t_i where $i = 1, 2, 3$.

i	Value of the Term
1	0
2	0
3	0.5

After the calculation of H_{jk} for other values, **H** is equal to

$$\begin{pmatrix} 0.50 & -1.33 & -1.00 \\ -1.33 & -8.89 & 0.00 \\ -1.00 & 0.00 & -0.50 \end{pmatrix}$$

At this step, we can calculate vector $\mathbf{f'}$ and matrix $\mathbf{Q}$.

From L and $\mathbf{f}$,$\mathbf{f'} = \begin{pmatrix} 11.00 \\ 1.04 \\ 13.00 \\ 6.67 \end{pmatrix}$

From **H**, **Q** is equal to $\begin{pmatrix} 0.50 & -1.33 & -1.00 & 0.00 \\ -1.33 & -8.89 & 0.00 & 0.00 \\ -1.00 & 0.00 & -0.50 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$

Then, we can solve the quadratic equation

$$P = \mathbf{f}'^{\mathbf{T}}\mathbf{s}' + \frac{1}{2}\mathbf{s}'^{\mathbf{T}}\mathbf{Q}\mathbf{s}'$$

Then, by an optimization tool, we found that the solution $(101)^T$ which gives the best value. We use this vector just obtained to be an initial point for the next iteration. We repeat the process until some stopping criteria is reached.

3.6 Hill-Climbing Approach

We are going to propose hill-climbing algorithm to deal with this problem. Let $f(S)$ be the function of the profit gain of the selection S of marketing items. Initially, we assign $S = \{\}$. Then, we will calculate $f(S \cup \{I_a\})$ for each item I_a. Then, we choose the item I_b with the greatest value of $f(S \cup \{I_b\})$ and insert it into set S. The above process repeats for the remaining items whenever $f(S \cup \{I_b\}) > f(S)$.

3.6.1 Efficient Calculation of Formula of Profit Gain

As the formula of the profit gain is computationally intensive, an efficient calculation of this formula is required. Hill Climbing chooses the item with the greatest profit gain for each iteration. Suppose S now contains k items at k-th iteration, for $k = 1, 2, ...$ At the iteration, we store the value of $f(S)$ in a variable f_S. At the $(k+1)$-th iteration, we can calculate $f(S \cup \{I_x\})$ from f_S efficiently for all $I_x \notin S$ and $b = 1, 2, .., n$. Let $\mathcal{T}$ be the set of transactions containing item I_x and at least one item in selection set S. We only calculate $f(S \cup \{I_x\})$ as

$$f(S \cup \{I_x\}) \leftarrow f_S + g(I_x) - h(S, \mathcal{T}) + h(S \cup \{I_x\}, \mathcal{T})$$

where

$$g(I_x) = \sum_{i=1}^{m} [(\alpha(\{I_x\}) - 1)prof(I_x, t_i) - \alpha(\{I_x\})cost(I_x, t_i)]$$

$$h(S, \mathcal{T}) = \sum_{t_i \in \mathcal{T}} \sum_{I_b \in d_i} (\alpha(t'_i) - 1)csfactor(t'_i, I_b)prof(I_b, t_i)$$

assuming all items in set S are selected for marketing. Function $g(I_x)$ is the profit gain of marketing item I_x in all transactions. Function $h(S, \mathcal{T})$ is the profit gain of non-marketing items for the selection S in all transactions in set $\mathcal{T}$.

We can view functions $g(.)$ and $h(.)$ in the another way. Given that all items in set S are chosen to be marketed, the profit gain of the marketing selection, in terms of $g(.)$ and $h(.)$ is:

$$\text{Profit Gain} = \sum_{I_a \in S} g(I_a) + h(S, \mathcal{T})$$

if $\mathcal{T}$ is the set of all transactions in the database.

