Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 7, No. 1, February 2016), Pages: 23-34
www.jacr.iausari.ac.ir

Association Rule Mining Using New FP-Linked List Algorithm

Mohammad Karim Sohrabi✉, Hamidreza Hasannejad Marzooni
Department of Computer Engineering, Semnan Branch, Islamic Azad University, Semnan ,Iran
amir_sohraby@yahoo.com; hamidreza.hasannejad.m@gmail.com

Received: 2015/02/10; Accepted: 2015/07/16

Abstract

Finding frequent patterns plays a key role in exploring association patterns, correlation, and many other interesting relationships that are applicable in TDB. Several association rule mining algorithms such as Apriori, FP-Growth, and Eclat have been proposed in the literature. FP-Growth algorithm construct a tree structure from transaction database and recursively traverse this tree to extract frequent patterns which satisfies the minimum support in a depth first search manner. Because of its high efficiency, several frequent pattern mining methods and algorithms have used FP-Growth's depth first exploration idea to mine frequent patterns. These algorithms change the FP-tree structure to improve efficiency. In this paper, we propose a new frequent pattern mining algorithm based on FP-Growth idea which is using a bit matrix and a linked list structure to extract frequent patterns. The bit matrix transforms the dataset and prepares it to construct as a linked list which is used by our new FPBitLink Algorithm. Our performance study and experimental results show that this algorithm outperformed the former algorithms.

Keywords: *Association Rule Mining, Support, frequent pattern, FP-Growth Algorithm, itemset*

1. Introduction

Frequent patterns are patterns that appear frequently in a dataset. In transaction databases, a frequent pattern is a set of items appear frequently together. This item set is also called frequent itemset.

Finding frequent patterns plays a key role in exploring association patterns, correlation, and many other interesting relationships that are applicable in TDB. The issue of finding frequent patterns was first proposed by Agrawal in 1993 to explore and analyze market basket. The results of his analysis indicate that most customers that purchase the good X buy the good Y as well. Therefore, for more convenience of the customers, we should put the goods X and Y beside each other while arranging the goods on the shelves [1].

There are a large number of algorithms to extract frequent patterns. Three main algorithms are Apriori, FP-Growth, and Eclat. Apriori algorithm is based on superficial search and was first proposed by Agrawal and Srikant in 1994 [2]. FP-Growth algorithm is based on deep search and was first proposed by Han et al. in 2000 [3]. Ecalt algorithm has a vertical structure and was proposed by Zaki in 2000 [4]. Each of these algorithms has their own advantages and disadvantages.

In later years, these algorithms have been changed and provided with new structures. The algorithm that is proposed in the present study use a new linked list structure based on FP-Growth algorithm. In this new algorithm, bit matrix and linked list will be used. Firstly, TDB is converted into a bit matrix whose columns and rows indicate items and transactions respectively. A sum row is added to the end of the matrix in which the sum of the numbers of each row is written. Before the bit matrix is created, the first pruning is conducted based on Minimum support. Afterwards, the logic operation of "And" between the rows of the linked list will be carried out. During creating the linked list, pruning operation will be conducted for the second time. Finally, frequent patterns will be found by searching the lined list.
Here is what this paper aims to contribute:

(1) A new structure is proposed to compact a transaction database to facilitate an appropriate environment for efficient frequent itemset mining.
(2) A new depth first search method is developed which uses a linked list structure and a bit matrix to reduce the time and space complexity of frequent itemset mining.

The remaining of the paper is organized as follow:

In section 2, we propose some of the most former algorithms and methods for mining frequent patterns and discuss about their advantages and disadvantages. Section 3 includes the definition and description of the ABC's and presents a primary model of frequent pattern mining problem. In this section a formal definition of frequent itemset mining is proposed. Our new proposed algorithm and its flowchart are touched in section 4. Experimental results and their analysis are presented in section 5 and finally we conclude our work in section 6.

2. Related Works

Since FP-Growth algorithm was proposed, different structures have been proposed for it. Agrawal proposed an algorithm based on superficial search [5]. In 2002 H-Mine algorithm was proposed by Pei et al. [6]. Two algorithms of top-down and bottom-up were proposed by Liu et al. in 2002 and 2003 [7, 8]. In 2003, Grahne and Zhu proposed an FP-Growth algorithm based on arrays of a tree structure [9].

