5. Experiments

In this section, we evaluate the performance of our algorithm on two cross selling examples with synthetic data which is produced with real patterns. One is a whole selling example, and the other is a financial example. The proposed algorithm is implemented with Borland Jbuilder 7.0 and tested on a PC with AMD Athlon 800K processor and 384 MB main memory under the Windows XP operation system.

5.1. A whole selling example

To illustrate how our whole method works, we make up a scenario first, and then generate the synthetic data to simulate the transaction database in our scenario. At last, we present the experimental result.

5.1.1. Experiment scenario

Suppose a whole selling has four branches in Taipei, and maintains a web to conduct a B2C e-business on Internet. For making promotion strategies for customers who consumes through different cannel in different season, we choose some related attributes and form a multi-dimension transaction database **MD** shown in Fig. 5.1.

T_ID	Date	Channel	Sex	Transaction content
001	05/03/01	Branch01	F	1, 3, 55
002	05/03/01	Branch01	M	56, 777
003	05/03/01	Branch01	M	23, 84
004	05/03/01	Branch01	F	111, 235, 96
. .	. .	. .	. .	. .

Figure 5.1. MD of scenario

There are three dimensions in **MD**, we build three concept hierarchies for each dimension according to user demand. The concept hierarchies are shown in Fig. 5.2.

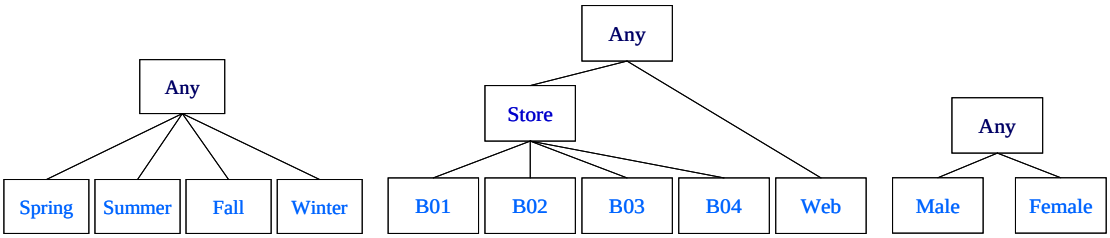

Figure 5.2. Concept hierarchies of scenario

There are 105 multi-dimension patterns generated from $\mathbf{CH_1}$, $\mathbf{CH_2}$ and $\mathbf{CH_3}$, 40 of them are element patterns and the other 65 of them are generalized ones. Our algorithm will discover all large itemsets for the 65 generalized patterns.

5.1.2. Data generation

We modify the transaction data generator proposed in [23] to generate our synthetic data with various characteristics. To decide the number of transactions in each element segmentation, we assign a weight of size WS_{Ei} to each element segmentation first. The WS_{Ei} of am element segmentation T[Ei] is generated by a uniform distribution between 0 and S_M. $|D|$ is the number of transactions in whole transaction database, Suppose there are n element segmentations, the number of transactions in element segmentation $T[E_i]$ is:

$$\left|D_{Ei}\right| = \frac{\left|D\right|}{\sum_{a=0}^{n} WS_{Ea}} WS_{Ei}$$

According to the data generation method proposed in [23], we first determine the size of the next transaction. The size is picked from a Poisson distribution with mean μ equal to $|T|$. We then assign items to the transaction. Each transaction is assigned a series of potentially large itemsets. If the large itemset on hand does not fit in the transaction, the itemset is put in the transaction anyway in half the cases, and the itemset is moved to the next transaction the rest the cases. The number of maximal

potentially large itemsets is set to $|L|$. A maximal potentially large itemsets is generated by first picking the size of the itemset from a Poisson distribution with mean μ equal to $|I|$. More details about the data generator are referred to [23].

To examine the effect of different customer behavior trends, we generate three types of data sets respectively. The characteristics of these three types of data sets are shown in table1. The parameters and the default values of the data sets are shown in table2.

Table 1. Three types of data set

Type1	Fist generate a single set of maximal potentially large itemsets, and then generate transactions for each element pattern E_i from the same maximal potentially large itemsets following the exact method in [23].
Type2	Beside generate a set of common maximal potentially large itemsets, generate maximal potentially large itemsets for each element pattern E_i. Then generate transactions for each element pattern E_i from its own maximal potentially large itemsets and the common maximal potentially large itemsets respectively following the exact method in [23].
Type3	Fist generate a set of maximal potentially large itemsets for each element pattern E_i, and then generate transactions for each element pattern E_i from its own maximal potentially large itemsets following the exact method in [23].

