Methods for Mining Cross Level Association Rule In Taxonomy Data Structures

V. Venkata Ramana[1]
Asst. Prof., CSE Dept.,
SSITS, Rayachoty,
Kadapa (dist), A.P. INDIA.

M V Rathnamma[2]
Asst.Prof., CSE Dept.,
SSITS, Rayachoty.
Kadapa (dist), A.P. INDIA.

A. Rama Mohan Reddy[3]
Professors, Dept. Of CSE,
SVU College of Engineering,
Tirupati, A.P., INDIA.

ABSTRACT
Mining of association rules mainly focuses at a single conceptual level. In a large database of transaction, where each transaction consists of a set of items, and taxonomy on items, it is required to find out the associations at multiple conceptual levels. In this paper, multilevel association rule mining algorithms have been evaluated and compared. And we will discover additional strong association rules in taxonomy data items. The performance indices used for performance comparisons are minimum support threshold at different levels and varying number of transactions.

***Index Terms*:**

Association rules, Multilevel Association Rules, Cross Level Association **R**ules

1. INTRODUCTION
Discovery of interesting association relationships among huge amounts of data will help marketing, decision making, and business management. Therefore, mining association rules from large data sets has been a focused topic in recent research into knowledge discovery in databases or Data mining.

Data mining refers to extracting or "Mining" knowledge from large amount of data. Association rule mining finds interesting association among a large set of data items.

1.1 Synthetic database:

Synthetic data generation for multilevel association rule mining requires generation of the *synthetic transaction databases* as a first step. The second step involves generation of the *item description tables* to build taxonomy over the items. Synthetic transaction databases are generated using a randomized item sets generation algorithm [2]. The following are the basic parameters [1] of the generated databases.

1. **D** No. of transaction.
2. **T** Average size of the transactions.
3. **I** Average size of the maximal potentially large item sets.
4. **L** maximal potentially large item sets.
5. **N** Number of Items.

To create a dataset, the size of the next transaction is determined first. The size is picked from a Poisson distribution with mean p equal to T. If each item is chosen with the same probability p, and there are N items, the expected number of items in a transaction is given by a binomial distribution with parameters N and p, and is approximated by a Poisson distribution with mean Np.

Then items to the transaction are assigned. Each transaction is assigned a series of potentially large item sets. If the large item set on hand does not fit in the transaction, the item set is put in the transaction anyway in half the cases, and the item set is moved to the next transaction in the rest of the cases. ref[7]

Large item sets are chosen from a set Γ of such item sets. The number of item sets in Γ is set to L. There is an inverse relationship between L and the average support for potentially large item sets. An item set in Γ is generated by first picking the size of the item set from a Poisson distribution with mean μ equal to I. Items in the first item set are chosen randomly. To model the phenomenon that large item sets often have common items, some fraction of items in subsequent item sets are chosen from the previous item set generated. An exponentially distributed random variable with mean equal to the correlation level is used to decide this fraction for each ref[2]

Each item set in Γ has a weight associated with it, which corresponds to the probability that this item set will be picked. This weight is picked from an exponential distribution with unit mean, and is then normalized so that the sum of the weights for all the item sets in Γ is 1. The next item set to be put in the transaction is chosen from Γ by tossing an L-sided weighted coin, where the weight for a side is the probability of picking the associated item

The discovery of interesting association relationship among huge amount of business transaction records can help in business decision making processes like catalogue design, cross marketing, and loss leader analysis

Mining of association rules mainly focuses at a single conceptual level. There are applications which need to find associations at multiple conceptual levels. In a large database of transaction, where each transaction consists of a set of items, and a taxonomy (is-a hierarchy) on items, it is required to find out associations between items at any level of taxonomy

To explore multilevel association rule mining [7], one needs to provide Data at multiple-level association at multiple levels of abstraction and efficient methods for multiple level rule mining. The first requirement can be

satisfied by providing concept taxonomies from the primitive level concepts to higher levels.

Second requirement requires efficient methods for multilevel rule mining. The following fig is the example:

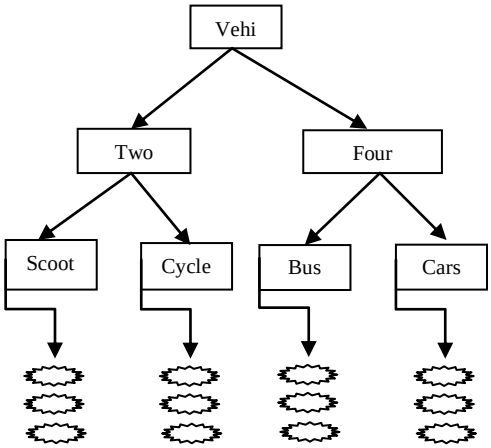

Fig 1: Example for Vehicle

2. Multilevel Association Rules

To discover the mining multilevel of association rules [7] from a large set of transaction data, we assume that the database contains: A transaction data set, T , which consists of a set of transactions $<T_i, \{A_p,..., A_q\}>$, where T_i is a transaction identifier, $A_i \in I$ (for $i = p,...,q$), and I is the set of all the data items in the item data set And The description of the item data set, D, which contains the description of each item in I in the form of $<A_i$, *description*>, where $A_i \in I$.

