Performance Analysis of Frequent Itemset Mining Using Hybrid Database Representation Approach

Shariq Bashir, Dr. A. Rauf Baig
Department of Computer Science
FAST-National University of Computer and Emerging Science, Pakistan
shariq.bashir@nu.edu.pk, rauf.baig@nu.edu.pk

Abstract

Frequent Itemset Mining is considered to an important research oriented task in data mining, due to its large applicability in real world applications. In recent years lot of algorithms and techniques are proposed for enumerating itemsets from transactional databases. In which some are best for dense type datasets, while some are best for sparse type datasets. Currently there is no single algorithm exist that is best for all type of datasets (sparse as well as dense). The main limitation of previous algorithm is that, they depend upon single approach and do not combine the best features of multiple approaches for speedup the process of itemset mining. In this paper, we first compare and contract the two main itemset mining strategies on different itemset mining factors, scalability of algorithm, item search order, dataset projection and itemset frequency counting. Then we introduce a new hybrid strategy that combines the best features of existing strategies. Our different experiments on benchmark datasets show that mining all and maximal frequent itemsets using hybrid approach outperforms the previous algorithms on almost all types of dense and sparse datasets, which shows the effectiveness of our approach.

Keyword: Data Mining, Association rules mining, Frequent itemset mining, Maximal frequent itemset mining.

1. Introduction

Since the introduction of Association Rule Mining (ARM) problem by (Agrawal 1993) [1], now it has become a core problem in data mining tasks, and has been successfully applied to market basket analysis, classification, clustering, sequential pattern, web mining and text mining applications. ARM is an undirected or unsupervised data mining technique, which works on variable length data, and it produces clear and understandable results. The process of ARM consists of two main steps:

1 Find the frequent item sets or large item sets with a minimum support.

2 Use the large item sets to generate association rules that meet a confidence threshold.

The prototypical application of ARM is in *market basket analysis*, where the items represent products and the records the point-of-sales data at large grocery or departmental stores. An example rule could be that, "90% of customers buying product A also buy product B."

The association mining task can be stated as follows: Let *T* is the transactions of the database and *X* is the set of items from ***{X ε (1 to n)}***. A set of items is called an *itemset*. An itemset *X* is frequent if it contains at least $\sigma(X)$ transactions, where σ is the minimum support. The set of all frequent itemsets are denoted by FI. An itemset *X* is maximal if is not subset of any other known frequent itemset. An itemset *X* is closed if it is not appear in any other known frequent itemset with same transactions as *X*. We denote *MFI* as set of all maximal frequent itemsets and *FCI* as set of all frequent closed itemset. Any maximal frequent itemset *X* is a frequent closed itemset since no nontrivial superset of *X* is frequent. Thus we have $MFI \sqsubseteq FCI \sqsubseteq FI$.

In last five years a number of all [2, 8, 9, 12], maximal [4, 6] and closed [10, 14] itemset mining algorithms are proposed. The basic strategy of mining frequent itemset by all these algorithms is divided into two main approaches: Candidate-generate-and-test (CGaT) [2] and pattern growth approach [8]. As reported in [5], no single approach yet has proven that it is best and efficient for all types of datasets (sparse or dense).

In this paper, we compare pattern growth approach and CGaT approach in terms of scalability, item search order, projection and itemset frequency counting. We also show the significance of these factors on hybrid approach [3], which we have recently proposed. In [3] we compared hybrid approach with only CGaT algorithms on MFI problem. In this we perform a detail comparison of hybrid approach with pattern growth and CGaT approach on all and maximal itemset problems.

Our experiments on different sparse and dense benchmark datasets show that mining frequent itemset with hybrid approach is more efficient and faster than other two approaches, which gives a global best solution.