Mining MFIs (Maximal Frequent Itemsets) were inherent in the border notion introduced by Mannila and Toivonen in [11]. Bayardo [12] introduced MaxMiner which extends Apriori to mine only "long" patterns (maximal frequent itemsets). Since MaxMiner only looks for the maximal frequent itemsets, the search space can be reduced. MaxMiner performs not only subset infrequency pruning, where a candidate itemset with an infrequent subset will not be considered, but also a look-ahead to do superset frequency pruning. MaxMiner still needs several passes of the database to find the maximal frequent itemsets.

In [13], Burdick et al. gave an algorithm called MAFIA to mine maximal frequent itemsets. MAFIA uses a linked list to organize all frequent itemsets. Each itemset I corresponds to a bit vector; the length of the bit vector is the number of transactions in the database and a bit is set if its corresponding transaction contains I, otherwise, the bit is not set. Since all information contained in the database is compressed into the bit vectors, mining frequent itemsets and candidate frequent itemset generation can be done by bit vector AND operations. Pruning techniques are also used in the MAFIA algorithm.

GenMax, another depth-first algorithm, proposed by Gouda and Zaki [14], takes an approach called progressive focusing to do maximality testing. This technique, instead of comparing a newly found frequent itemset with all maximal frequent itemsets found so far, maintains a set of local maximal frequent itemsets. The newly found FI is only compared with itemsets in the small set of local maximal frequent itemsets, which reduces the number of subset tests.

In [15], FPmax algorithm was presented for mining MFIs using the FP-tree structure. FPmax is also a depth-first algorithm which takes advantage of the FP-tree structure so that only two database scans are needed. The experimental results in [15] showed that FPmax outperforms GenMax and MAFIA for many, although not all, cases.

Another method that uses the FP-tree structure is AFOPT [16]. In the algorithm, item search order, intermediate result representation, and construction strategy, as well as tree traversal strategy, are considered dynamically; this makes the algorithm adaptive to general situations.

SmartMiner [17], also a depth-first algorithm, uses a technique to quickly prune candidate frequent itemsets in the itemset lattice. The technique gathers "tail" information for a node in the lattice. The tail information is used to determine the next node to explore during the depth-first mining. Items are dynamically reordered based on the tail information. The algorithm was compared with MAFIA and GenMax on two data sets and the experiments showed that SmartMiner is about 10 times faster than MAFIA and GenMax.

Recently, many attempts have been given to applying bitmap techniques in the frequent patterns mining algorithms [10, 18-22]. In [10] and [18], authors have used the bitmap approach to address the problems of colossal pattern mining and parallel frequent pattern mining, respectively. BitTableFI [19] is a recently proposed efficient BitTable-based algorithm, which exploits BitTable both horizontally and vertically. Since BitTableFI may suffer from the high cost of candidate generation and test, a new algorithm Index-BitTableFI was proposed [20]. Index-BitTableFI also uses BitTable horizontally and vertically. To make use of BitTable horizontally, index array and the corresponding computing method were proposed. In [21] a fast mining algorithm was represented which integrate the merits of the matrix algorithm and Index-BitTableFI algorithm, and design an efficient algorithm for mining the frequent itemsets. CFP-Tree and CFP-Array [22] are two novel data structures which use lightweight compression techniques to reduce memory consumption of FP-tree based algorithms by an order of magnitude. CFP-Tree exploits a combination of structural changes to the FP-Tree and bitmap techniques.

In [23], a new efficient vertical top down algorithm called VTD (Vertical Top Down) is described to conduct mining of frequent itemsets in high dimensional datasets. This top down approach employed the minimum support threshold to prune the rowsets which any itemset could not be extracted from them.

3. Problem Definition

In this section we first formally define the frequent pattern mining problem and then we concentrate on FP-Growth algorithm as one the most important approaches of frequent pattern mining and explain FP-tree structure and FP-Growth method.

3.1- Frequent pattern mining problem

The frequent pattern mining problem can be defined as follows: Let $I = \{i_1, i_2, \ldots, i_n\}$ be the set of all distinct items in a transaction database, called TDB. The support of an item set α (a set of items) is the number of transactions in TDB which containing α. A k-itemset α, which consists of k items from I, is frequent if α's support is no less than a user-specified minimum support threshold which is called minsup. Given a database D and minimum support threshold minsup, the problem statement is to find the complete set of frequent itemsets in TDB which are itemsets with support of more than or equal to minsup. For example, given the dataset in table 1 and minimum support threshold minsup=3, the frequent 1-itemsets include a, b, c, d and e while f and g are infrequent because f and g occur only 2 times. Similarly, ab, ac, ad, ae, bc, bd are frequent 2-itemsets and abc is the only frequent 3-itemset of dataset.

