Comparative Investigations and Performance Evaluation for Multiple-Level Association Rules Mining Algorithm

Suraj Srivastava
Department of Computer Science and Engineering
National Institute of Technology, Jalandhar, Punjab, India

Deepti Gupta
Department of Computer Science and Engineering
National Institute of Technology, Jalandhar, Punjab, India

Harsh K Verma
Department of Computer Science and Engineering
National Institute of Technology, Jalandhar, Punjab, India

ABSTRACT
This paper focuses on the comparative investigation and performance evaluation of the ML_TMLA algorithm that generates multiple transaction tables for all levels in one database scan with that of ML_T2L1 and ML_T1LA algorithms. The performance study has been carried out on different kinds of data distributions (three synthetic and one real dataset) and thresholds that identify the conditions for algorithm selection. The AR Tool has been used for the experimental and comparative evaluation of the proposed algorithm with other algorithms.

Keywords
Data mining, Knowledge discovery in databases, Association rules, multiple-level association rules.

1. INTRODUCTION
Various applications of computers, database technologies and automated data collection techniques require large amount of data to be collected into databases. It, therefore, creates great demands for analyzing such data and turning it into useful knowledge. Data mining or Knowledge Discovery in Database (KDD) emerges as a solution to the data analysis problem. Association rules is one of the data mining techniques that can be used to discover interesting rules or relationships among attributes in databases. It is often desirable to discover knowledge at multiple conceptual levels, which shall provide a spectrum of understanding, from general to specific, for the underlying data. Mining association rules from large data sets has been a focused topic in recent research into knowledge discovery in databases [1, 2, 3, 11, 13, and 14].

In order to explore multiple-level association rule mining, one needs to provide data at multiple levels of abstraction as well as 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. In many applications, the taxonomy information is either stored implicitly in the database, such as, Wonder wheat bread is a wheat bread which is in turn bread, or provided by experts or users, such as, Freshman is an undergraduate student, or computed by applying some cluster analysis methods [7]. With the recent development of data warehousing and OLAP technology, arranging data at multiple levels of abstraction has been a common practice [5]. Therefore, in this study, it has been assumed that such concept taxonomies exist and the study focuses at the second requirement, the efficient methods for multiple-level rule mining.

There are several possible directions to explore efficient mining of multiple-level association rules. One method is the direct application of the existing single level association rule mining methods to multiple-level association mining. One may, for example, apply the Apriori algorithm [2] to examine data items at multiple levels of abstraction under the same minimum support and minimum confidence thresholds. Second method is the application of different minimum support thresholds and possibly different minimum confidence thresholds as well as mining associations at different levels of abstraction. Especially, the most typical case is explored: progressively reducing the minimum support thresholds at lower levels of abstraction. This leads to mining interesting association rules at multiple concept levels, which may not only discover rules at different levels, but may also have high potential to find nontrivial, informative association rules because of its flexibility for focusing the attention to different sets of data and applying different thresholds at different levels.

The necessity for mining multiple-level association rules or using taxonomy information at mining association rules has also been observed by other researchers such as in [4]. A major difference between this study and theirs is that they use the same support threshold across all the levels [4], whereas we have used different support thresholds for different levels of abstraction and different datasets (three synthetic & one real data set).

Moreover, using a single support threshold will allow many uninteresting rules to be generated together with the interesting ones if the threshold is rather low, but will disallow many interesting rules to be generated at low levels if the threshold is rather high. Therefore, in this study, attempt has been made on how to identify and remove the redundant rules across different levels.

2. MINING MULTIPLE-LEVEL ASSOCIATION RULES
Association rules are an important class of regularities within data which have been extensively studied by the data mining community. The general objective here is to find frequent co-occurrences of items within a set of transactions. These found co-occurrences are called associations. The idea of discovering such rules is derived from market basket analysis where the goal is to mine patterns describing the customer's purchase behavior [12].