Definitions related to multilevel association rules are given as follows:

Definition 2.1 An item set, A, is a set of data items $\{A_i, A_j\}$, where A_i, $A_j \in I$. The **support** of an item set A in a set S, $\sigma(A/S)$, is the number of transactions (in S) which contain A versus the total number of transactions in S. The **confidence** of $A \rightarrow B$ in S, $\varphi(A \rightarrow B/S)$, is the ratio of $\sigma(A \cup B / S)$ versus $\sigma(A/S)$, i.e., the probability that item set B occurs in S when item set A occurs in S.

Definition 2.2 An item set A is **large** in set S at level l if the support of A is no less than its corresponding minimum support threshold σ'_l. The confidence of a rule *"A* $\rightarrow$ *B/S"'* is **high** at level l if its confidence is no less than its corresponding minimum confidence threshold φ'_l.

Definition 2.3 A rule "A $\rightarrow$ B/S" is **strong** if "A ∪ B/S" is B/S" is large at the current level and the confidence of "A $\rightarrow$ B/S" is high at the current level.

3. Multilevel Association Rule Mining

Algorithms

A method [7] for mining multilevel association uses a taxonomy information encoded transaction table. The taxonomy information presented in of Figure 1. is encoded as a sequence of digits in the transaction table *T[1]* (Table 1).

TID	Items
T_1	{111, 121, 211, 221}
T_2	{111, 211, 222, 323}
T_3	{112, 122, 221, 411}
T_4	{111, 121}
T_5	{111, 122, 211, 221, 413}
T_6	{211, 323, 524}
T_7	{323, 411, 524, 713}

Table 1. Encoded Transaction Table: *T[1]*

For example, the item *'2% Foremost milk'* is encoded as *'112'* in which the first digit, *'1'*, represents *'milk'* at level-1, the second, *'1'*, for *'2% (milk)'* at level-2, and the third, *'2'*, for the brand *'Foremost'* at level-3.

The derivation of the large itemsets at level 1 proceeds as follows. Let the minimum support at level 1 be 4 transactions (i.e., minsup[1] = 4). Notice that since the total number of transactions is fixed, the support is expressed in an absolute value rather than a relative percentage, for simplicity. The level-1 large 1-itemset table *L[1,1*] can be derived by scanning *T[1]*, registering support of each generalized item, such as *1***, ..., *4***, if a transaction contains such an item is smaller than the minimum support, which results in *L[2,1]* of Figure 3.3.

Level-1 minsup = 4

Level-1 large 1-itemsets:

L[1,1]

Itemset	*Support*
*{1**}*	*5*
*{2**}*	*5*

Filtered transaction table:

T[2]

TID	Items
T_1	*{111, 121, 211, 221}*
T_2	*{111, 211, 222}*
T_3	*{112, 122, 221}*
T_4	*{111,121}*
T_5	*{111, 122, 211, 221}*
T_6	*{211}*

Level-1 large 2-itemsets:

L[1,2]

Itemset	*Support*
*{1**, 2**}*	*4*

Fig 2: Large item sets at level 1 and filtered transaction table *T[2]*.

Similarly, the large 2-itemset table *L[2 2*] is formed by the combinations of the count and removing those whose support entries in *L[2,1]*, Together with the support derived from *T[2]*, filtered using the corresponding threshold level-

2 1-itemsets L[2,1] can be derived from the filtered transaction

and filtering out those whose accumulated support count is lower than the minimum support. *L[1, 1]* is then used to filter out (1) any item which is not large in a transaction, and (2) the transactions in *T[1]* that contain only small items. This results in the filtered transaction table *T[2]* of Figure 3.2. Moreover, since there are only two entries in *L[1,1]*, the level-1 large-2 itemset table *L[1,2*] may contain only 1 candidate item *{1**, 2**}*, which is supported by 4 transactions in *T[2]*.

According to the definition of multilevel association rules (ML-association rules), only the descendants of the large items at level-1 (i.e., in *L[1,1])* are considered as candidates for the level-2 large table *T[2]* by accumulating the support.

Likewise, the large 3-itemset table *L[2,3]* is formed by the combinations of the entries in *L[2,2]*. Finally, *L[3,1]* and *L[3,2]* at level 3 are computed similarly, with the results shown in Figure 3.3. The computation terminates since there is no deeper level in the hierarchy. The derivation also terminates when an empty large 1-itemset table is generated at any level [7].