Table1- A dataset with minimum support threshold = 3

TID	Items	Ordered frequent items
1	b, d, a	a, b, d
2	c, b, d	b, c, d
3	c, d, a, e	a, c, d, e
4	d, a, e	a, d, e
5	c, b, a	a, b, c
6	c, b, a	a, b, c
7	f, g	
8	b, d, a	a, b, d
9	c, b, a, e, f, g	a, b, c, e

A set of items X is called a closed pattern if there exists no Y such that Y is a subset of X, and Sup(X) = Sup(Y), i.e., there is no superset of X with same support. Put another way, the row set that contains the superset Y, must not be exactly the same as the row set that contains the set X. An Itemset X is called a frequent closed pattern, if it is closed and frequent.

3.2- The FP-growth algorithm and FP-tree structure

FP-growth is a well-known algorithm proposed by Han et al. [3] for frequent pattern mining. It utilizes the FP-tree (frequent pattern tree) to efficiently discover the frequent patterns. FP-tree is an extended prefix-tree structure that uses the horizontal database format and stores compressed information about patterns. It consists of one root node, a set of item prefix sub-trees as the children of the root, and a frequent-item header table. Each node of the FP-tree includes an item name, a link to the next node in the linked list of its appropriate frequent item and a count indicating the number of transactions that contains all items in the path from the root node to the current node. The header table stores the frequent items in frequency descending order. Each entry of this table includes item name, item support and a link to the head node in the linked list of its frequent item. The FP-tree is organized so that if two transactions share a common prefix, the shared part can be merged as long as the count properly reflects the occurrence of each item set in the transactions.

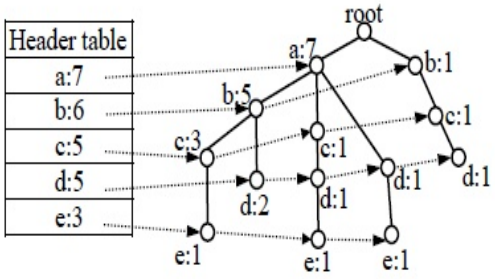

Figure 1- FP-tree constructed from the dataset in Table 1

FP-growth requires only two scans on the dataset. In the first scan, the frequency items are found to generate the header table. The dataset is re-scanned to achieve and sort the frequent items in each transaction as illustrated in the third column in Table 1. These items are inserted into FP-tree in frequency descending order. If the appropriate node of an item exists, its count is increased by one. Otherwise, a new node is inserted to the FP-tree. Figure 1 illustrates the FP-tree constructed from the dataset in Table 1.

The next step is to mine the FP-tree by constructing the conditional pattern base and then constructing the conditional FP-tree of each frequent item. The frequent items of a resulting conditional FP-tree are combined with the suffix-pattern to generate the new frequent patterns.

4. The proposed algorithm

In this section, the proposed algorithm will be presented with an example. Suppose that Table 2 shows the TDB which is used as input of frequent pattern mining problem and the minsup is set 3.

Table 2- An example TDB

TID	Items
T1	a, b, f, e, c
T2	a, b, d
T3	b, c, d, e
T4	b, c, e, f
T5	a, c, e, f
Minsup=3	

Our new proposed method work as follows:

Step 1: Making matrix Bit-TDB by using TDB

Before making matrix Bit-TDB, the items that their number is less than Minsup, should be removed from the TDB. In fact, at this stage, the first pruning is done. After the first pruning a matrix N *M should be made, where N is the number of transactions, and M is the number of items TDB.

Due to the desired TDB, M = 5, N = 5. How to initialize is so that if the number of column of an item such as x is equal to i, and this item is seen in in the transaction j, then the Bit-TDB (i, j) is equal to one, otherwise its value is zero. Due to the above TDB, Table 3 shows its Bit-TDB.

Step 2: Create a Summation array and sorting Bit-TDB by it

Summation array is a line array where its elements are sum of elements of each column in Bit-TDB matrix. In other words, it shows the number of items in the TDB. After obtaining the Summation array, due to the numbers of the Items, columns of the Bit-TDB matrix will be moved. In this case, if the summation (b) is more than others, b in Bit-TDB is in the first column, and if the Summation is equal for both items, based on the Latin alphabet will be sorted. According to the explanation given, f in Bit-TDB will be in the last column. The Bit-TDB is shown before sorting in Table 3. It is shown after sorted in table 4.

Step 3: Making a linked list

At this stage, a linked list of frequent pattern can be built gradually, so that, for each of the columns (items), we have a linked list. Primary element of each of these lists is items that are shown in the Table 4.