2. Pattern growth Approach

The pattern growth approach is a divide-and-conquer technique, which divides search space into disjoint sub search spaces (Partitions) called conditional databases D_i. The conditional database of item a_i can be construct by $a_i \cup á$ ($á$ is the parent conditional pattern), which includes all those transactions where $a_i \cup á$ is appear as a prefix. In short pattern growth approach eliminates the recursive candidacy generation and test operations. The FP-growth basis algorithm is described in Figure 1.

```
Procedure FP-Growth (Tree, á)

1.  for each item ai in Tree do
2.      generate pattern B = ai U á with support
        ai.support
3.      construct B's conditional pattern and
        B's conditional FP-Tree TreeB
4.      if TreeB $\neq \theta$
5.          then call FP-Growth (TreeB, B)
```

Figure 1: pattern growth algorithm

2.1 Scalability

The basis operation of pattern growth mining approach is recursive creation of $a_i \cup á$ conditional patterns, conditional FP-tree and counting $a_i \cup á$ frequency. For sparse datasets recursion creation of $a_i \cup á$ conditional patterns from parent nodes is almost equal to the total number of frequent itemset, which takes a large memory for storage and also large processing time for itemset mining. Therefore numbers of scalable pseudo-construction techniques H-Mine [11], OP [9] are proposed. However, pseudo-construction cannot reduce traversing cost as efficiently as physical construction.

2.2 Item search order

As per literature review, two item search order are proposed, static lexicographically order and dynamic ascending frequency order [10]. Static order is a fixed order and all the nodes of search spaces follow the same order. Pseudo construction techniques H-Mine and OP follow this order. As alternative, ascending frequency order, adopt by AFOPT [10], which not only reduces search space but also reduces the total number of recursive conditional patterns.

2.3 Projection

Pattern growth approach is a divide-and-conquer methodology, and projecting compressed or relevant conditional patterns on lower nodes of search space is a key feature of this approach, which dramatically reduces the total mining time. Construction of projected conditional FP-tree $Tree_{ai \cup á}$ from parent node's conditional FP-tree $Tree_{á}$ can be considered from the following definition.

Definition1: Given a parent conditional FP-tree $Tree_{á}$. Item a_i projected conditional FP-tree $Tree_{ai \cup á}$ is a sub tree of its parent FP-tree $Tree_{á} \supset Tree_{ai \cup á}$, which can be construct by including only those branches where a_i is appear either as a node or leave.

2.4 Frequency counting

There are two ways of checking, whether item a_i is frequent or infrequent. First, is to direct count from transaction database which takes O(n) cost. Second one is to collect support from item a_i parent conditional pattern, which depend on the size of conditional FP-tree, not on size of transaction database. Lemma1 describes the completeness of frequency counting from conditional FP-tree.

Lemma1: Given a conditional pattern *CDP* and min_sup threshold. The support of frequent itemset a_i can be derived from *CDP*.

Proof: Based on the FP-tree $Tree_{CDP}$, the support of frequent itemset a_i in $Tree_{CDP}$ can be count from a_i mapped nodes (mapped nodes can be retrieved from header table) to root of $Tree_{CDP}$.

3. Candidate Generate and test approach (CGaT)

```
Procedure CGaT (F:Itemset)

1.  Output F
2.  For each e $\varepsilon$ E, e > tail(F) do
3.      if F U {e}is frequent item call CGaT(F U
        {e})
```

Figure 2: Candidate-generate-and-test algorithm

The CGaT approach essentially uses block nested lop join i.e. the search space is the inner relation and it is divided into blocks according to itemset length (G. Liu 2003). The basis operation involves in CGaT is recursive creation of candidate itemset F_i from parent itemset P_{i-1}, and testing support ($F_i \cup P_{i-1}$) whether it is frequent or infrequent. Figure 2 shows the basis strategy of CGaT algorithm.

3.1. Scalability

The search space of CGaT can be traverse by depth first search order (DFS) or breadth first search order (BFS), where DFS is consider to be most popular and scalable approach. By traversing search space with DFS only one tree path is explored at any time so total space consumption cost always remains equal to O(ML), where ML is the maximum length of tree path.

3.2. Item search order

Dynamically following ascending frequency order is key feature of CGaT approach. Almost all CGaT algorithms [2, 4] follow ascending frequency order, which drastically reduces the itemset search space and mining time.