The above discussion leads to the following algorithms for mining strong ML-association rules.

1. **D** No. of transaction.
2. **T** Average size of the transactions.
3. **l** Average size of the maximal potentially large itemsets.
4. **L** maximal potentially large itemsets.
5. **N** Number of Items.

3.1 Algorithm ML_T2L1

Algorithm ML_T2L1 [7] finds multilevel large itemsets for mining strong ML-association rules in a transaction database.

Input:
T[1], a hierarchy-information-encoded and task-relevant set of a transaction database, in the format

Output: Multiple level large itemsets.

Method: A top down, progressively deepening process which collects large itemsets at different conceptual levels as follows.

Starting at level 1, derive for each level *l*, the large k-items sets, *L[l, k]*, for each k, and the set of large itemsets, *LL[l]* (for all k's).

1. *for (l := 1; L[l, 1] != ∅ and l < max level; l++) do {*
2. *if l = 1 then {*
3. *L[l, 1] := get_large_1_itemsets(T[1], l);*
4. *T[2] := get_filtered_t_table(T[1], L[1, 1]);*
5. *}*
6. *6. else L[l, 1] := get_large_1_itemsets(T[2], l);*
7. *L[l, k] := {c ∈ C_k | c.support ≥ minsup[l]}*
8. *. }*
9. *15. LL[l] := ⋃$_k$ L[l, k];*
10. *16. }*

3.2 Algorithm ML_T1LA

This algorithm [7] uses only one encoded transaction table *T[1]*, that is, no filtered encoded transaction table *T[2]* will be generated in the processing. At the first scan of *T[1]*, large 1-itemsets *L[l, 1]* for every level *l* can be generated in parallel, because the scan of an item i in each transaction t may increase the count of the item in every *L[l, 1]* if its has not been incremented by t. After the scanning of *T[1]*, each item in *L[l, 1]* whose parent (if *l > 1*) is not a large item in the higher level large 1-itemsets or whose support is lower than minsup[*l*] will be removed from *L[l, 1]*.

After the generation of large 1-itemsets for each level *l*, the candidate set for large 2-itemsets for each level l can be generated by the apriori-gen algorithm[4]. The get subsets function will be processed against the candidate sets at all the levels at the same time by scanning *T[1]* once, which calculates the support for each candidate itemset and generates large 2-itemsets *L[l, 2]*. Similar processes can be processed for step-by-step generation of large k-item-sets *L[l, k]* for *k > 2*.

This algorithm avoids the generation of a new encoded transaction table. Moreover, it needs to scan *T[1]* once for generation of each large k-itemset table. Since the total number of scanning of *T[1]* will be k times for the largest k-itemsets, it is a potentially efficient algorithm. However, *T[1]* may consist of many small items which could be wasteful to be scanned or examined. Also, it needs a large space to keep all *C[l]* which may cause some page swapping.

The algorithm is summarized as follows.

It uses only one encoded transaction table *T[1]*. The input and output specifications are the same as Algorithm ML_T2L1. The procedure is described as follows.

1.{L[1,1],…,L[max_l,1]}:=get_all_large_1_itemsets(T[1]);
2. more_results := true;
3. for (k := 2; more_results; k++) do begin
4. more_results := false;
5. for (l := 1; l < max_l; l++) do
6. if L[l, k] != ∅ then begin
7. C[l] := get_candidate_set(L[l, k-1]);

3.3 Algorithm ML_TML1

This algorithm [7] generates multiple encoded transaction tables *T[1], T[2], …, T[max_l + 1]*, where max_*l* is the maximal level number to be examined in the processing.

Level-2 minsup = 3

Level-2 large 1-itemsets:

L[2,1]

Itemset	*Support*
{11}*	*5*
{12}*	*4*
{21}*	*4*
{22}*	*4*

Level-2 large 2-itemsets:

L[2,2]

Itemset	*Support*
{11, 12*}*	*4*
{11, 21*}*	*3*
{11, 22*}*	*4*
{12, 22*}*	*3*
{21, 22*}*	*3*

Level-2 large 3-itemsets:

L[2,3]

Itemset	*Support*
{11, 12*, 22*}*	*3*
{11, 21*, 22*}*	*3*

Level-3 minsup = 3

Level-3 large 1-itemsets:

L[3,1]

Itemset	*Support*
{111}	*4*
{211}	*4*
{221}	*3*

Level-3 large 2-itemsets:

L[3,2]

Itemset	*Support*
{111, 211}	*3*

Fig 3: Large item sets at level 2 and level 3 and filtered.

This algorithm first scans *T[1]* and generates the large 1-itemsets *L[1,1]* which are used to filter out small items from *T[1]* . *T[2]* results from this filtering process and is used in the generation of large k-itemsets at level 1.