We are now back to the calculation method of $f(S \cup \{I_x\})$. For the first part, we need to add the profit gain of the newly added marketing item I_x after marketing(i.e. $g(I_x)$) to f_S. For the remaining part, it is easy that we only deal with the transactions in set $\mathcal{T}$ to calculate the profit gain if we want to update the variable f_S for the selection set S which is now added with item I_x. We need to subtract the profit gain of non-marketing items for the selection S in all the transactions in set $\mathcal{T}$ (i.e. $h(S, \mathcal{T})$) and then add the profit gain of non-marketing items for the new selection $S \cup \{I_x\}$ in all the transactions in set $\mathcal{T}$ (i.e. $h(S \cup \{I_x\}, \mathcal{T})$).

As we know that the set $\mathcal{T}$ is typically small compared with the whole database, the calculation of $f(S \cup \{I_x\})$ is efficient.

3.6.2 FP-tree Implementation

The transactions in the database are examined for computation whenever the confidence term $conf(\diamond t'_i \to I_j)$ is calculated. So, we need to do this operation effectively. 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 [23]. 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 processed only once with $\mathcal{FPT}$. By traversing $\mathcal{FPT}$ once, we can count the number of transactions containing any items in set t_i' and item I_b and number of transactions containing any items in set t_i'. From our experiments this mechanism can greatly reduce the overall running time.

3.7 Empirical Study

We have used Pentium IV 2.2GHz PC to conduct our experiments.

In our experiments, we are going to study the profit gain of marketing. Besides, we conducted another kinds of marketing, called *direct marketing*.

In our problem we do not have a specific parameter of the number of marketing items. We choose such parameters in direct marketing by using the following method. After the hill-climbing of our algorithm, there are J resulting items (or marketing items). For direct marketing, we will choose J items with greatest values of Equation (3.1) in Section 3.3 if no cross-selling effect is considered (i.e. $csfactor(t_i', I_b) = 0$ for any set t_i and item I_b).

There are experimental setup for Algorithm QP. In QP, we need to generate an initial data point. The initial data point is a binary vector of (n+1) dimension, where n is the number of items. If item I_j is selected, the j-th position of the vector is marked as 1. In our experiment, by using the approach of direct marketing, the algorithm selects 5% of the items as the selected items. The reason why we choose the approach of direct marketing is to try to make the approximations used in QP better. This is because, in Taylor Series, we have truncate some term with high order. If the initial point is chosen as close as the optimal point, then the evaluated objective value will be more accurate. Besides, we conducted the experiment of QP with 5 iterations. In our experiment, we found that it is sufficient to run with 5 iterations to get the great profit gain. The execution time of QP used in the following graph is the total execution time over 5 iterations.

3.7.1 Data Set

We have used two kinds of data sets - synthetic data set and real data set for our experimental results.

For synthetic data set, we use the IBM synthetic data generator in [10] to generate the data set with the following parameters 500 items, 5,000 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 [28]. We use the same settings as [52]. 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. This data set is called *Synthetic Data Set 1*.

We also generates another synthetic data set (*Synthetic Data Set 2*) with the same setting. But, the number of items and the number of transactions are now set to be 1,000 and 10,000, respectively. This setting gives the same parameter setting as [52].

For real data set, we adopted the data set from BMS WebView-1, which contains clickstream and purchase data collected by a web company and is part of the KDD-Cup 2000 data [34]. There are 59,602 transactions and 497 items. The average transaction size is 2.5. The profit of each item is generated similarly as described above.

3.7.2 Experimental Results

For all data sets, we have studies two situations - discount items and free items. The situation of discount items is to make the selling price of marketing items half. The situation of free items is to make marketing item free of charge.

For synthetic data set, the experimental results of profit gains and execution time against α for the situation of discount items are shown in Figure 3.14 and Figure 3.15. The ones for the situation of free items are shown in Figure 3.16 and Figure 3.17.

For real data set, the experimental results of profit gains and execution time against α for the situation of discount items are shown in Figure 3.18 and Figure 3.19. Those for the situation of free items are shown in Figure 3.20 and Figure 3.21.