The problem of mining association rules can be stated as follows: $I=\{i_1, i_2, \ldots, i_m\}$ is a set of items, $T=\{t_1, t_2, \ldots, t_n\}$ is a set of transactions, each of which contains items of the itemset I. Thus, each transaction t_i is a set of items such that t_i subset and equal I. An association rule is an implication of the form: $X \rightarrow Y$, where X

subset of *I*, *Y* subset of *I* and $X \cap Y = \Phi$. *X* (or *Y*) is a set of items, called itemset [12].

Looking at an association rule of the form $X \rightarrow Y$, *X* would be called the antecedent, *Y* the consequent. It is obvious that the value of the antecedent implies the value of the consequent. The antecedent, also called the "*left hand side*" of a rule, can consist either of a single item or of a whole set of items. This applies for the consequent, also called the "right hand side", as well.

The most complex task of the association rule mining process is the generation of frequent itemsets. Different combinations of items have to be explored, especially in large databases, which can be a very computation-intensive task. Often, a compromise has to be made between discovering all itemsets and computation time. Generally, only those itemsets that fulfill a certain support requirement are taken into consideration. Support and confidence are the two most important quality measures for evaluating the interestingness of a rule as described.

In order to study the mining of association rules from a large set of transaction data, it has been assumed that the database contains (1) 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 belongs to T (for i = p, . . . , q), and T is the set of all the data items in the item data set; and (2) the description of the item data set, D, which contains the description of each item in T in the form of (A_i, description i), where A_i belongs to T.

3. METHODS FOR MINING MULTIPLE-LEVEL ASSOCIATION RULES

The methods for mining multiple-level association rules use a hierarchy-information encoded transaction table, instead of the original transaction table, in iterative data mining. First, a data mining query is made in relevance to a portion of the transaction database, such as food, instead of all the items. Second, encoding can be performed during the collection of task-relevant data, and thus there is no extra "encoding pass" required. Third, an encoded string, which represents a position in a hierarchy, requires fewer bits than the corresponding object-identifier or bar-code. Moreover, encoding allows more items to be merged (or removed) due to their identical encoding, which further reduces the size of the encoded transaction table. The encoding can always be performed on the fly [6].

3.1 ML_T2L1 Algorithm

It is a top-down, progressively deepening process which collects large itemsets at different conceptual levels. Starting at level 1, it derives 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). After finding the frequent itemsets, the set of association rules for each level l can be derived from the frequent itemsets LL [l] based on the minimum confidence at this level, minconf[*l*] as in [3].

3.2 Variations of the Algorithm for Potential Performance Improvement

Potential performance improvements of Algorithm ML_T2Ll have been considered by exploration of the sharing of data structures and intermediate results and maximally generation of results at each database scan, etc. which leads to the following variations of the algorithm [8][9], that is, ML_TlLA: using only one encoded transaction table (thus Tl) and generating L[*I*, l] for all the levels at one database scan (thus LA); ML_TMLl: using multiple encoded transaction tables and generating L [*I*, l] for one corresponding concept level and ML_T2LA: using two encoded transaction tables (T[1] and T[2]) and generating L[*I*, l] for all the levels at one database scan.

3.3 Algorithm ML_TMLA

INPUT: (1) T[1], a hierarchy information-encoded and task-relevant set of a transaction database, in the format of <TID, Itemset>, in which each item in the Itemset contains encoded conceptual hierarchy information, and (2) the minimum support threshold (minsup[l]) for each conceptual level l.

OUTPUT: Multiple-level large itemsets.