On the other hand, Algorithm ML_T2L1, *T[2]* is not repeatedly used in the processing of the lower levels. Instead, a new table *T[l + 1]* is generated at the processing of each level *l*, for $l > 1$. This is done by scanning *T[l]* to generate the large 1-itemsets *L[l, 1]* which are used to filter out small items from *T[l]*. *T[[l + 1]* results from this filtering process and is used in the generation of large k-itemsets (for $k > l$) at level *l* and table *T [l + 2]* at the next lower level.

The algorithm is summarized as follows.

It uses multiple encoded transaction tables. The input and output specifications are the same as Algorithm ML_T2L1.

1. for (l :=1;L[l, 1]!= Φ *and l < max_level; l ++) do begin*
2. if l = 1 then L[l, 1] := get_large_1_itemsets(T[1], l);
3. {T[l + 1], L[l + 1, 1]}:=get_filtered_T_table and large_1_itemsets(T[l],L[l,1]);
4. for (k := 2; L[l, k - 1] != Φ *; k++) do begin*
5. C_k := get_candidate_set(L[l, k - 1]);
6. for each transaction t $\in$ *T [l + 1] do begin*
7. Ct := get_subsets(C_k, t);
8. for each candidate c $\in$ *Ct do c.support++;*
9. end
10. L[l, k] := {c $\in$ *C_k|c.support – minsup[l]}*
11. end
12. LL[l] := $\cup_k$ *L[l , k];*

3.4 Algorithm ML_T2LA

This algorithm [7] uses the same two encoded transaction tables *T[1]* and *T[2]* as in Algorithm ML_T2L1, but it integrates some optimization techniques considered in the algorithm ML_T1LA.

The scan of *T[1]* first generates large 1-itemsets *L[1 1]*. An additional scan of *T[1]* using *L[1,1]* will generate a filtered transaction table *T[2]* and all the large 1-itemset tables for all the remaining levels, i.e., *L[l, 1]* for $1 \leq \max\_l$ by incrementing the count of every *L[l,1]* at the scan of each transaction and removing small items and the items whose parent is small from *L[l, 1]* at the end of the scan of *T[1]*.

The candidate set for the large 2-itemsets at each level *l* can then be generated by the apriori-gen algorithm [1], and the get subsets routine will extract the candidate sets for all the level *l* ($l \geq 1$) at the same time by scanning *T[2]* once. This will calculate the support for each candidate itemset and generate large 2-item-sets *L[l,2]* for $l \geq 1$.

Similar processes proceed step-by-step which generates large k-item-sets *L[l,k]* for $k>2$ using the same *T[2]*. This algorithm avoids the generation of a group of new filtered transaction tables. It scans *T[1]* twice to generate *T[2]* and the large 1-itemset tables for all the levels. It then scans *T[2]* once for the generation of each large k-itemset, and thus scans *T[2]* in total k-1 times for the generation of all the k-itemsets, where k is the largest such k-itemsets available. Since k-itemsets generation for $k > 1$ is performed on *T[2]* which may consist of much less items than *T[1]*, the algorithm could be a potentially efficient one.

The algorithm is summarized as follows.

It uses two encoded transaction tables. The input and output specifications are the same as Algorithm ML_T2L1. The procedure is described as follows.

1. L[1, 1] := get_large_1_itemsets(T[1], 1);
2 .{T[2],L[2,1],…,L[max_l,1]}:=get_filtered_t_table and_l large_1_itemsets(T[1], L[1,1]);
3. more_results := true;
4. for (k := 2; more_results; k++) do begin
5. more_results := false;
6. for (l := 1; l < max_l; l++) do
7. if L[l, k -1] != $\varnothing$ *then begin*

4. Cross Level Association Rules

The multilevel association rule mining algorithms generate association rules which are confined to level-by-level relation in a hierarchy. But there may be strong association rules among the concepts at "*cross-level*" of a hierarchy. For example, *"2% foremost milk → Wonder bread"* in which the two concepts are at different levels of a hierarchy shown in Figure 2.

This can be achieved by making modifications to multilevel association rule mining algorithms. The cross-level association rule mining require the itemsets like <{112, 2∗1}>. Let minimum support at each level be: minsup =4 at level-1, and minsup = 3 at levels 2 and 3. The derivation of the large itemsets at level 1 proceeds in the same way as in algorithm ML_T2L1. Which generates the large itemsets tables *L[1, 1]* and *L[1, 2]* at level *1*, and the same filtered transaction table *T[2]*, as shown in Figure 3.1.

Level-3 minsup = 3

Level-3 large 1-itemset:
L[3,1]

Itemset	*Support*
{111}	*4*
{211}	*4*
{221}	*3*

Level-3 large 2-itemset:
L[3,2]

Itemset	*Support*
{111, 211}	*3*
{111, 21∗}	*3*
{111, 22∗}	*3*
{111, 2∗∗}	*4*
{11∗, 22∗}	*3*
{1∗∗, 2∗∗}	*3*

Level-3 large 3-itemset:
L[3,3]

Itemset	*Support*
{111, 21∗, 22∗}	*3*

Fig 3.1: Cross-level large itemsets at level 3.