3.3. Projection and Frequency counting

Most of CGaT algorithms use vertical tid [6] or vertical bitmaps [4] as an initial database representation. Vertical database representation has two major disadvantages. First, with vertical representation approach we can't share prefix on dense datasets as FP-tree. Second, vertical database representation approach projects almost all transactions on each node of search space which takes O(n) cost in itemset frequency counting.

Burdick at al in [4], proposed a bit-vector projection technique known as projected bitmap. The main deficiency of projection using projected bitmap technique is that, it requires a high processing cost (time) for its creation. Due to this reason, they proposed adaptive compression in [4], since projection is done only when saving from the compressed bitmaps outweigh the cost of projection. However, with adaptive compression, projection cannot be possible on all nodes of search space. The major advantage of vertical technique is that, it optimizes frequency counting cost with a factor of 1/32, if we represent 32 rows per single vertical bit-vector region [13].

4. Hybrid Approach

As from related work review, we can conclude that pattern growth approach is better in terms of projection and frequency counting, where CGaT is better in terms of scalability and item search order. In [3] we combined the best features of both approaches into a single hybrid approach, which not only optimizes itemset frequency calculation cost by using pattern growth approach [11], but it also reduces search space by ascending frequency order using vertical bitmaps [4].

In [3], we combined H-Mine [11] (a pattern growth approach) and vertical bitmaps [4] into a single hybrid approach. The basic strategy of our hybrid approach for mining complete itemset is, it traverses search space by DFS and on each node of search space, it projects relevant transactions by Hyper-Mine and reorder and remove infrequent items from tail by ascending frequency order with using vertical bitmaps. Figure 3 shows a sample hybrid initial database representation.

Table 1 compares pattern growth, CGaT and hybrid approach in terms of scalability, item search order, projection and frequency counting cost.

Table 1: Comparisons of different ARM approaches

Approach	**Scalable**	**Item search order**	**Projection**	**Frequency counting cost**
Pattern growth (physical construction)	Not for sparse datasets	Ascending Frequency order	Yes	O (size of conditional database)
Pattern growth (Pseudo construction)	Yes	Static fix order	Yes	O (size of conditional database)
Candidate generate and test	Yes	Ascending Frequency Order	No	O (size of bitmap)
Hybrid	Yes	Ascending Frequency Order	Yes	O (size of conditional database)

The completeness of projecting transactions of tail items in hybrid approach can be proof from the following lemma2.

Lemma2: Let P be the node of search space and let *proj(P)* be its projection, its tail items t_i projection can construct from P projection.

Proof: We know that all tail items are $t_i \subseteq P$, and *proj(P)* contains all those transactions, where P is appear as a prefix. So tail items *proj*(t_i) can be calculate directly from *proj(P)*, because $proj(t_i) \subseteq proj(P)$, this completes our proof.

A	B	C		
B	D	E	F	G
E	H			
C	E	G	H	
A	C	G		
D	E			
A	C	D	F	H
F	I			

Transaction	A	B	C	D	E	F	G	H	I
1 Bitmap	1	1	1	0	0	0	0	0	0
Array	A	B	C						
2 Bitmap	0	1	0	1	1	1	1	0	0
Array	B	D	E	F	G				
3 Bitmap	0	0	0	0	1	0	0	1	0
Array	E	H							
4 Bitmap	0	0	1	0	1	0	1	1	0
Array	C	E	G	H					
5 Bitmap	1	0	1	0	0	0	1	0	0
Array	A	C	G						
6 Bitmap	0	0	0	1	1	0	0	0	0
Array	D	E							
7 Bitmap	1	0	1	1	0	1	0	1	0
Array	A	C	D	F	H				
8 Bitmap	0	0	0	0	0	1	0	0	1
Array	F	I							

Figure 3: A sample hybrid initial transaction dataset representation

```
Procedure Reorder ( Node P)

1   for each threaded transaction x in proj(P)
2      if P.tail less than ATL
3        for each array item of x {y ε item in x}

4      increment support of y in support table
5            make new threaded link y in header
             table of P

6      else
7      for all tail elements in P.tail y ε P.tail
8     if y is '1' in transaction bitmap of x
9            increment support of y in support
table

10    return support storage
```