Synthetic Data Set 1

1. **Discounted Marketing Items**

   For the scenario of discounted marketing items in the synthetic data set, the graphs of the total profit gain against iterations for different α are shown in Figure 3.2. Figure 3.3 shows the graph of the execution time of different algorithms against α.

   From the experiment, we found that the curve of the profit gain of all algorithms (QP, Hill Climbing and Direct Marketing) increases with α because more customers will purchase non-marketing items if a larger marketing effect is considered. In the graph, the profit gain of Algorithm Hill Climbing is larger than that of Algorithm QP. This is because Algorithm Hill Climbing utilizes the exact formula for convergence. However, algorithm QP approximates the formula of the profit gain, which yields less profit gain. It is noted that the objective function of QP is approximated by two types of approximations. The first one is the removal of the terms with higher degree in the Taylor's series. This approximation can be minimized by choosing a suitable initial point. In our experiment, we adapt the approach of direct marketing so as to minimize the error. The second one is the approximation in using the principle of inclusion-exclusion. The reason for the smaller value of profit gain in QP is such approximations. Algorithm Direct Marketing gives the lowest profit gain as it does not consider any cross-selling effect.

In the above experiment, we conducted the experiment of algorithm QP 5 iterations and chose the best profit gain for the 5 iterations. The graph of the profit gain against iterations for different α values are shown in Figure 3.6 ($\alpha = 1.5$), Figure 3.7($\alpha = 2$), Figure 3.8 ($\alpha = 2.5$) and Figure 3.9($\alpha = 3$).

In each graph for a particular value of α, the curve of the profit gain fluctuates over different α values. This is because different quadratic equations are formed for different iterations as each iteration stats at a new point (or a binary vector). So, different quadratic equations may lead to different optimizations. So, the curve fluctuates. However, in [47], it is a good way to use such a strategy for optimization.

2. **Free Marketing Items**

   The results in the scenario of free marketing items are just similar as the scenario of discounted marketing items.

   The graph of the profit gain against α for different algorithms is shown in Figure 3.4. The graph of the execution time is shown in Figure 3.5.

   The graphs of the profit gain against iterations for different α are shown in Figure 3.10 ($\alpha = 1.5$), Figure 3.11($\alpha = 2$), Figure 3.12 ($\alpha = 2.5$) and Figure

3.13($\alpha = 3$).

In the remaining parts of this thesis, we will use Algorithm "Hill Climbing" for comparison because Algorithm "Hill Climbing" algorithm performs the best compared with Algorithm QP.

Synthetic Data Set 2

1. **Discounted Marketing Items**

   For the scenario of discounted marketing items in the synthetic data set, in Figure 3.14, the profit gain for hill climbing is greater than that for direct marketing. This is because our algorithm of hill climbing considers the cross-selling effect between items, but the direct marketing methodology does not.

   In Figure 3.15, the execution time of hill climbing is greater, compared with direct marketing. The trend of the curve of hill climbing increases with α.

2. **Free Marketing Items**

   For the scenario of free marketing items in the synthetic data set, the difference in profit gains between hill climbing and direct marketing is much larger, compared with the scenario of discounted marketing items. It is noted that there is no profit gain for direct marketing. As we remembered, direct marketing may choose the items, which may not cause a strong cross-selling effect to other profitable items. The items to be marketed are free. Thus, the profit gain for direct marketing becomes low. The execution time of the scenario of free marketing item in Figure 3.17 is similar to that in Figure 3.15.

Real Data Set

In the real data set, the trends of the curves for the scenario of free marketing items and discounted marketing items are just similar to that in the synthetic data set. It is noted that in the scenario of free marketing items, direct marketing may lead to negatived profit gain, which means that direct marketing may make profit lose.