Itemset	*Support*
{11∗, 12∗}	*4*
{11∗, 21∗}	*3*
{11∗, 22∗}	*4*
{12∗, 22∗}	*3*
{21∗, 22∗}	*3*
{11∗, 2∗∗}	*4*
{12∗, 2∗∗}	*3*
{21∗, 1∗∗}	*3*
{22∗, 1∗∗}	*4*

Level-2 minsup = 3

Level-2 large 1-itemset:
L[2,1]

Itemset	*Support*
{11∗}	*5*
{12∗}	*4*
{21∗}	*4*
{22∗}	*4*

Level-2 large 3-itemset:
L[2,3]

Itemset	*Support*
{11∗, 12∗, 22∗}	*3*
{21∗, 22∗, 1∗∗}	*3*

Fig 3.2: Cross-level large itemsets at level 2.

This can be computed, with the results shown in Figure 3.4. The entries which pair with their own ancestors are not included since it is contained implicitly in their corresponding 2-itemsets. For example, *<{11*, 12*}, 4>* in *L[2,2]* implies *<{11∗, 12∗,1∗∗,} 4>* in *L[2,3]*. Finally, the large 1-itemset table at level 3, *L[3,1]*, should be the same as Figure 3.3. The large 2-itemset table includes more itemsets since these items can be paired with higher level large items, which leads to the large 2-itemsets *L[3, 2]* and large 3-itemsets *L[3, 3]* as shown in Figure 3.2.

Similarly, the itemsets *{111,11*}* and *{111, 1**}* have the same support as *{111}* in *L[3, 1]* and are thus not included in *L[3,2]*. Since the large k-itemset (for $k > 1$) tables do not explicitly include the pairs of items with their own ancestors, since the existence of a specialized item always indicates the existence of an item in that class, such as *"2% milk → milk(100%)"*, such trivial rules should be eliminated.

The below are some of tables and corresponding graphs

Item Table	# node at level-1	M2	M2
I1	10	10	10
I2	20	8	5

Database	S	T
DB1	4	100,000
DB2	6	1000,00

Table 2: Parameters setting of the item description (hierarchy) tables

Two database settings are used, DB1, with average size (the number of frequent items) of potentially frequent itemsets of 4 and average transaction size (the number of items) of 10 and DB2, with average size of potentially frequent itemsets of 6 and average transaction size of 20.

Two item tables are used in the testing: the first one, I1, has 10, 10, and 10 branches at the levels 1, 2, and 3 respectively; whereas the second, I2, has 20, 8, and 5 branches at corresponding levels.

5. PERFORMANCE COMPARISON

The testing results presented in this section are on the two synthetic transaction databases: DB1I1, which uses the database setting DB1 and the item description table I1, and DB2I2, which uses the databases setting DB2 and the item description table I2.

5.1 Minimum Support Threshold at Level 1

Figure 5.1(a) shows the running time of the four algorithms on DB1I1 respect to the minimum support

threshold at level 1.

The minimum supports at level 2 and 3 are fixed to 2 percent and 0.75 percent. The level 2 minimum support threshold is set to 2 percent which means no filtering of items in transactions at level 2. Therefore, T[3] has the same size of T[2] and the derivation of T[3] is a waste.

The four curves in Figure 5.1(a) show that ML_T2LA has the best performance, while the ML_T1LA has the worst among the four algorithms under the threshold setting 70 percent and 60 present at level 1. This is because the first threshold filters out many small 1-itemsets at level 1 which results in a much smaller filtered transaction table T[2]. The filter mechanism at level 2 is not so strong, so parallel derivation of L[*l*, k] without derivation of T[3] and T [4] is more beneficial. These lead ML_T2LA to be the best algorithm. And ML_T1LA is the worst algorithm since it consults a large T[1] at every level.