Figure 4: Pseudo code for reordering

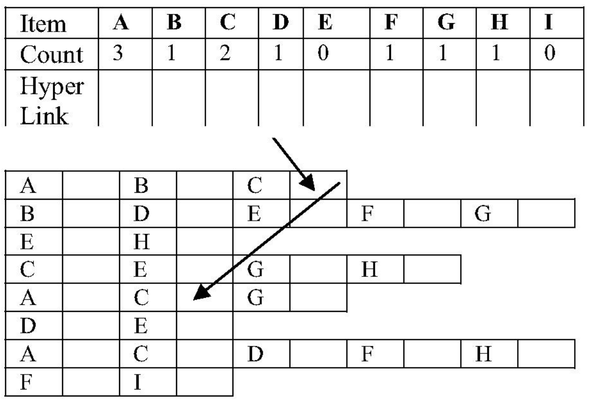
5. The Search Methodology

Let we take a generic iteration of Figure 4 and Figure 5 hybrid algorithm. Procedure *reorder(Node P)* in Figure reorders tail elements by ascending frequency order and removes infrequent tail items. Line from 4 to 5 in Figure makes new projections of frequent tail items of node *P* according to lemma2. Figure 6 shows a projected transaction of itemset AC from its parent projection *proj(A)*.

```
Procedure HybridMining (Node P, Support)

1    Reorder (P)
2    use PEP to trim the tail, and sort items by
     ascending  support
3      for each item x in P.reorder tail
4            HybridMining (x, Support[x])
```

Figure 5: Pseudo of Hybrid approach for frequent itemset mining



Item	A	B	C	D	E	F	G	H	I
Count	3	1	2	1	0	1	1	1	0
Hyper Link									

A		B		C					
B		D		E		F		G	
E		H							
C		E		G		H			
A		C		G					
D		E							
A		C		D		F		H	
F		I							

Only those transactions are included where itemset AC is appear as a prefix

Figure 6: Projection of itemset AC

Note on top level nodes of search space were tail items are larger that ATL (Average Length of transaction database), reordering tail items by traversing items of treaded-transactions *T* of *proj(P)* is more beneficial (see Figure 7) than bitmaps which takes O(m) where m is number of tail items. On lower level nodes where tail items become constant and shorter than ATL, here reordering by bitmaps is more useful. Definition3 describes the reordering cost and tradeoff with both (threaded-transactions and bitmaps) techniques.

Definiion2: Let t_i are tail items of node *P* and let *ATL* is average transaction length of transaction dataset. If total tail items are greater than *ATL* then total cost of reordering by threaded-transactions *T* will be O(m*ATL) where *m* is total number of items in *T*. If tail items are less than *ATL* then total cost of reordering by vertical bitmaps will be O($m*n$) (m are total P tail items and n is number of T transactions) which is less than O($m*ATL$).

Line 3 in Figure 5 makes child nodes of P from frequent tail items and line 4 traverses child nodes in DFS order.

6. Computational Experiments

In this section we now describe the effectiveness of our hybrid approach comparing with pattern growth and CGaT approach. For comparison we used the best implementations of pattern growth and CGaT approach available at http://fimi.cs.helsinki.fi/src/.