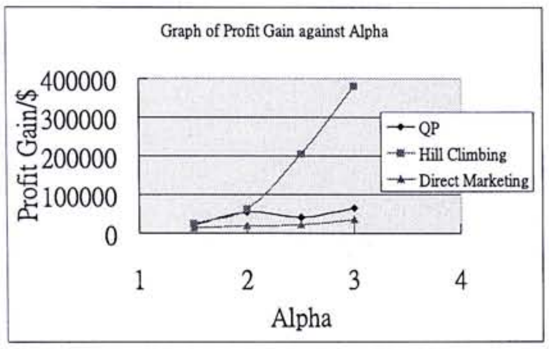

Figure 3.2: Graph of Profit Gains against α for Synthetic Data Set 1 (Discount)

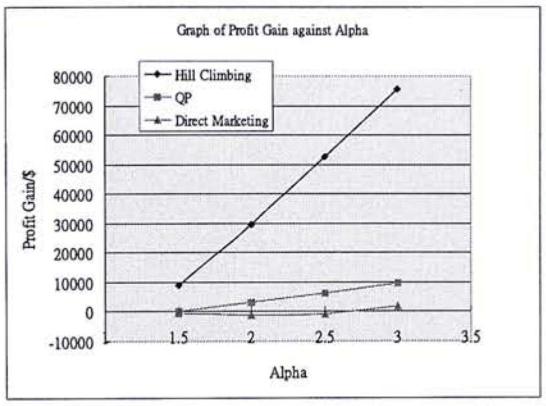

Figure 3.4: Graph of Profit Gains against α for Synthetic Data Set 1 (Free)

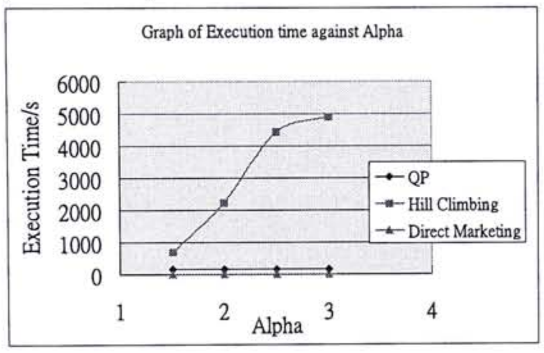

Figure 3.3: Graph of Execution Time against α for Synthetic Data Set 1 (Discount)

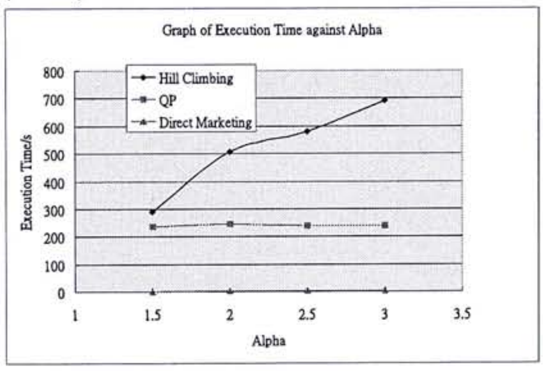

Figure 3.5: Graph of Execution Time against α for Synthetic Data Set 1 (Free)

So, it is better not to apply direct marketing. As the marketing items are free, direct marketing may choose those non-boosting items, which yields to not much profitable items. This leads to a negative profit gain.

3.8 Conclusion

In this thesis, we have formulated formally the problem Item Selection for Marketing (ISM) with cross-selling effect. Besides, we proved that the simple version of this problem is NP-hard. We also proposed two algorithms to deal with this prob-