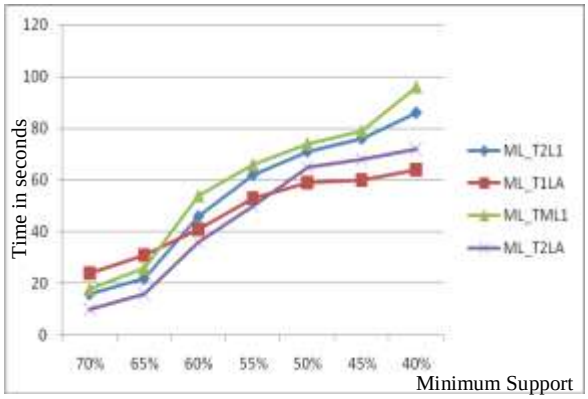

Fig 5.1(a): Minimum support at level 1 for database DB1I1

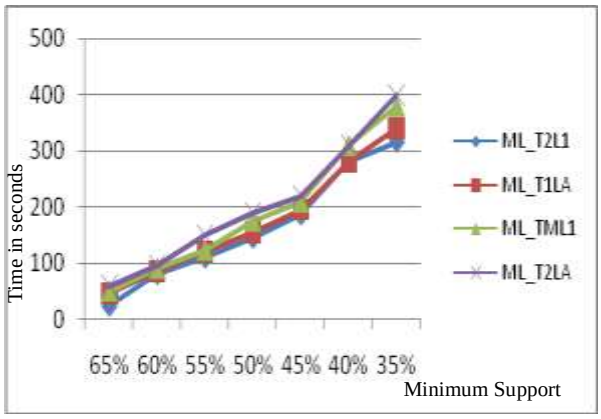

Fig 5.1(b): Minimum Support at level 1 for database DB 2I2

5.2 Minimum Support Threshold at Level 2

Figure 5.2(a) shows the running time for the four algorithms on DB1I1 with respect to the minimum support threshold at level 2. The minimum supports at levels 1 and 3 are fixed to 60 percent and 0.75 percent.

The stronger the filtering mechanism, the more 1-itemsets are filtered out at each level, and the smaller large 1-itemsets are resulted in. Thus ML_TML1, which generates a sequence of filtered transaction tables, has the lowest cost at the minimum support threshold 14 percent and 11 percent but the highest cost at threshold 8 percent, 5 percent and 2 percent since few items are filtered out. On the contrary, ML_T2L1 has highest cost at 14 and 11 percent. This because the first threshold is not big enough and lower level also has not reasonable small threshold. So few small 1-itemsets filtered out at level 1 which results in almost same sized transaction table T[2] and generation of multiple filtered transaction tables are beneficial at lower levels. ML_T2LA performing the best at threshold 8percent, 5percent and 2 percent because the threshold is reasonable small at the lower levels.

Figure 5.2(b) shows the running time of the four algorithms with respect to the minimum support threshold at level 2 but using different database DB2I2. The minimum supports at level 1 and 3 are fixed to 55 percent and 1 percent.

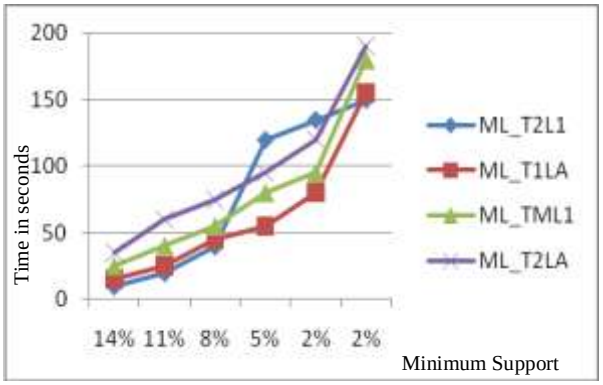

Fig 5.2(a): Minimum support at level 2 for database DB1I1

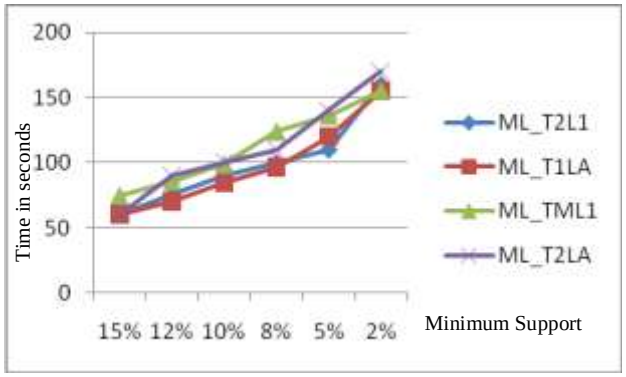

Fig 5.2(b) Minimum Support at level 2 for database DB 1I1

5.3 Minimum Support Threshold at Level 3

Figure 5.3(a) shows the running time of the four algorithms on DB1I1 with respect to the minimum support threshold at level 3. The minimum supports at levels 1 and 2 are fixed to 60 percent and 2 percent. Ref[5]

The four curves in Figure 5.3(a) show that ML_T2LA performing the best because the first threshold filters outs many small 1-itemsets at first level 1 which results in a much smaller filtered transaction table T[2]. The thresholds at lower levels are reasonable small so parallel derivation of L[*l*, k] without derivation of T[3] and T[4] is more beneficial. This leads ML_T2LA to be the best algorithm and ML_T1LA and ML_TML1 are the worst algorithms.

Figure 5.3(b) shows the running time of the four algorithms with respect to the minimum support threshold at level 2 but using different database DB2I2. The minimum supports at level 1 and 3 are fixed to 55 percent and 1 percent.