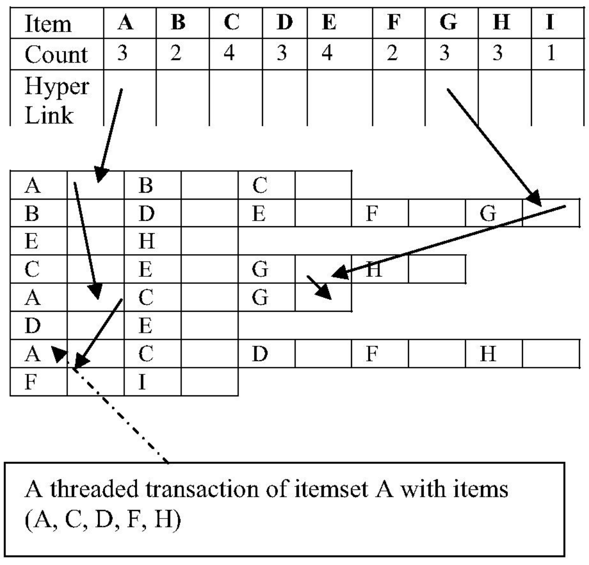


Figure 7: A sample hyper threaded projection for item A and G

All the source code of HybridMine is written in C. Experiments are conducted on Pentium model 3 processor 1.0 GHz with 40 GB ASUS hard disk. For experiment purpose we used small 160 MB main memory, which show s that our hybrid approach is scalable and does not create any memory problems. We also realized in our experiments that pattern growth approach sometimes run out of space on dense datasets, where CGaT approach sometimes run out of space on sparse datasets.

6.1. Datasets used in experiments

The experiments reported in this paper have been conducted on several dense and sparse benchmark datasets which were frequently used in previous work and can be downloaded from http://fimi.cs.helsinki.fi/data/.

6.2. Mining All Frequent Itemset

Let I be the set of items and S be set of transactions, where each transaction $t\varepsilon S$ contains set of items t_{items} which are subset of $t_{item}\unlhd I$. Given a support threshold min_sup, an itemset $X\unlhd I$ is frequent if its support is sup(X) $\geq$ min_sup. If X subset Y is infrequent, we does not need to check sup(X) because we now if subset is infrequent its superset will be also infrequent. Figure 8 shows the performance curves of all algorithms. The performance measure is the execution time of the algorithms datasets with different support thresholds. As we can see from Figure, the Hybrid approach outperforms the other algorithms on almost all types of datasets, and gives global best performance. The performance improvements of Hybrid approach over other algorithms are significant at low support thresholds.

6.3. Mining Maximal Frequent Itemset

Mining MFI is considered to be more advantage able than mining FI, since it mines small and useful long patterns. However, mining MFI is more complicated than mining FI, since for each candidate maximal itemset; we not only check its frequency (support) but also its maximality, which takes O (MFI) cost is worst case. In practice, checking itemset maximally is considered to be an important factor in MFI mining. As per literature review two techniques, progressive focusing [6] and MFI-tree [7] has been proposed for checking MFI maximally, efficiently. Where, progressive focusing is widely used in most of the MFI mining algorithms [4, 7]. Figure 9 shows the performance curves of all algorithms. The performance measure is the execution time of the algorithms datasets with different support thresholds. As we can see from Figure, the Hybrid approach outperforms the other algorithms on almost all types of datasets, and gives global best performance. The performance improvements of Hybrid approach over other algorithms are significant at low support thresholds.

7. Conclusion

In recent years lot of algorithms are proposed for efficient mining of all and maximal frequent itemsets. In which some are best for sparse type datasets, while