The procedure is described as follows:

1. {L[1,1],……..L[max_*l*,1]} := get_all_large_1_itemsets (T[*l*]);
2. {T[*l*+1],L[*l*+1,1]} := get_filtered_T_table (T[*l*],L[*l*,1]);
3. for (k := 2; L[*l*,k-1] ≠ Ø; k++) do begin
4. for (*l* := 1; *l* < max_l; *l*++) do
5. if L[*l*,k-1] ≠ Ø then begin
6. C[*l*] := get_candidate_set (L[*l*, k-1]);
7. foreach transaction t**Є**T[*l*+1] do begin
8. D[*l*] := get_subsets (C[*l*], t); // Candidates contained in t
9. foreach candidate c **Є** D[*l*] do c.support ++;
10. end
11. L[*l*,k] := { c**Є**C[*l*,*k*] | c.support ≥ minsup[*l*]}
12. end
13. end
14. for (*l*:= 1; *l* < max_ *l* ; *l* ++) do LL[*l*] = $\cup_k$ [*l*,k];

According to Algorithm ML_TMLA, the discovery of large support items at each level proceeds as follows.

Step 1: At the first scan of T[*l*], 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[*l*], 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].

Step 2: The first scan of T[*l*] generates the large 1-itemsets L [*l*, 1] which then serves as a filter to filter out from T[*l*] any small items or transactions containing only small items. A new table T [*l*+1] results from this filtering process and is used in the generation of large k-itemsets at level *l*. The filtered transaction table T[*l*+1] is derived by "get_filtered_t_table (T[*l*], L[*l*,1])", which uses L[*l*, 1]

as a filter to filter out (a) any item which is not large at level 1, and (b) the transactions which contain no large items.

Step 3: 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 serves as a filter to remove from T[l] any small items or transactions containing only small items and results in T[l+ 1], which will be used for the generation of large k-itemsets (for k > 1) at level l and table T [l + 2] at the next lower level.

Step 4: 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 [13]. The get_subsets function will be processed against the candidate sets at all the levels at the same time by scanning T[l] 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.

Step 5: For each transaction t in T[2].for each of t's K-item subset c, increment c's support count if c is in the candidate set C[l,k]. Then collect into L[l,k] each c(together with its support) if its support is no less than minsup[l].

Step 6: The large itemsets at level l, LL [l], is the union of L [l, k] for all the k's.

After finding the large itemsets, the set of association rules for each level l can be derived from the large itemsets LL [l] based on the minimum confidence at this level, *minconf* [l]. This is performed as follows [10]. For every large itemset r, if a is a nonempty subset of r, the rule "a → r - a" is inserted into rule_set [l] if support(r)/support (a) ≥ *minconf* [l], where *minconf* [l] is the minimum confidence at level l.

4. PERFORMANCE STUDY

The comparative analysis and evaluation has been conducted on different kinds of data distributions (three synthetic and one real dataset) and thresholds, which identify the conditions for algorithm selection. The AR Tool has been used to generate the results and perform the comparative investigations of ML_T1LA, ML_TML1 and MLTMLA algorithms. The performance of the different multiple-level association rule mining algorithms has been experimentally evaluated in the context of execution time (milliseconds) and disk input/output on different datasets. This section describes the various datasets, followed by a description of the experimental results [15].

4.1 Datasets

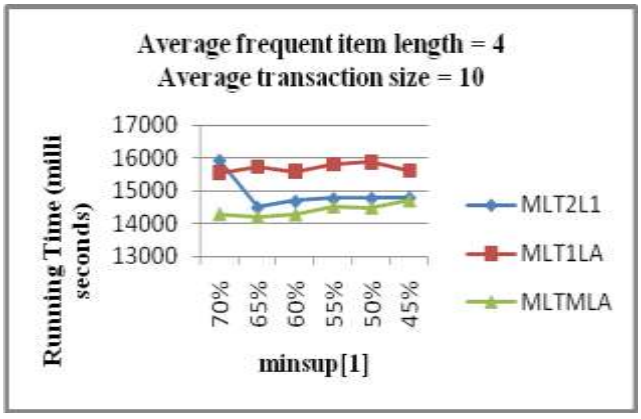
Four different datasets three synthetic and one real have been used in the performance comparison of the algorithms described above. The synthetic datasets used were generated using the AR tool. Table 1 summarizes the names and parameter settings for each dataset. For the synthetic datasets N (Number of items) was set to 1000 and |L| (Number of maximal potentially large itemsets) was set to 2000. We chose three values for |T|: 5, 10, and 20. We also chose three values for |I|: 2, 4, and 6. The number of transactions was set to 100,000. The real dataset used in our experiments was MUSHROOMS (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mushroom/agaricus-lepiota.data).