The above figures show two interesting features. First, the relative performance of the four algorithms is highly relevant to the threshold setting especially the level 1 and level 2 thresholds. Thus, based on the effectiveness of a threshold, a good algorithm can be selected to achieve good performance. Second, the parallel derivation of L[*l*, k] is very useful and the derivation of T[2] is usually beneficial. Results show ML_T2LA is always the best or the second best algorithm

6. Extended Algorithms

The input and output specifications are the same as Algorithm ML_T2L1. The procedure is described as follows. ref[5]

1. for (l := 1; L[l, 1] != ∅ and l < max level; l++) do {
2. if l = 1 then {
3. L[l, 1] := get_large_1_itemsets(T[1], l);
4. T[2] := get_filtered_t_table(T[1], L[1, 1]);
5. }
6. else L[l, 1] := get_large_1_itemsets(T[2], l);
7. for (k := 2; L[l, k - 1] != ∅; k++) do {
L[l, k];
8.}

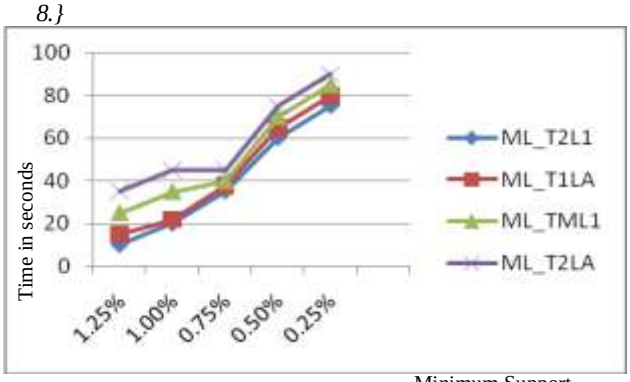

Fig 5.3(a): Minimum support at level 3 for database DB1I1

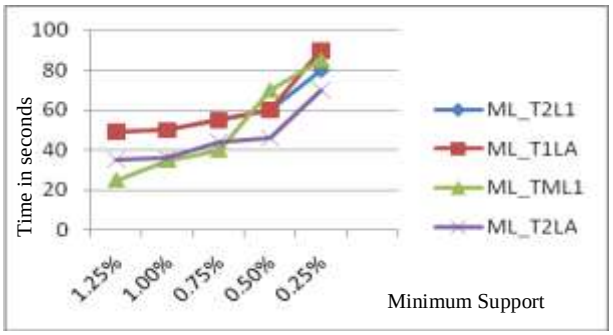

Fig 5.3(b): minimum support at level3 fordatabase DB2I2

Function get_cross_candidate_set at line 8 generates cross level large item sets that combined with candidate item sets in line 9.

Comparison of extended algorithm with ml_t2l1

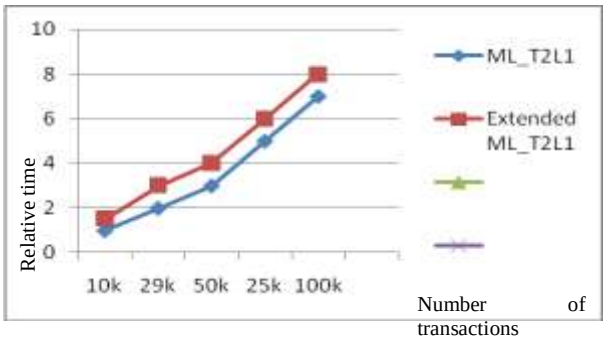

Fig 5.4: Performance with threshold (60, 8, 1)

- Multilevel rules can provide richer information than single level rules, and represents the hierarchical nature of the knowledge discovery process.
- The relative performance of the four algorithms is highly relevant to the threshold setting especially the level 1 and level 2 thresholds but relatively independent of the number of transactions used in the testing. Thus, based on the effectiveness of a threshold, a good algorithm can be selected to achieve good performance.

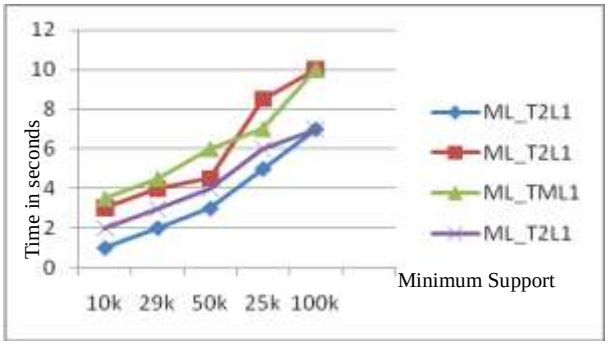

Fig 5.5 Performance with threshold (65, 2,1)