Table 2. Parameters and default values of data sets

Notation	Meaning	Default
$\|D\|$	Number of transactions	100K
$\|T\|$	Avg. size of the transactions	6
$\|I\|$	Avg. size of the maximal potentially large itemsets	4
$\|L\|$	Number of maximal potentially large itemsets	1000
N	Number of items	1000
S_M	The maximum size of segmentation.	50

5.1.3. Experiment result

We first examine the scalability of our algorithm. We generate series of data sets and perform a set of experiments. The default value of minimum support *minsup*, match ratio *m*, and the number of element patterns are 0.5%, 1, and 40 respectively. The experimental results in Fig.5.3 show that our algorithm takes time linear to the number of transactions on all the three types of data set.

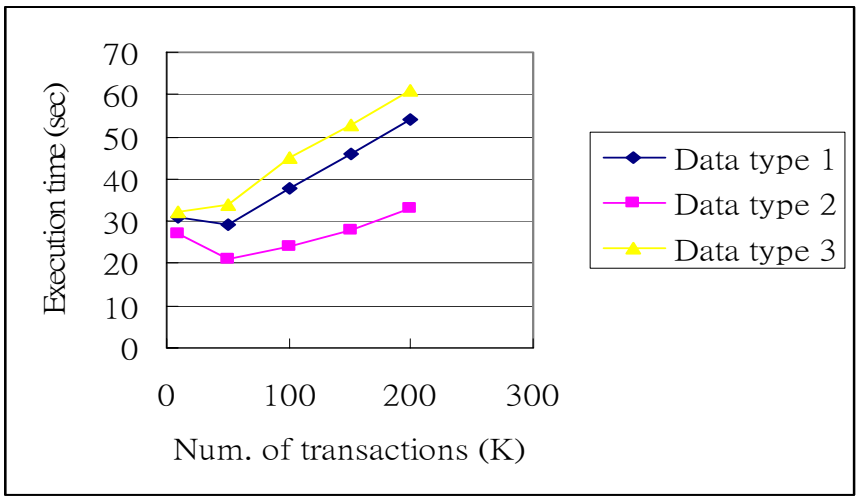

Figure 5.3. Scalability experiment

The next set of experiments is to evaluate the effect of minimum support *minsup* on our algorithm, and the experimental results are shown in Fig. 5.4. All the algorithms are sensitive to the minimum support, the smaller the minimum support, the longer the execution time. However, the real execution time of the update step in our algorithm is relativity much shorter than the whole process.

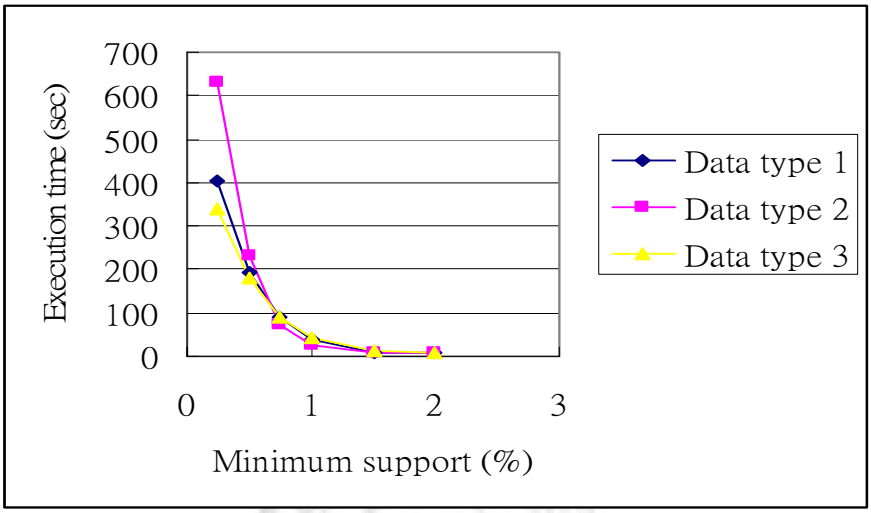

Figure 5.4. Effects of minimum support on efficient.