Table 1. Parameter settings of datasets

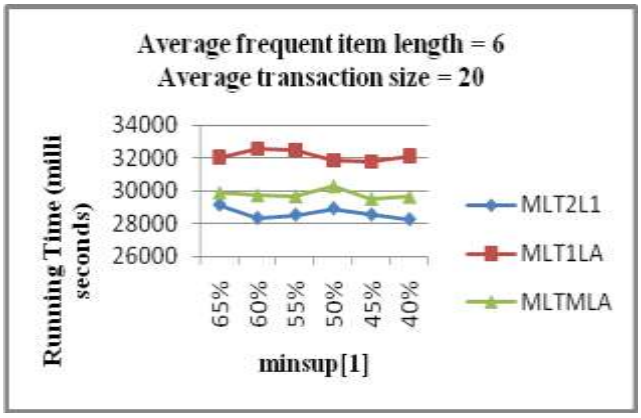
Name	# of objects	Average Size	# of items
T100000 AT10 I1000 P2000 AP 4dB	100k	10	1000
T100000 AT20 I1000 P2000 AP 6dB	100k	20	1000
T100000 AT5 I1000 P2000 AP 2dB	100k	5	1000
MUSHROOMS	8416	23	128

4.2 Experimental Results-Execution Time (milliseconds)

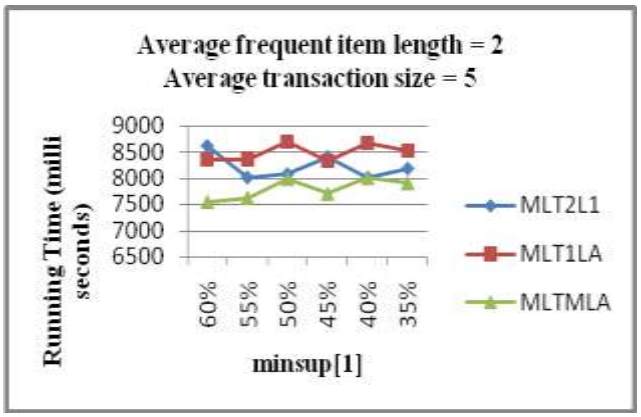
Fig.1 shows the performance comparison of ML_T2L1, ML_T1LA, and ML_TMLA algorithms with respect to running time under various support values for minimum support threshold level 1 in case of datasets T100000 AT10 I1000 P2000 AP 4dB, T100000 AT20 I1000 P2000 AP 6dB and T100000 AT5 I1000 P2000 AP 2dB respectively.



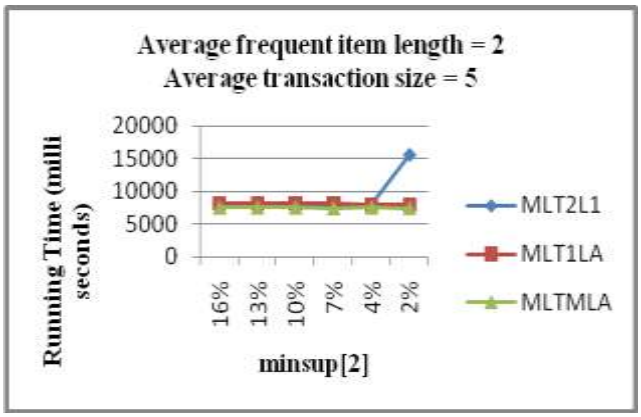
(a)



(b)



(c)