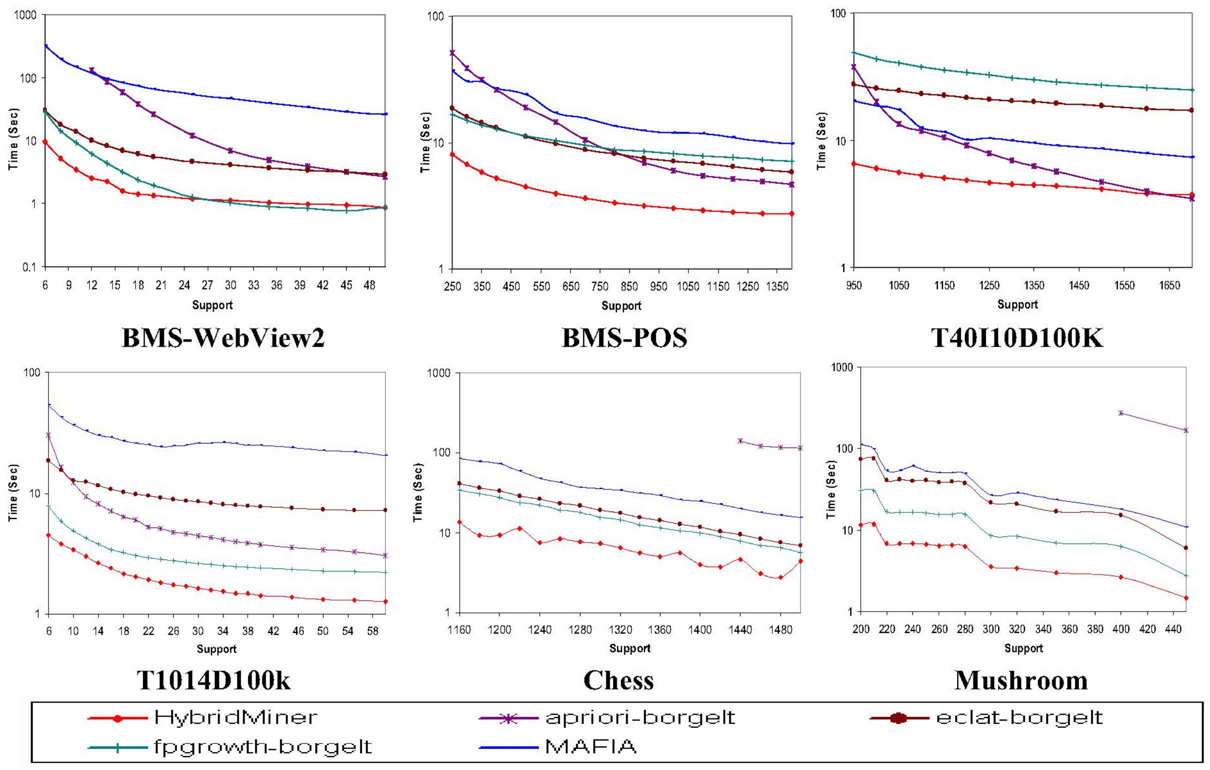


Figure 8: Performance results of Hybrid, CGaT and pattern-growth algorithms on all frequent itemset mining.