Table3. Unsorted Bit-TDB matrix and Summation array

Items / TID	a	b	f	e	c
1	1	1	1	1	1
2	1	1	0	0	0
3	0	1	0	1	1
4	0	1	1	1	1
5	1	0	1	1	1
Summation	3	4	3	4	4

Table4. Sorted Bit-TDB matrix and Summation array

Items / TID	b	c	e	a	f
1	1	1	1	1	1
2	1	0	0	1	0
3	1	1	1	0	0
4	1	1	1	0	1
5	0	1	1	1	1
Summation	4	4	4	3	3

At this stage, a logical AND between each column (item), like column with next columns, b c, e, a, f is done, and the column matrix can be made. In description of the algorithm, item b, the first item and each of items c, e, a, f, can be considered as the second item. For example, in this instance, for the first item b, column matrices and ((bc, be, ba, bf)) is obtained. Then, for each of the "AND matrix", the sum of elements is computed, and, if the sum is greater than or equal minsup, the second item will be added to a list which the first item is the beginning member. Creating a frequent pattern linked list of the example cited by forming "AND matrices", step by step will be shown. The first item is b, Then according to the above conditions, If an item has items qualify for inclusion in the list, will be listed in a list in which the first element is item b:

According to figure 2, since $\sum_{k=0}^{n}(bc)_i = 3 \geq \text{Minsup}$, then item c will be put in the list with start of b.

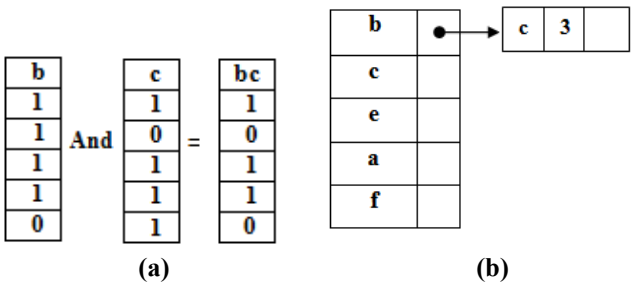

Figure 2- (a) Checking bc pattern with minsup. (b) Fp-Linked list after checking

According to figure 3, since $\sum_{k=0}^{n}(be)_i = 3 \geq$ Minsup, then, item e will be put in the list with start of b.

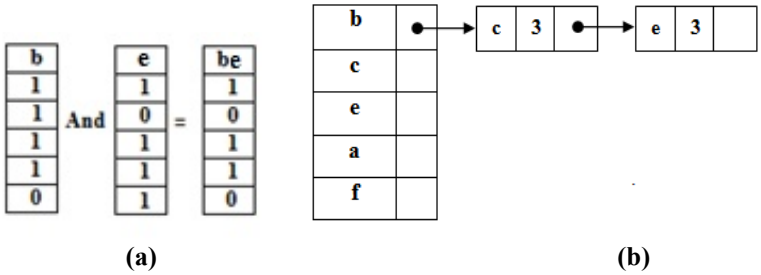

Figure 3- (a) Checking be pattern with minsup. (b) Fp-Linked list after checking

According to figure 4, since $\sum_{k=0}^{n}(ba)_i = 2 \ngeq$ Minsup, then item a will not be listed at the end of any list.

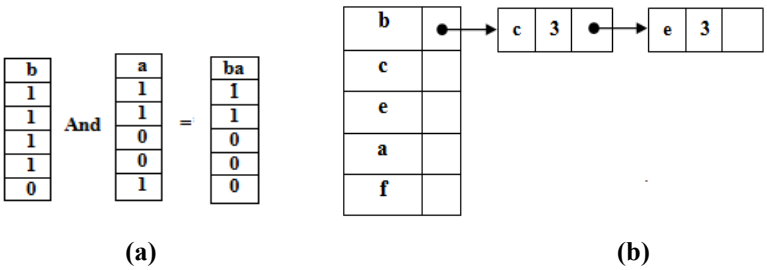

Figure 4-(a) Checking ba pattern with minsup. (b) Fp-Linked list after checking

According to figure 5, since $\sum_{k=0}^{n}(bf)_i = 2 \ngeq$ Minsup, then item f will not be listed at the end of any list.