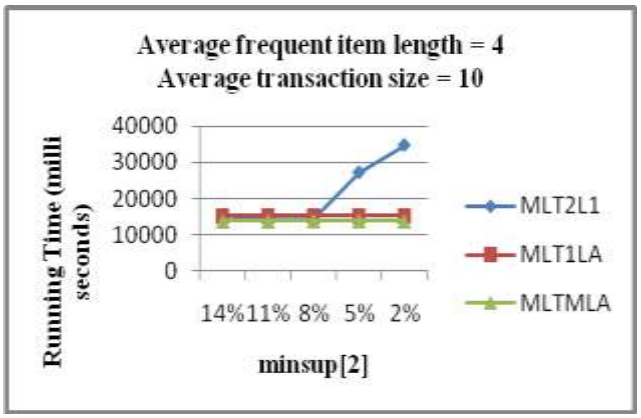
(b)

Fig.1 Execution Time comparison at minsup[1] for (a) T100000 AT10 I1000 P2000 AP 4dB (b) T100000 AT20 I1000 P2000 AP 6dB (c) T100000 AT5 I1000 P2000 AP 2dB

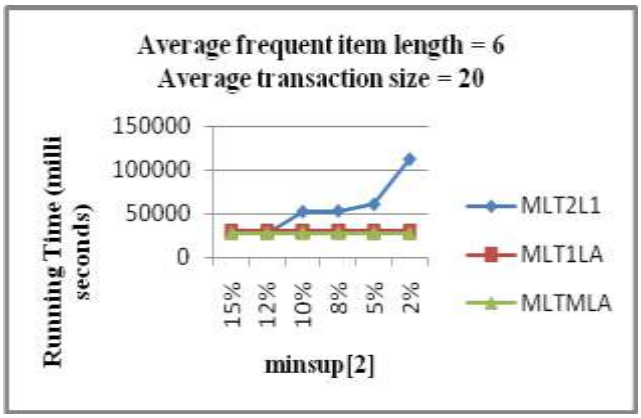
It has been observed that as the minimum support decreases the execution times of all the algorithms increase because of increase in the total number of candidate and large itemsets. However, the execution time for ML_TMLA algorithm is lowest among the three algorithms. In other words, it executes faster.

Fig.2 indicates the performance comparison of ML_T2L1, ML_T1LA, and ML_TMLA algorithms with respect to running time under various support values for minimum support threshold level 2 in case of datasets T100000 AT10 I1000 P2000 AP 4dB, T100000 AT20 I1000 P2000 AP 6dB and T100000 AT5 I1000 P2000 AP 2dB respectively.

It has been concluded that on an average the ML_TMLA algorithm runs faster than the ML_T2L1 and ML_T1LA algorithms for all support levels.



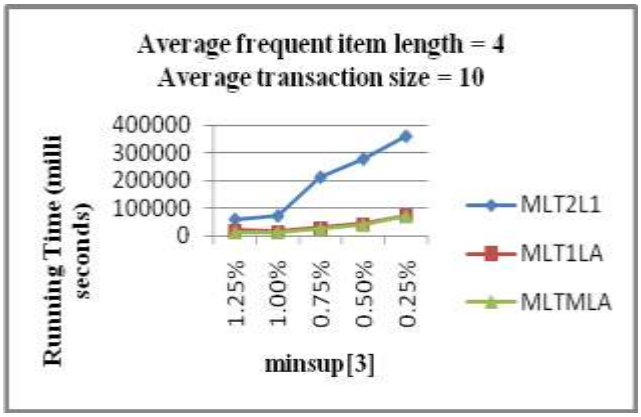
(a)