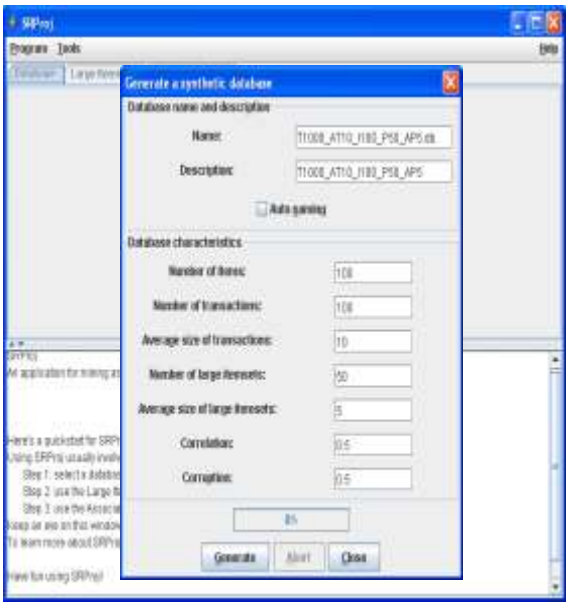

Fig 5.6: Generating a synthetic database

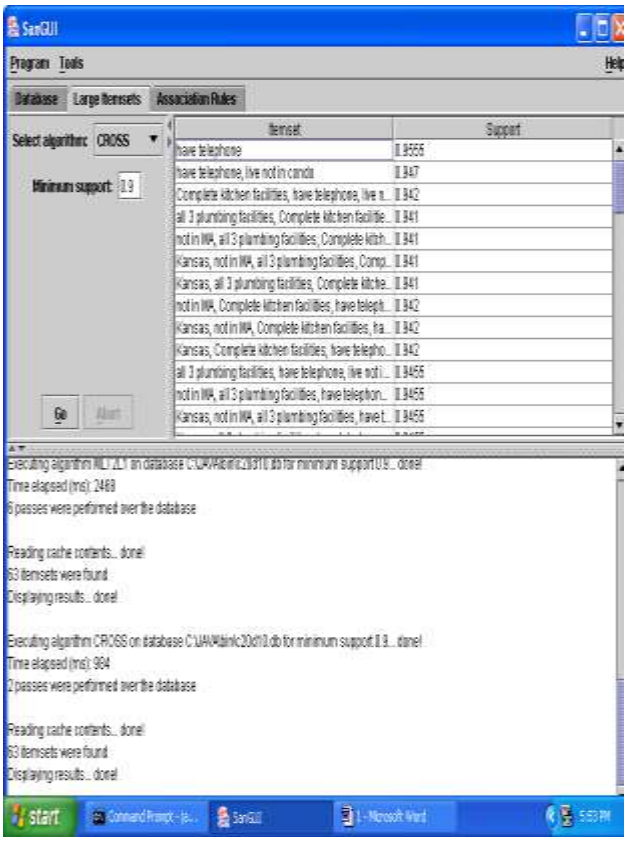

Fig 5.7: Selecting the stored database

8. REFERENCES

[1] R. Agrawal and R. Srikant, "Fast Algorithms for mining Association Rules", Proceeding of the 20th VLDB Conference, Chile, Sept. 1994, Page(s): 487-499.W.-K. Chen, *Linear*

[2] *Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] M. S. Chen, J. Han, and P. S. Yu, "Data Mining: An Overview from a Database Prespective", IEEE Transaction on Knowledge and Data Engineering, Vol. 8, No. 4, July/Aug 1996, Page(s): 866-833.

[4] Charu C. Aggarwal and Philip S. Yu, "Mining Associations with the Collective Strength Approach", IEEE Transaction on Knowledge and Data Engineering, Vol. 13, No. 6, Nov./Dec. 2001, Page(s): 863-873.

[5] fig Liu, Wynne Hsu and Yiming Ma, "Mining Association Rules with Multiple Minimum Support", IEEE Transaction on Knowledge and Data Engineering, Vol 13, No. 1, Jan/Feb 2001, Page(s): 64-78.

[6] N. Rajkumar, M.R. Karthik and S. N. Sivananadam, "Fast Algorithm for Mining Multilevel Association Rules", IEEE Transaction on Knowledge and Data Engineering, Vol 13, No. 1, Nov./Dec 2003, Page(s): 64-69.

[7] Edith Cohen, Mayur Datar and Shinji Fajiwara, "Finding Interesting Associations without support Pruning", IEEE Transaction on Knowledge and Data Engineering, Vol 13, No. 1, Jan/Feb 2001, Page(s): 64-78.

[8] Charu C. Aggarwal and Philip S. Yu, "Mining Associations with the Collective Strength Approach", IEEE Transaction on Knowledge and Data Engineering, Vol. 13, No. 6, Nov./Dec. 2001, Page(s): 863-873.