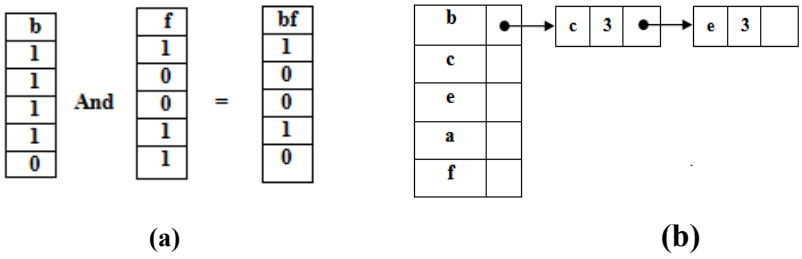

Figure 5-(a) Checking bf pattern with minsup. (b) Fp-Linked list after checking

The first next item is c. according to the above conditions, if an item qualifies for inclusion in the list, will be listed in a list in which the first element is item c.

According to figure 6, since $\sum_{k=0}^{n}(ce)_i = 4 \geq$ Minsup, Then, item e will be put in the list with start of c.

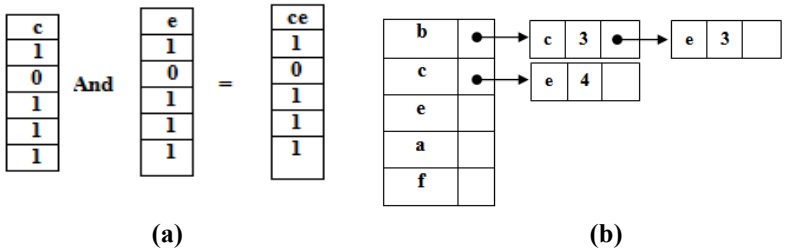

Figure 6- (a) Checking ce pattern with minsup. (b) Fp-Linked list after checking

According figure 7, since $\sum_{k=0}^{n}(ca)_i = 2 \ngeq$ Minsup, then item a will not be listed at the end of any list.

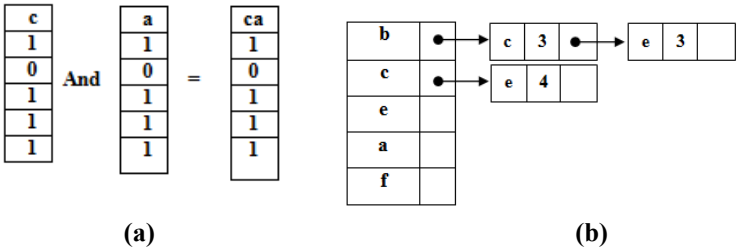

Figure 7- (a) Checking ca pattern with minsup. (b) Fp-Linked list after checking

According to figure 8, since $\sum_{k=0}^{n}(cf)_i = 3 \geq$ Minsup, Then, item f will be listed in the of the list with start of c.

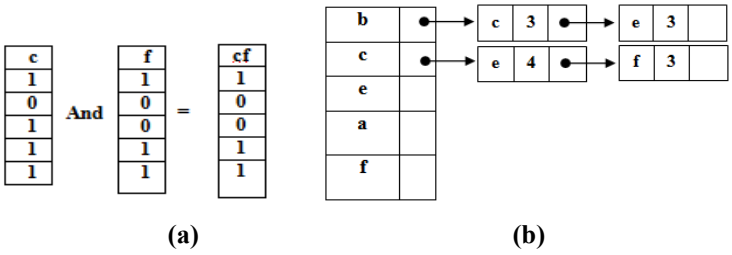

Figure 8- (a) Checking cf pattern with minsup. (b) Fp-Linked list after checking

The first next item is e. according to the above conditions, if an item qualifies for inclusion in the list, will be listed in a list in which the first element is item e.

According to figure 9, since $\sum_{k=0}^{n}(ea)_i = 2 \ngeq$ Minsup, Then item e will not be listed at the end of any list.

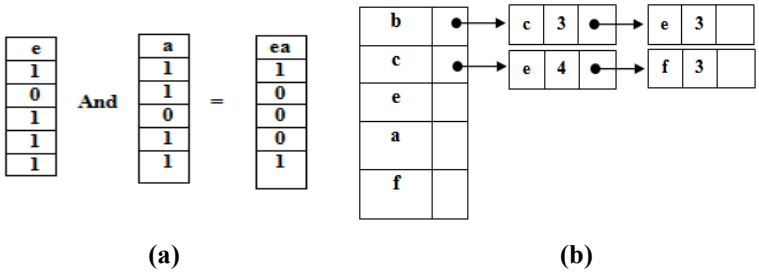

Figure 9- (a) Checking ea pattern with minsup. (b) Fp-Linked list after checking

According to figure 10, since $\sum_{k=0}^{n}(ef)_i = 3 \geq$ Minsup, then item f will be listed in the tail of the list of e.