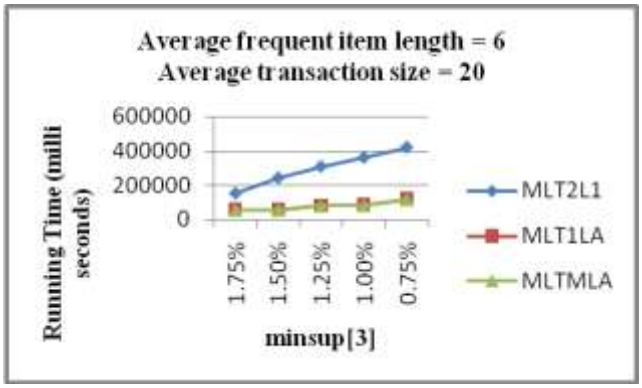
(c)

Fig.2 Execution Time comparison at minsup[2] for (a) T100000 AT10 I1000 P2000 AP 4dB (b) T100000 AT20 I1000 P2000 AP 6dB (c) T100000 AT5 I1000 P2000 AP 2dB

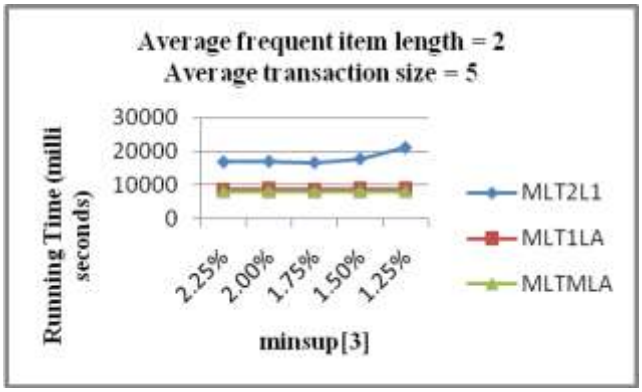
Fig.3 depicts the performance comparison of ML_T2L1, ML_T1LA, and ML_TMLA algorithms with respect to running time under various support values for minimum support threshold level 3 in case of datasets T100000 AT10 I1000 P2000 AP 4dB, T100000 AT20 I1000 P2000 AP 6dB and T100000 AT5 I1000 P2000 AP 2dB respectively.



(a)



(b)

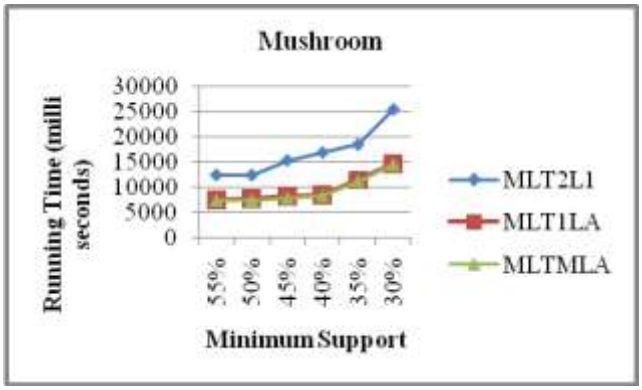


(c)

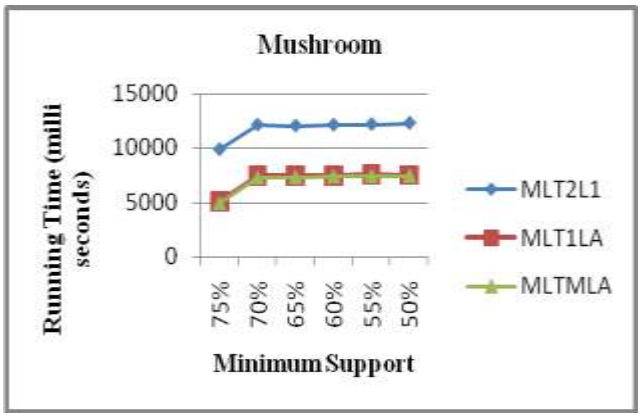
Fig.3 Execution Time comparison at minsup[3] for (a) T100000 AT10 I1000 P2000 AP 4dB (b) T100000 AT20 I1000 P2000 AP 6dB (c) T100000 AT5 I1000 P2000 AP 2dB

It has been concluded that on an average the ML_TMLA algorithm runs faster than the ML_T2L1 and ML_T1LA algorithms for all support levels.