We last evaluate the effect of number of *element patterns* on our algorithm, and the result is shown in Fig. 5.5. This result shows our algorithm also takes time linear to the number of element patterns on all the three data set types.

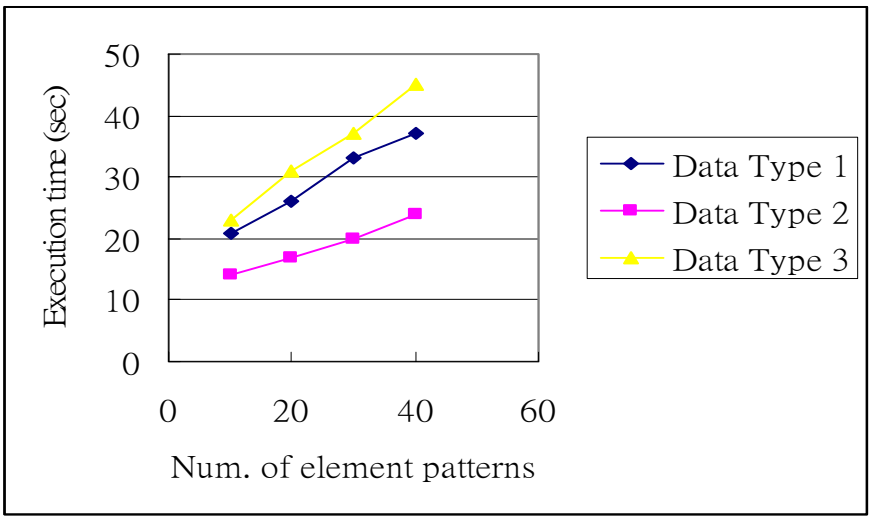

Figure 5.5. Effects of number of element patterns on efficient.

We then evaluate the rules discovered by our method. As discussed above, the objective of our work is discovering association rules which hold in most of meaningful element segmentations belong to their domain, and mining rules which are lost in traditional association rules mining. We define two factors: *discrete ratio* and *lost ratio* to evaluate the rules discovered.

Definition 10.1. *discrete ratio* is the ratio of the number of rules pruned by our algorithm to the number of rules discovered by traditional method.

$$discrete\ ratio = \frac{|\{\ r \mid r \text{ holds in } T[Gj] \cap\ \langle Gj, r\rangle \text{ doesn't hold in } \mathbf{MD}\ \}|}{|\{r \mid r \text{ holds in } T[Gj]\ \}|}$$

Definition 10.2. *lost ratio* is the ratio of the number of rules mined by our algorithm but lost in traditional association rule mining to the number of rules discovered by our algorithm.

$$lost\ ratio = \frac{|\{\langle Gj, r\rangle \mid \langle Gj, r\rangle \text{ holds in } \mathbf{MD} \cap\ r \text{ doesn't hold in } T[Gj]\ \}|}{|\{\langle Gj, r\rangle \mid \langle Gj, r\rangle \text{ holds in } \mathbf{MD}\ \}|}$$

The results of the measurement of *discrete ratio* are shown in Fig. 5.6 and Fig. 5.7. The results show that our algorithm can prune rules which only hold in several element segmentations belong to its domain effectively.

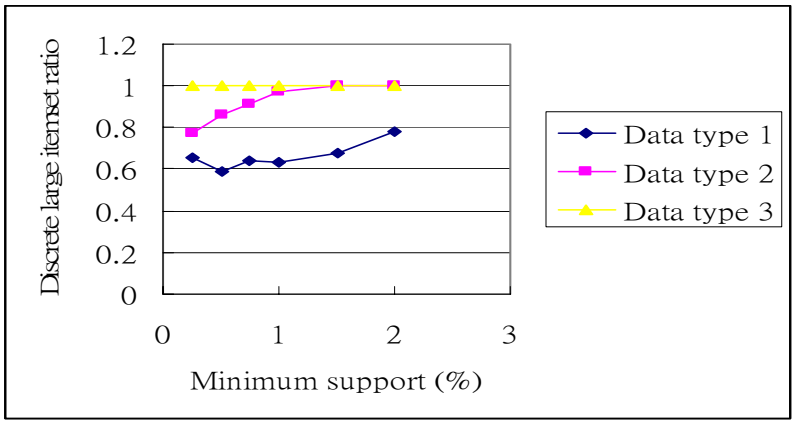

Figure 5.6. Effects of minsup on discrete large itemsets ratio.