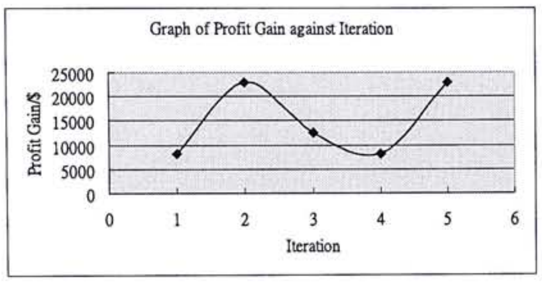

Figure 3.6: Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 1.5$

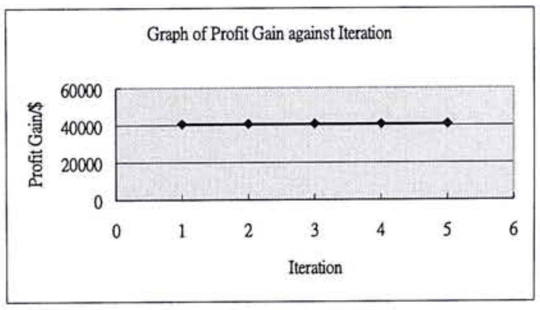

Figure 3.8: Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 2.5$

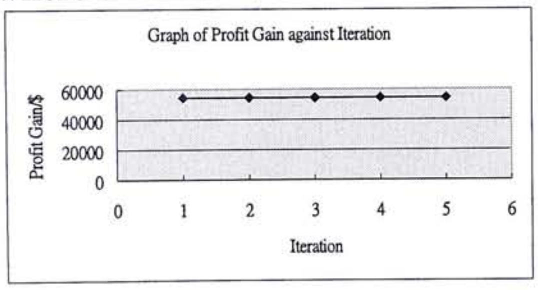

Figure 3.7: Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 2$

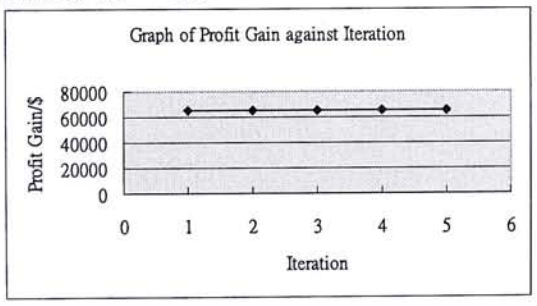

Figure 3.9: Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Discount), where $\alpha = 3$

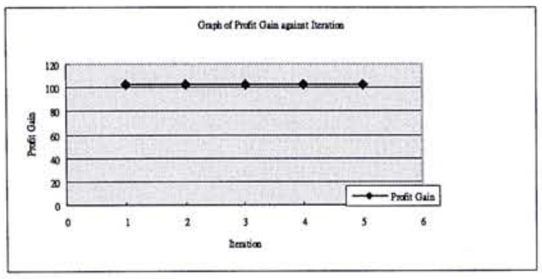

Figure 3.10: Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 1.5$

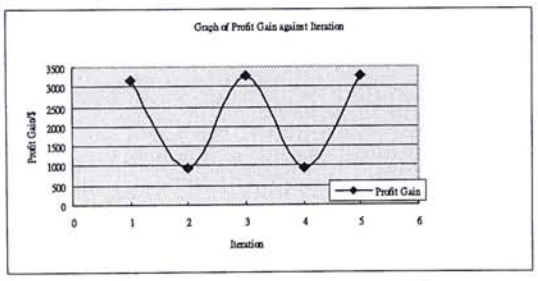

Figure 3.11: Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 2$

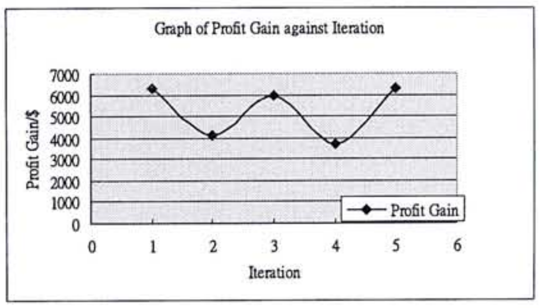

Figure 3.12: Graph of Profit Gains against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 2.5$

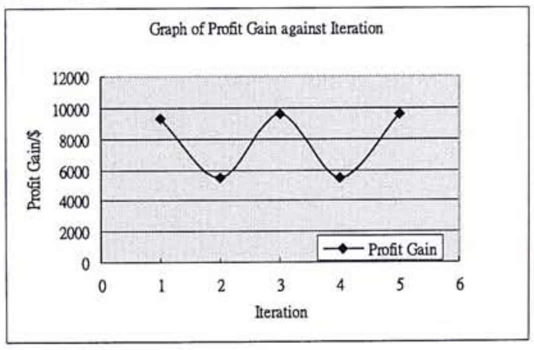

Figure 3.13: Graph of Execution Time against No. of Iterations in QP for Synthetic Data Set 1 (Free), where $\alpha = 3$