Fig.4 indicates the performance comparison of ML_T2L1, ML_T1LA, and ML_TMLA algorithms with respect to running time under various support values for minimum support threshold value varying between 55% to 30 % and 75% to 50% in case of Mushroom dataset. It has been concluded that on an average the ML_TMLA algorithm runs faster than the ML_T2L1 and ML_T1LA algorithms for all support levels.



(a)



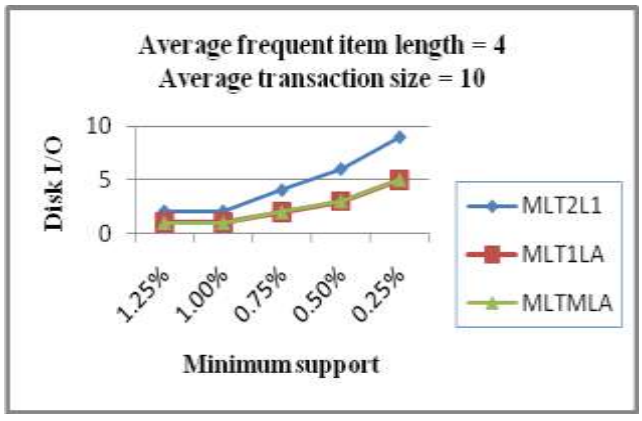
(b)

Fig.4 Execution Time comparison for Mushroom dataset at minsup values varying from (a) 55% to 30% and (b) 75% to 50%

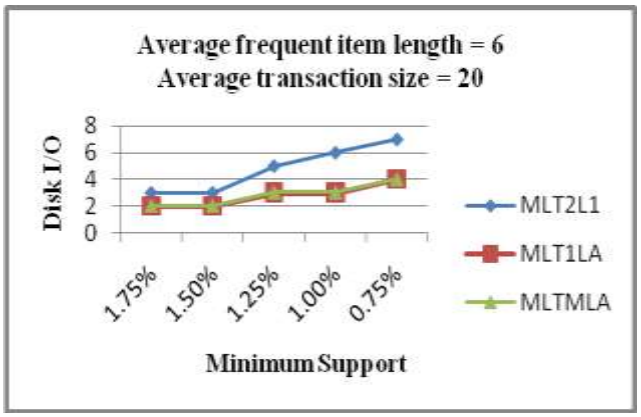
4.3 Experimental Results-Disk Input/Output

Fig.5 shows the performance comparison of ML_T2L1, ML_T1LA, and ML_TMLA algorithms with respect to disk input/output under various support levels in case of datasets T100000 AT10 I1000 P2000 AP 4dB, T100000 AT20 I1000 P2000 AP 6dB and T100000 AT5 I1000 P2000 AP 2dB respectively.

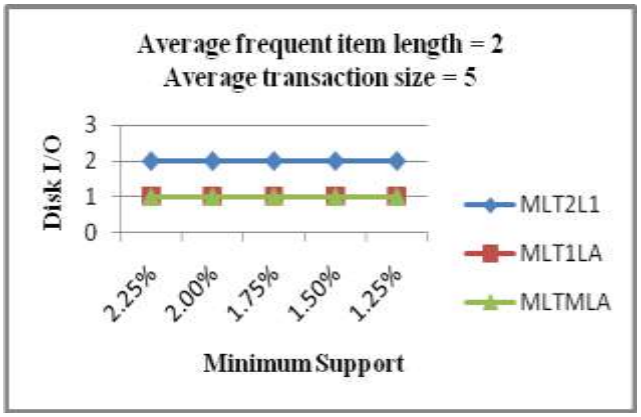
It has been observed that the ML_T1LA and the ML_TMLA algorithms depict almost same disk input/output for all the datasets chosen. It has been observed that for the minimum support levels taken the ML_T2L1 algorithm reads the database multiple number of times ranging from 2 to 9 for different datasets. However, the number of read operations performed by the ML_T1LA and the ML_TMLA algorithms on the database are comparatively less. This is because the exact number depends on the minimum support and the data characteristics and, therefore, cannot be determined in advance. Moreover, when the support level is set very high no itemsets are found to have the required support. In such cases, the algorithms read the database only once.