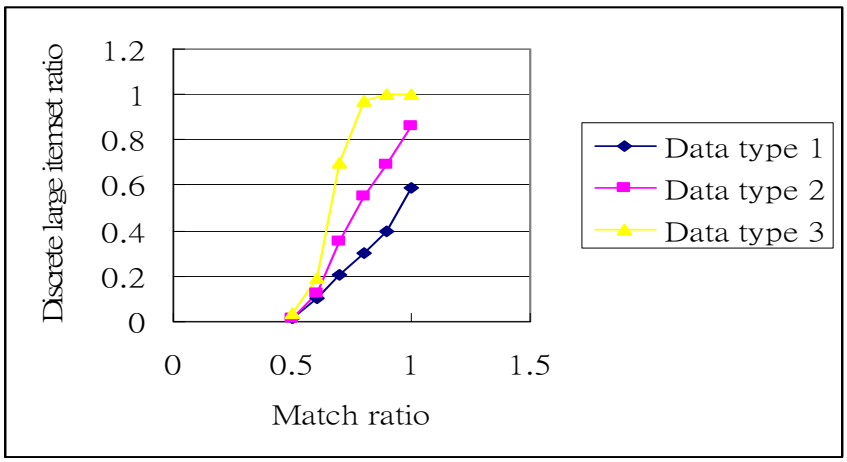

Figure 5.7. Effects of match ratio on discrete large itemsets ratio.

The results of the measurement of *discrete ratio* are shown in Fig. 5.8. According to the result, our algorithm has the ability to discover rules which only hold in parts of the database.

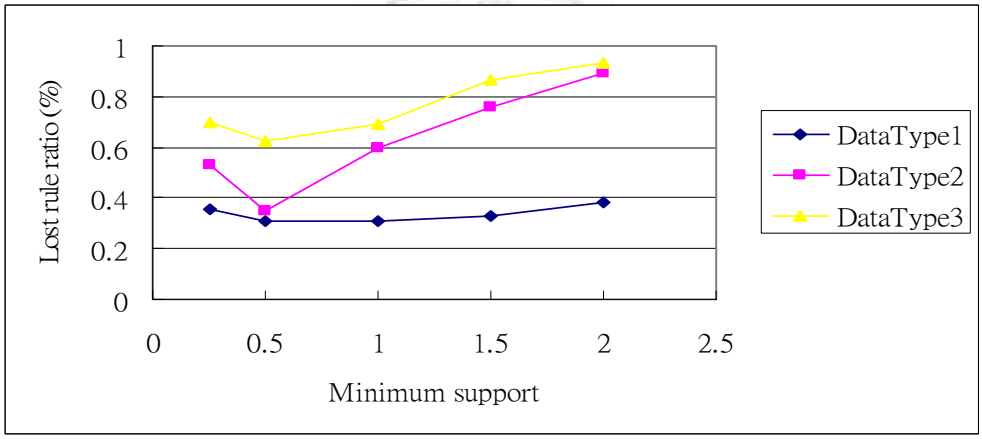

Figure 5.8. Effects of match ratio on lost large itemsets.

5.2. A financial example

5.2.1. Experiment scenario

As a perspective of a bank, time deposit customers produce low value but lead to interest pay out. If we can promote fund to these customers and induce them buy funds with their time deposit interest, they will make more benefit to the bank. The two major types of fund in Taiwan are equity fund and bond fund. Balanced fund, which buys a combination of common stock, preferred stock, bonds, and short-term bonds, has a rapidly growth market. Actually, the market scale of balanced fund is still much smaller than the above two major ones. So, we focus on only the two major two types in this paper. The description of above types of funds is shown in table3.

Table 3. Definition of funds

Equity Fund	A mutual fund also called a stock fund, the value of this mutual fund depends on the value of stocks within a fund's portfolio.
Balanced Fund	A mutual fund that buys a combination of common stock, preferred stock, bonds, and short-term bonds, to provide both income and capital appreciation while avoiding excessive risk.
Bond Fund	A mutual fund invests in different bond issues. A bond fund may invest in one particular type of bond such as corporate or US Government, or in all different types of bonds. Common objectives of bond funds are stability and income.