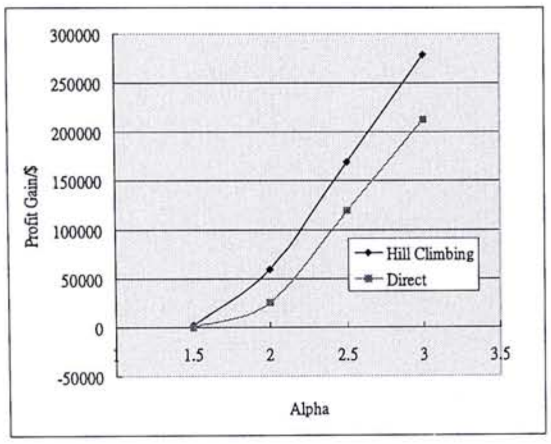

Figure 3.14: Graph of Profit Gains against α for Synthetic Data Set 2 (Discount)

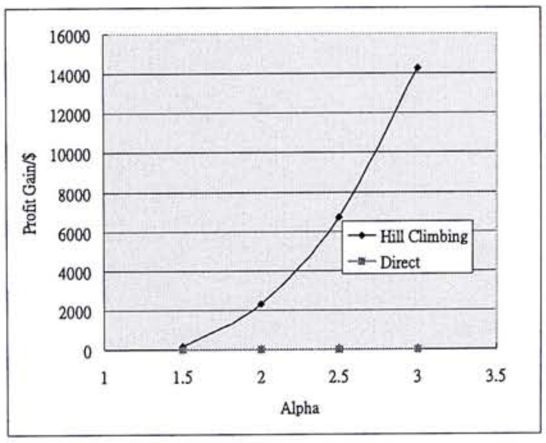

Figure 3.16: Graph of Profit Gains against α for Synthetic Data Set 2 (Free)

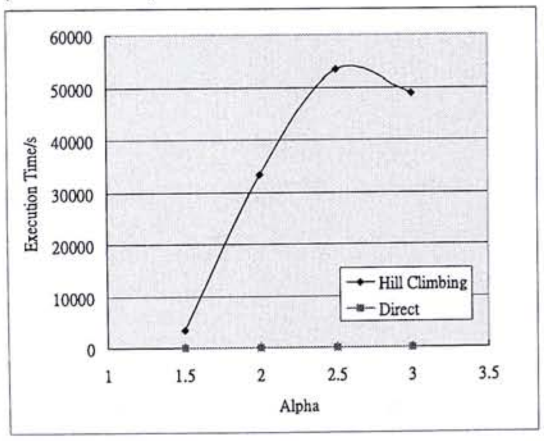

Figure 3.15: Graph of Execution Time against α for Synthetic Data Set 2 (Discount)

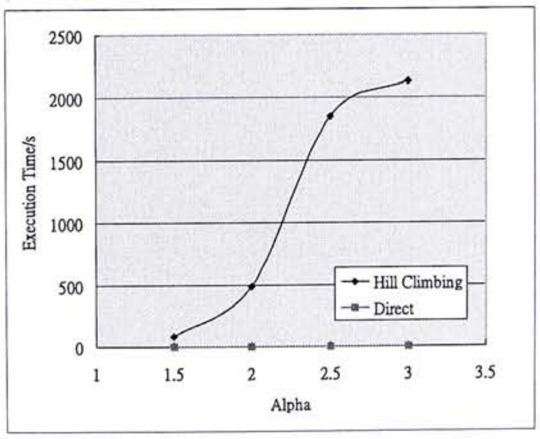

Figure 3.17: Graph of Execution Time against α for Synthetic Data Set 2 (Free)