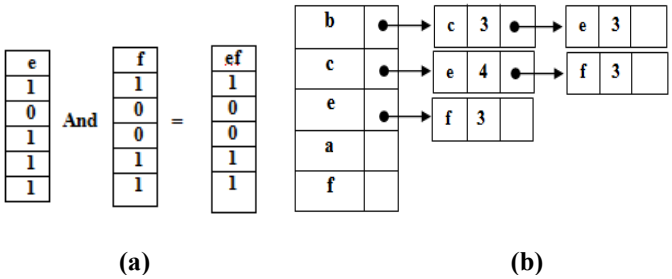

Figure 10- (a) Checking ef pattern with minsup. (b) Fp-Linked list after checking

The next item is a. according to the above conditions, if an item qualifies for inclusion in the list, will be listed in a list in which the first element is item a.

According to figure 11, since $\sum_{k=0}^{n}(af)_i = 2 \not\geq$ Minsup, then item f will not be listed at the end of any list.

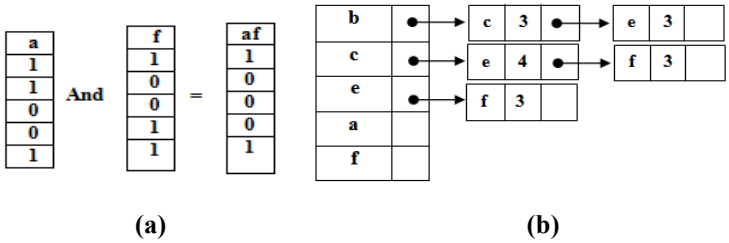

Figure 11- (a) Checking af pattern with minsup. (b) Fp-Linked list after checking

Finally, the last item is f. there is no column after that in Bit-TDB. Therefore, the second item does not exist, and the final list remains without any change. Figure 12 shows the final FP-Linked list.

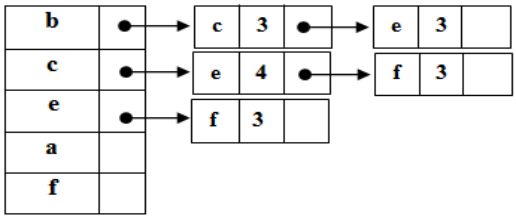

Figure 12- Fp-Linked list

The flowchart of figure 13 shows the architecture of our algorithm.

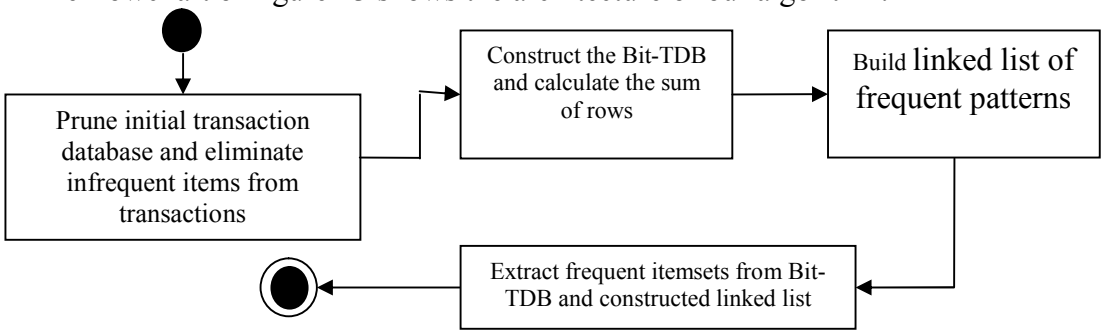

Figure 13- The flowchart of proposed algorithm

5. Experimental Results

In this section we will study the performance of our algorithm. All the reported runtime include both computation time and IO time. AIM [24] has already shown its better performance than other FP-tree based algorithms. So we compare our algorithm with AIM algorithm.
We use 3 real standard datasets from FIMI [25] to compare the algorithms. Figure 14 shows some statistical information about the dataset used for performance study.

Table 5- Characteristics of test datasets

datasets	size	Transactions#	Items#
mushroom	0.56M	8124	119
pumsb	16.3M	49046	2113
Retail	3.97M	88162	16469

Figures 14-16 compare execution time of our new proposed algorithm with best algorithms for a standard dataset. The best algorithm for the standard Chess dataset is Aim Algorithm.