5.2.2. Data generation

There are a lot of attributes in a customer database, we choice four of them (abode, sex, marry and age) which may influence the type of fund bought by a customer to be the dimensions of our algorithm. Then, we build a concept hierarchy for each dimension as fig. 5.9.

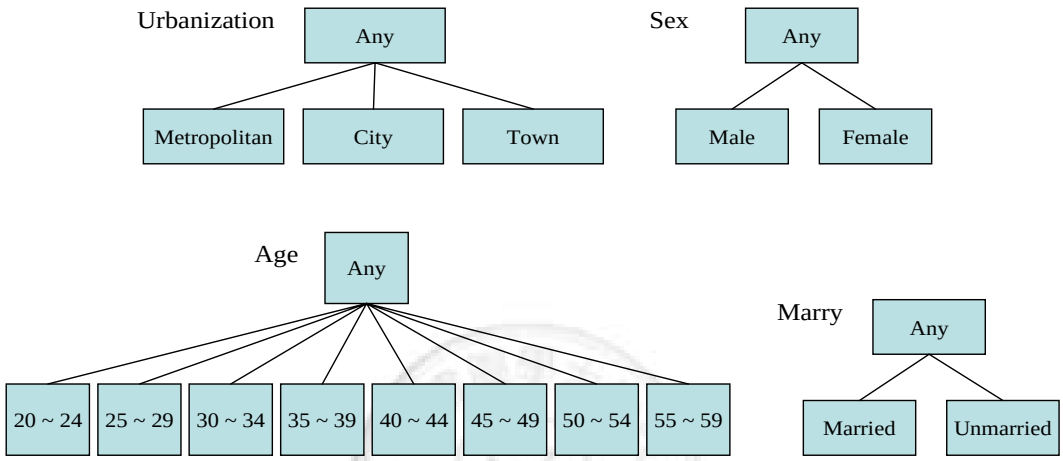

Figure 5.9. Concept hierarchies in the example.

5.2.3. Experiment result

We first examine the scalability of our algorithm again. The result is shown in figure. 5.10. The experimental result shows that our algorithm takes time linear to the number of transactions on this example.

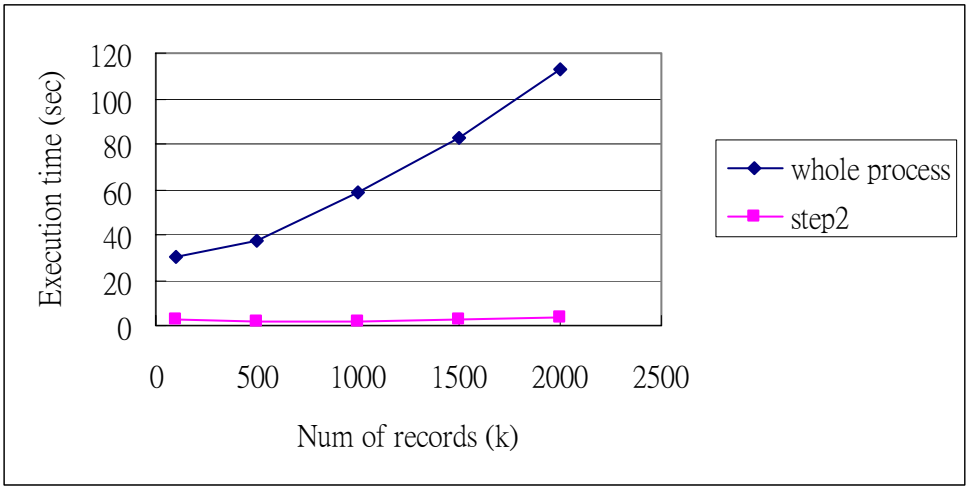

Figure 5.10. Scalability experiment

We found some associations between their profiles and the type of funds they bought. Some of the experiment results are shown below:

1. 30% time deposit customers also bought funds, most of them bought bond funds.
2. Certain percent of time deposit customers who are unmarried and whose age are 20-29 bought equity funds.
3. Certain percent of female customers whose age are 20-29 bought equity funds. On the other hand, male customers whose age are 20-39 have more interest in equity funds.
4. The type of funds be bought have slight different between customers who live in a metropolitan and customers who live in a town, but the number of customers who live in a town but bought funds is much less than customers who live in a metropolitan and bought funds.