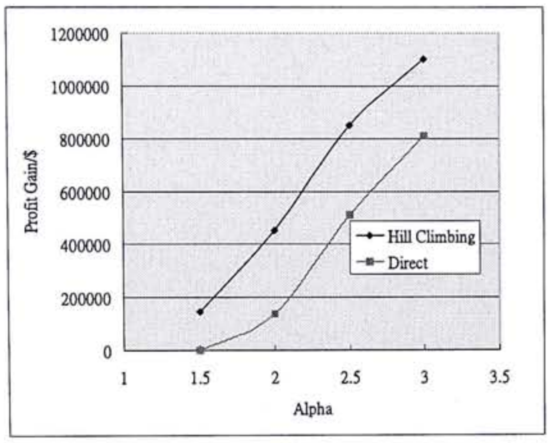

Figure 3.18: Graph of Profit Gains against α for Real Data Set (Discount)

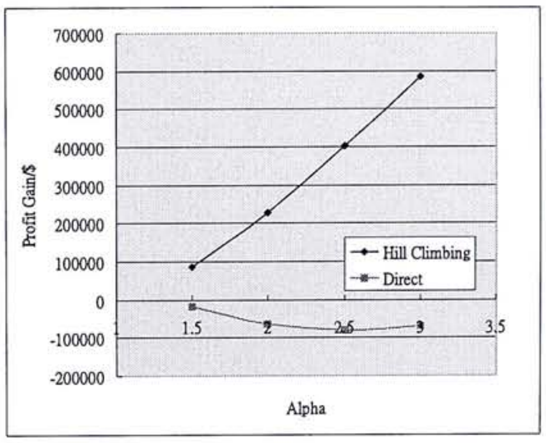

Figure 3.20: Graph of Profit Gains against α for Real Data Set (Free)

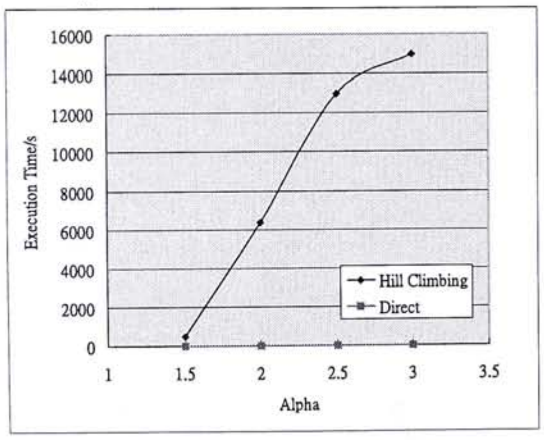

Figure 3.19: Graph of Execution Time against α for Real Data Set (Discount)

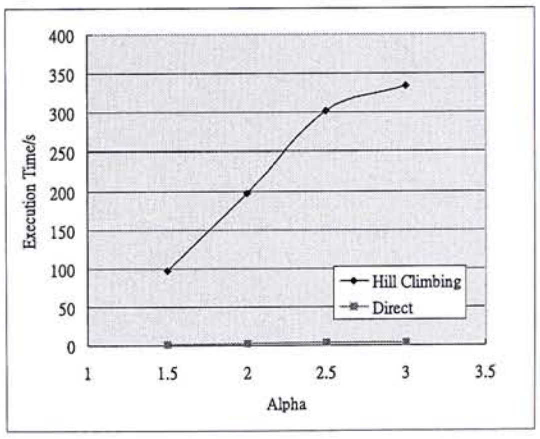

Figure 3.21: Graph of Execution Time against α for Real Data Set (Free)

lem. We also conducted the experiments. The results show that our algorithms are effective and efficient.