(a)



(b)



(c)

Fig.5 Disk Input/Output comparison for (a) T100000 AT10 I1000 P2000 AP 4dB (b) T100000 AT20 I1000 P2000 AP 6dB (c) T100000 AT5 I1000 P2000 AP 2dB

5. CONCLUSION

The work described in this paper comparatively evaluates the performance of the ML_TMLA algorithm that generates multiple transaction tables for all levels in one database scan with that of ML_T2L1 and ML_T1LA algorithms. The execution time for ML_TMLA algorithm is lowest among the three algorithms. It has been concluded that on an average the ML_TMLA algorithm runs faster than the ML_T2L1 and ML_T1LA algorithms for all support levels. It has been observed that the ML_T1LA and the ML_TMLA algorithms depict almost same disk input/output for all the datasets chosen. Moreover, for the minimum support levels taken the ML_T2L1 algorithm reads the database multiple number of times ranging from 2 to 9 for different datasets. However, the number of read operations performed by the ML_T1LA and the ML_TMLA algorithms on the database are comparatively less. The results obtained clearly indicate that the proposed algorithm executes fast and performs less number of read operations.

6. REFERENCES

[1] Agrawal, R., Imielinski, T., and Swami, A. 1993. Mining association rules between sets of items in large databases. In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data, pp. 207-216, Washington, D.C.

[2] Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In Proc. 1994 Int. Conf. Very Large Data Bases, pp. 487-499, Santiago, Chile.

[3] Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In Proc. 1995 Int. Conf. Data Engineering, pp. 3-14, Taipei, Taiwan.

[4] Agrawal R. ,Srikant R. 1995. Mining Generalized Association Rules. Proc. 1995 Int'l Conf. Very Large Data Bases, pp. 407±419, Zurich.

[5] Chaudhuri S. ,and Dayal, U. 1997. An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record, vol. 26, pp. 65±74.

[6] Fu Yongjian. 1996. Discovery of Multiple-Level Rules from Large Databases. Phd Thesis.

[7] Han J., Cai Y., and Cercone N. 1993. Data-Driven Discovery of Quantitative Rules in Relational Databases. IEEE Trans. Knowledge and Data Eng., vol. 5,pp. 29±40.

[8] Han Jiwawei,Fu Yongjian. 1999. Mining Multiple-Level Association Rules in Large Databases. IEEE.

[9] Han, Jiawei and Yongjian, Fu. 1995. Discovery of Multiple-Level Association Rules from Large Databases. Proceedings of the 21st VLDB Conference Zurich, Swizerland.

[10] Han, Jiawei. 2005. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1558609016.

[11] Klemettinen, M., Mannila, H. and Ronkainen, P., Toivonen, H., and Verkamo, A. I. 1994. Finding interesting rules from large sets of discovered association rules. In Proc. 3rd Int '1 Conf. on Information and Knowledge Management, pp. 401-408, Gaithersburg, Maryland.

[12] Liu, Bing. 2007. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer.

[13] Park, J.S., Chen, M.S. and Yu, P.S. 1995. An effective hash-based algorithm for mining association rules. In Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data, pp. 175-186, San Jose, CA.

[14] Piatetsky-Shapiro, G. 1991. Discovery, analysis, and presentation of strong rules. In Knowledge Discovery in Databases, pp. 229-238, AAAIIMIT Press.

[15] Wasilewska Anita. 2007. Mining Association Rules in Large Databases.