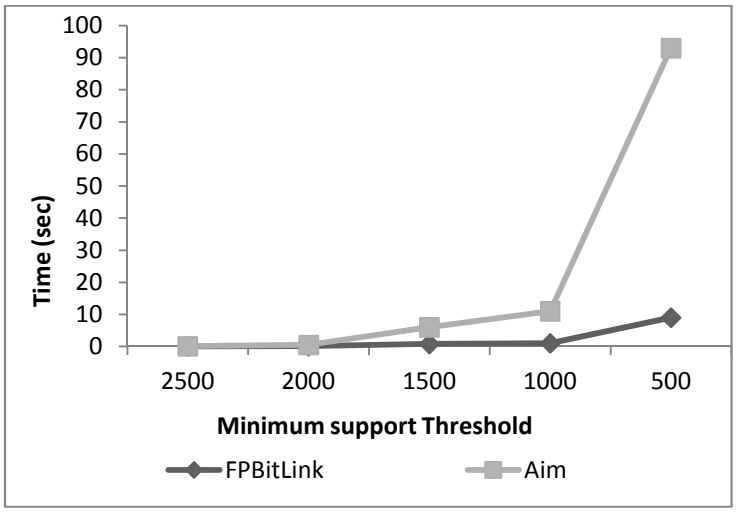

Figure 14- Dataset mushroom

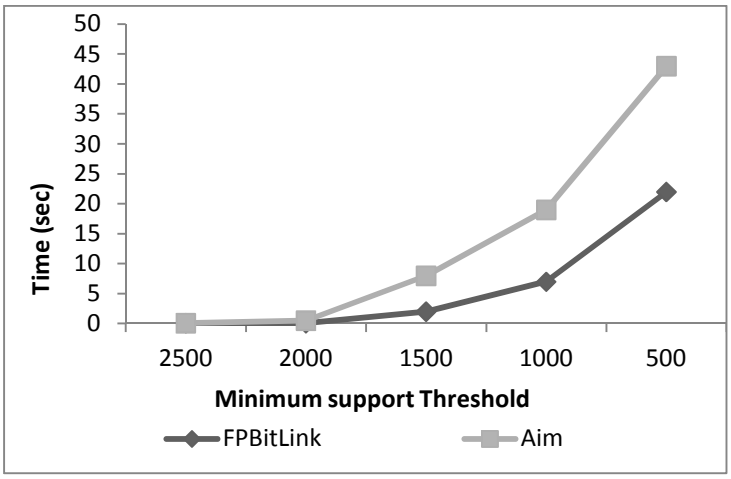

Figure 15- Dataset pumsb

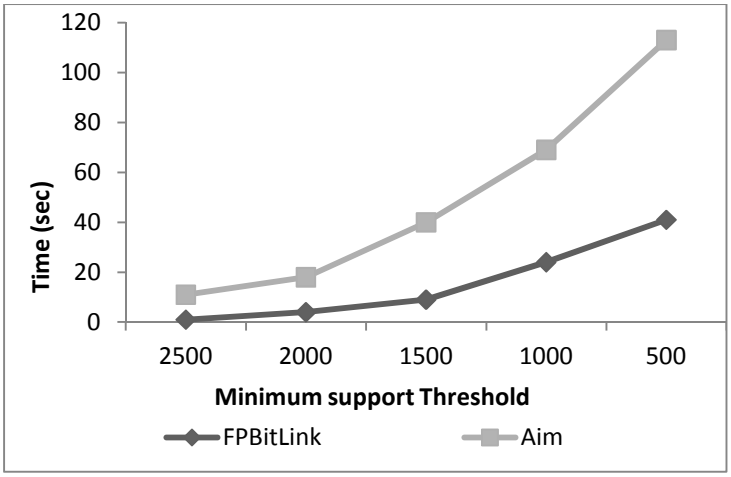

Figure 16- Dataset retail

According to the diagrams, if the minsup value is relatively large, FPBitLink will perform much better than existing algorithms because the large number of items will be pruned at the beginning and Bit-TDB matrix becomes small therefore in this case the number of And operator and items at the beginning of the list would be decreased.

6. Conclusion

In this paper, we have investigated frequent pattern mining problem in transaction databases. Most of the reviewed techniques have been practiced based on FP-Growth algorithm's idea in mind and they try to optimize it. Actually, FP-Growth is learned as the basis for many depth first search horizontal pattern mining algorithms. A FP-tree structure is used to compact the transaction database and make the mining process more efficient in FP-Growth. In order to improve the time and space complexity of mining algorithm, we introduced a new mining algorithm based on a linked list structure using a bit wise approach to compact the dataset. Represented experimental results showed that our new algorithm outperformed the former algorithms.

References

[1] Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993ACM-SIGMOD, Washington, DC, pp 207–216.