Chapter 4

Conclusion

In this thesis, two applications of the concepts of association rules are addressed. Each problem has its own purpose. The first one is MPIS (Maximal-Profit Item Selection with cross-selling considerations) while the second one is ISM (Item Selection for Marketing with Cross-Selling Effect). MPIS is the problem of discarding losing items out of the stock. ISM is the problem of choosing marketing items for some marketing strategy in order to boost the sales of stock. For each of the two problems, we prove that a simple version of the problem is NP-hard. Besides, we propose two modelings of the cross-selling effect among items. We propose the loss rule in MPIS and the gain rule in ISM. For these two problems, we also propose some methods to tackle the problems. Quadratic programming (QP) approach and some heuristics approaches are proposed in MPIS and ISM. In MPIS, we additionally propose an evolutionary approach to tackle the problem. We also conducted experiments to show the effectiveness and efficiency of our proposed algorithms.

Retailers would like to apply more user-friendly tools in order to help their business. There is a company in Hong Kong called Lifewood installing MPIS_Alg algorithm into their products in order to test its feasibility. We hope that the power of the MPIS_Alg algorithm can be utilized and help a lot of enterprises in their business so as to meet customer requirements and earn more profit in the future. We hope that ISM can also be applied in the enterpises for the sake of aiding them in choosing marketing items.

Bibliography

[1] Baron, http://archimedes.scs.uiuc.edu/baron/baron.html.

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

[3] Gams, http://www.gams.com/.

[4] Lindo, http://www.lindo.com/.

[5] Mosek, http://www.mosek.com/.

[6] Opbdp, http://www.mpi-sb.mpg.de/units/ag2/software/opbdp/.

[7] Oprrisk fortqp, http://www.osp-craft.com/.

[8] Tomlab, http://tomlab.biz/.

[9] Wsat(oip), http://www.ps.uni-sb.de/~walser/wsatpb/wsatpb.html.

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

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

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

[13] F. Bass. A new product growth model for consumer durables. In *Management Science*, 1969.

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

[15] A. Bemporad and D. Mignone. http://control.ee.ethz.ch/~hybrid/miqp/.

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

[17] 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.

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

[19] D. J. Cavicchio. Adaptive search using simulated evolution. In *Ph. D. dissertation, Univ. Michigan, Ann Arbor, MI*, 1970.

[20] S. Chapra and R. P. Canale. Numerial methods for engineers. In *McGraw Hill (Third Edition)*, 1998.

[21] P. Domingos and M. Richardson. Mining the network value of customers. In *SIGKDD*, 2001.

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

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

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

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

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

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

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

[29] L. Iasemidis, P. Pardalos, J. Sackellares, and D. 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.

[30] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, 2003.

[31] J. Kleinberg. Authoritative sources in a hyperlink environment. In *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

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

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

[34] R. Kohavi, C. Brodley, L. M. B. Frasca, and Z. Zheng. Kdd-cup 2000 organizers' report: Peeling the onion. In *SIGKDD Explorations 2000*, 20010.

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

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

[37] V. Mahajan, E. Muller, and F. Bass. New product diffusion models in marketing: A review and directions for research. In *Journal of Marketing*, 54:1(1990).

[38] S. W. Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving from Nature, 2*, 1992.

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

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

[41] M. A. Potter, K. A. D. Jong, and J. J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In *Sixth International Conference on Genetic Algorithms*, 1995.

[42] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *SIGKDD*, 2002.

[43] 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.

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

[45] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings Thrid International Conference Genetic Algorithms*, 1989.

[46] B. Taylor. Chapter 16: Inventory management. In *Introduction to Management Science, 7th Edition.* Prentice Hall, 2001.

[47] D. E. Tronrud. Methods of minimization and their implications. In *CCP4 Daresbury Study Weekend, nos. DL/SCI/R35, ISSN 0144-5677. Warrington WA4 4AD, UK: Daresbury Laboratory, for Daresbury Laboratory*, 1994.

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

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

[50] S. Wasserman and K. Faust. Social network analysis. In *Cambridge University Press*, 1994.

[51] R.-W. Wong and A.-C. Fu. Ism: Item selection for marketing with cross-selling considerations. In *PAKDD*, 2004.

[52] R.-W. Wong, A.-C. Fu, and K.Wang. Mpis: Maximal-profit item selection with cross-selling considerations. In *ICDM*, 2003.