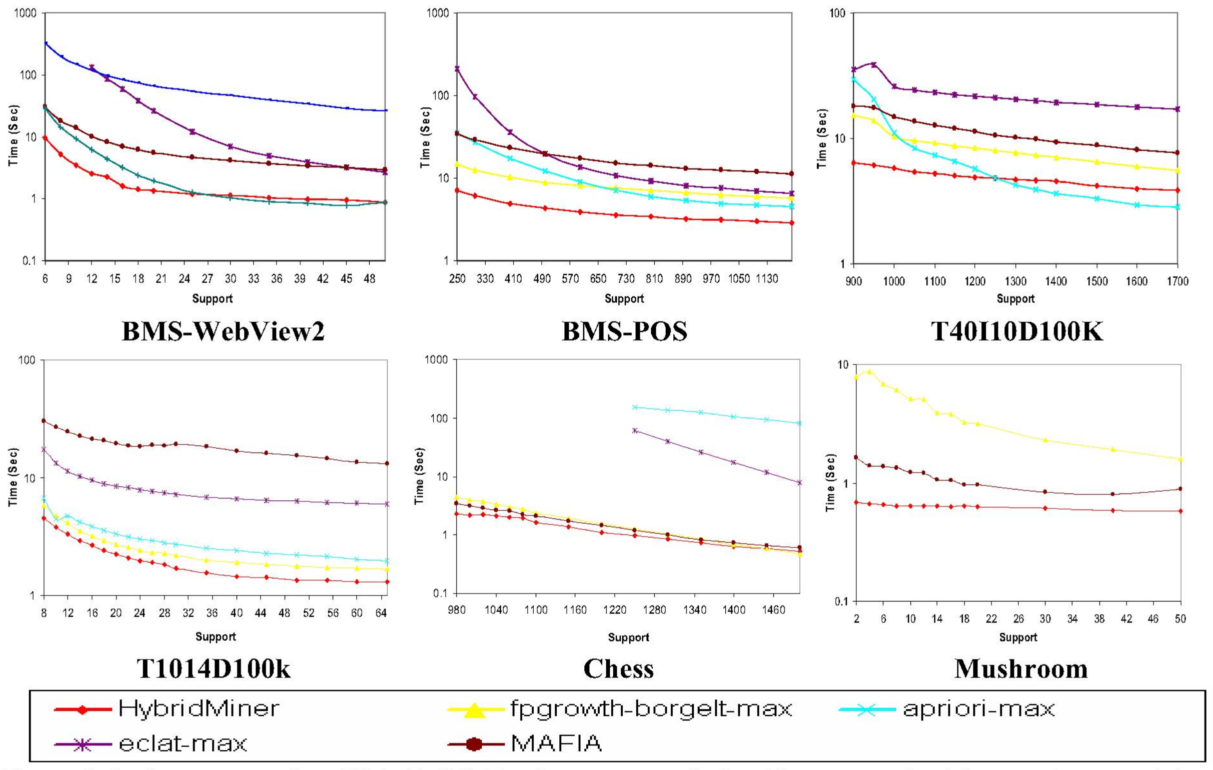


Figure 9: Performance results of Hybrid, CGaT and pattern-growth algorithms on maximal frequent itemset mining.

some are best for dense type datasets. Currently there is no single algorithm exist that shows global best performance on all types of datasets. The main limitation of previous algorithms is that, they really upon only single strategy and do no combine the best features of multiple strategies for speedup the process of itemset mining. In this paper, we show that combining the best features of multiple strategies into a single hybrid is more beneficial and efficient than relying upon single strategy. Our different experimental results on benchmark datasets show that mining all and maximal frequent itemsets using our hybrid approach is more efficient than existing algorithms and gives global best performance.

8. References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACMSIGMOD Int'l Conf. Management of Data*, pp. 207-216, May 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. Int'l Conf. Very Large Data Bases, pp. 487-499, Sept. 1994.

[3] S. Bashir, A. Rauf Baig, "HybridMiner: Mining Maximal Frequent Itemsets using Hybrid Database Representation Appraoch", In *Proc. of 10th IEEE-INMIC conference,* Karachi, Pakistan, 2005.

[4] D. Burdick, M. Calimlim, and J. Gehrke, "Mafia: A maximal frequent itemset algorithm for transactional databases", In *Proc. of ICDE Conf*, pp. 443-452, 2001.

[5] *Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations*, B. Goethals and M.J. Zaki, eds., CEUR Workshop Proc., vol. 80, Nov. 2003, http://CEUR-WS.org/Vol-90.

[6] K. Gouda and M. J. Zaki, "Efficiently mining maximal frequent itemsets", *In ICDM*, pp. 163–170, 2001.

[7] G. Grahne and J. Zhu, "Efficiently Using Prefix-trees in Mining Frequent Itemsets", *In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

[8] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", *In SIGMOD*, pages 1–12, 2000.

[9] J. Liu, Y. Pan, K. Wang, and J. Han. "Mining frequent item sets by opportunistic projection", In *Proc. of KDD Conf*, 2002.

[10] G. Liu, H. Lu, Y. Xu, and J. X. Yu, "Ascending frequency ordered prefix-tree: Efficient mining of frequent patterns", In *DASFAA*, 2003.

[11] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang and D. Yang, "H-Mine: Hyper-structure mining of frequent patterns in large databases", In Proc. of *ICDM* Conf, pp. 441.448, 2001.

[12] A. Pietracaprina and D. Zandolin, "Mining Frequent Itemsets Using Patricia Tries," *Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations*, CEUR Workshop Proc., vol. 80, Nov. 2003.

[13] T. Uno, M. Kiyomi, H. Arimura. *LCM ver 3.: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining.* In: 1st Int'l Workshop on Open Source Data Mining (in conjunction with SIGKDD2005), 2005.

[14] M. J. Zaki and C. Hsiao, "Charm: An efficient algorithm for closed association rule mining", *In Technical Report 99-10, Computer Science, Rensselaer Polytechnic Institute*, 1999.