[2] Agrawal R. and Srikant R. "Fast algorithms for mining association rules." In Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94), pages 487–499, Santiago, Chile, Sept. 1994.

[3] Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceeding of the 2000 ACM-SIGMOD, Dallas, TX, pp 1–12.

[4] Zaki MJ, Parthasarathy S, Ogihara M, Li W (1997) Parallel algorithm for discovery of association rules. data mining knowl discov, 1:343–374.

[5] Agarwal R, Aggarwal CC, Prasad VVV (2001) A tree projection algorithm for generation of frequent item-sets. J Parallel Distribut Comput 61:350–37.1

[6] J. Pei "Pattern-Growth Methods for Frequent Pattern Mining" PhD Thesis, 2002.

[7] Liu J, Pan Y, Wang K, Han J (2002) Mining frequent item sets by opportunistic projection. In: Proceeding of the 2002 ACM SIGKDD, Edmonton, Canada, pp 239–248.

[8] Liu G, Lu H, Lou W, Yu JX (2003) On computing, storing and querying frequent patterns. In: Proceeding of the 2003 ACM SIGKDD, Washington, DC, pp 607–612.

[9] Grahne G, Zhu J (2003) Efficiently using prefix-trees in mining frequent itemsets. In: Proceeding of the ICDM'03 international workshop on frequent itemset mining implementations (FIMI'03), Melbourne, FL, pp 123–132.

[10] Mohammad Karim Sohrabi, Ahmad Abdollahzadeh Barforoush (2012) Efficient colossal pattern mining in high dimensional datasets, Published in: Knowledge-Based Systems Elsevier Science Publishers B. V. Amsterdam, The Netherlands, The Netherlands, pp 41-52.

[11] H. Mannila and H. Toivonen, "Levelwise Search and Borders of Theories in Knowledge Discovery," Data Mining and Knowledge Discovery, vol. 1, no. 3, pp. 241-258, 1997.

[12] R.J. Bayardo, "Efficiently Mining Long Patterns from Databases," Proc. ACM-SIGMODInt'l Conf.Management of Data, pp.85-93,1998.

[13] D. Burdick, M. Calimlim, and J. Gehrke, "MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases," Proc. Int'l Conf. Data Eng., pp. 443-452, Apr, 2001.

[14] K. Gouda and M.J. Zaki,(2001), "Efficiently Mining Maximal Frequent Itemsets," Proc. IEEE Int'l Conf. Data Mining, pp. 163-170.

[15] G. Grahne and J. Zhu, "High Performance Mining of Maximal Frequent Itemsets," Proc. SIAM Workshop High Performance Data Mining: Pervasive and Data Stream Mining, May, 2003.

[16] G. Liu, H. Lu, J.X. Yu, W. Wei, and X. Xiao, "AFOPT: An Efficient Implementation of Pattern Growth Approach," Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, CEUR Work- shop Proc., vol. 80, Nov,2003.

[17] Q. Zou, W.W. Chu, and B. Lu, "SmartMiner: A Depth First Algorithm Guided by Tail Information for Mining Maximal Frequent Itemsets," Proc. IEEE Int'l Conf. Data Mining, Dec, 2002.

[18] Mohammad Karim Sohrabi, Ahmad Abdollahzadeh Barforoush (2013) Parallel frequent itemset mining using systolic arrays, Published in: Knowledge-Based Systems Elsevier Science Publishers B. V. Amsterdam, The Netherlands, The Netherlands, pp 462-471, January, 2013.

[19] J. Dong, M. Han, BitTableFI: an efficient mining frequent itemsets algorithm, Knowledge Based Systems 20 (4) (2007) 329–335.

[20] W. Song, B. Yang, Z. Xu, Index-BitTableFI: an improved algorithm for mining frequent itemsets, Knowledge Based Systems 20 (4) (2007) 329–335.

[21] Z. Xu, D. Gu, S. Wei, An Efficient Matrix Algorithm for Mining Frequent Itemsets, (2009) International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009.

[22] Schlegel B., Gemulla R., Lehner w. (2011) Memory-Efficient Frequent Itemset Mining, in: Proc. 14th International Conference on Extending Database Technology (EDBT), 2011.

[23] Sohrabi M K, Ghods V, Top-down vertical itemset mining, Proceedings of the SPIE 9443 sixth International Conference on Graphic and Image Processing (2014)

[24] A. Fiat, S. Shporer. AIM: Another Itemset Miner. In IEEE ICDM'03 Workshop FIMI'03, Melbourne, Florida, USA, 2003.

[25] Frequent Itemset Mining Implementations Repository: http://fimi.cs.helsinki.fi/