Int. J. Advance Soft Compu. Appl, Vol. 10, No. 2, July 2018
ISSN 2074-8523

An Enhanced Accelerator Frequent Pattern Growth for Association Rules Mining

Mokhtar Al-Hamadi[1]*, Mossab Ghillan[2] , Faisal Saeed[3,4]

[1]Faculty of Computer and Information Technology, Yemen Academy for Graduate Studies, Yemen
e-mail: moktaralhamadi@gmail.com
[2]Faculty of Computer and Information Technology, Sana'a University, Yemen
e-mail: mghilan@gmail.com
[3]College of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia
[4]Information Systems Department, Faculty of Computing, Universiti Teknologi Malaysia, Johor, Malaysia
e-mail: fsaeed@taibahu.edu.sa

Abstract

The Association Rule Mining (ARM) a data mining technique that plays an important rule in the Knowledge Discovery Databases (KDD). In the literature, a noticeable number of algorithms have attempted to discover the knowledge from the database, such as Frequent Pattern Growth (FP-growth). However, many experimental results have shown that the efficiency of building conditional FP-trees during mining big datasets is very high in terms of computational time and space. Although some researchers have tried to mitigate these issues, the issues still exist and need to be resolved. Therefore, the aim of this work is to introduce an efficient mining frequent patterns algorithm based on FP-growth algorithm, which is called an Accelerator Frequent Pattern Growth (AFP-growth) algorithm. The main idea of the AFP-growth is to avoid building the conditional FP-trees, which in turns, will increase the efficiency of AFP-growth. The experimental results showed that the proposed algorithm reduces the time and space during mining frequent itemsets from database.

Keywords: *Association rules mining, FP-Growth Algorithm, Itemset, Frequent pattern, Frequent Itemsets Table (FIT).*

1 Introduction

Association rules mining (ARM) is an important technique of data mining that focuses on the process of finding the interesting associations or correlation relationships between itemsets in big data [1],[2]. Association rules mining are also aiming at finding strong association rules. The process of ARM includes two steps, finding frequent itemsets and generating association rules based on the discovered itemsets. The first step requires extensive computation time and storage and thus becomes a challenging problem in ARM [3]. There are a large number of algorithms to extract frequent patterns. Despite the success surrounding these algorithms, three main and well-known algorithms that are considered in this study, namely Apriori, FP-Growth, and Eclat[4]. Apriori algorithm [5],[6] was proposed by Agarwal in 1994, it is based on superficial search. This algorithm suffers from some weaknesses such as: scanning the whole database that includes many transactions repeatedly and thus it consumes a lot of memory and CPU time; also it generates huge candidate sets [3],[7]. On the other hand, Frequent Pattern growth (FP-Growth) [8],[9] provide a better solution to these weaknesses and limitations.

A new divide and conquer strategy was adopted by FP-growth algorithm so that it only require two database scans and does not generate any candidates [3],[10, 11]. However, in the FP-Growth algorithm a lot of time is required to build FP-tree and conditional FP-tree[12]. In this paper, a new algorithm, which is called an Accelerator Frequent Pattern Growth (AFP-growth) algorithm, was introduced to improve the efficiency of the mining process by avoiding building the conditional FP-trees, which helped to The paper is organized as follows: Section 2 discusses the related work, the methods are described in Section 3. The experimental results and discussion are included in Section 4, while Section 5 concludes the findings of this paper.

2 Related Work

As discussed earlier, FP-Growth algorithm has some limitations, such as it is not scalable in mining big databases. To solve this problem, there are many structures that have been proposed [13]. For instance, a new method was proposed in [14], which is known as memory-based hyper structure, H-struct or Hyperlinked structure for mining purpose. In this method, conditional FP-tree is not generated by H-mine in order to improve the efficiency of mining process by saving more time and space. In [15] , FPmax algorithm was presented to mine Maximal Frequent Itemsets based on the FP-tree structure. This algorithm is a depth-first and it requires only two database scans. The experimental results in [15] showed that FPmax performance is better than GenMax and MAFIA. In addition,

Ascending Frequency Order Prefix Tree (AFOPT) also uses the FP-tree structure [16],[17]. In AFOPT, the conditional database is represented by the proposed compressed data structure and the top-down strategy was adopted, which can minimize the total number of conditional databases. The efficiency of AFOPT was bette than FP-growth, as shown in the experimental results. Furthermore, in [18] two data structures were proposed which are CFP-Array and CFP-Tree that use compression method and the order of magnitude for reducing the memory consumption of FP-tree based method. The CFP-Tree takes the advantage of combining the structural changes to bitmap and FP-Tree methods. According to [1], Painting-Growth algorithm and N (not) Painting-Growth algorithm build two-item permutation sets to get the association sets for all frequent items. Both methods require only one scan for the database to obtain the results of mining process. In addition, Sohrabi and Marzooni [19] proposed FP-Linked List Algorithm that used bit matrix and linked list structure based on FP-Growth algorithm.

3 Problem Formulations or Methodology

3.1 Frequent patterns

Frequent patterns are patterns that appear frequently in a dataset. Let I= $\{i_1, i_2, ...,i_n\}$ as a collection of the whole different items in the database, each transaction T is a subset of I, that is, $T \subseteq I$, and database D is a collection of transactions. An itemsets $X \subseteq I$ is a subset of items. A transaction T = (tid; X) is a tuple; X is an itemsets and tid is the transaction-id. The sup(X), which refers to the support of an itemsets X in transaction database TDB, can be calculated as the number of transactions in TDB that contain X, i.e., sup(X) =count(X)/N. The number of items in an itemsets is considered the dimension or length of this itemset; if the length of the itemsets is k, the itemsets is called k-itemsets.

For the support threshold *minsup*, which is a user-specific, X is considered as a frequent itemset if $sup(X) \geq minsup$. The problem here is finding the all frequent patterns in TDB using specific *minsup*. Therefore, The Frequent Itemsets Mining (FIM) requires more computational time than the rules generation. So majority of the literatures concentrate on designing fast and scalable algorithms for mining frequent itemsets [14] ,[20],[21].

3.2 Motivation scenario

The transaction database in Table 1 shows the TDB, which is used in this paper as input of frequent pattern mining problem and minsup set as 3.

Table1: A transaction database

TID	*Items Bought*	*(Ordered) Frequent Items*
100	f , a, c, d, g, I, m, p	c, f, a, m, p
200	a, b, c, f, l, m, o	c, f, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	C, f, a, m, p

3.2.1 FP-growth algorithm

In this algorithm, two scans only are required on the dataset. We can obtain a set of frequent items with their support count when the database is scanned for the first time. The collection of frequent items is ordered by decreasing sequence of support count; then the header table is generated. In in each transaction, for sorting the frequent items, the dataset is re-scanned as shown in Table 1 (thir column).

For instance, if the transaction database (DB) is represented in Table 1, with minsup = 3. As shown in Table 1, there are five transctions, and each transaction contains some alphabets. The list of frequent items are computed and generated by FP-growth algorithm, and then the infrequent items are removed and the transaction are sorted in descending order. The remaining iterms in this example are: f, c, a, b, m, and p. For the transaction with ID 100 (TID 100), the items sets were pruned and reordered from {f,a,c,d,g,i,m,p} to{c,f,a,m,p}. Using these transactions, the FP-tree was bulit, and its root is NULL, while the tree nodes refer to the items. The transactions that have similar prefix are represented in one path of the FP-tree. Each node has a count number which refer to the number of transactions represented by the portion of the path reaching this node. The frequent items are stored in the The Header Table in descending order, and the nodes are linked to other nodes that have the same lable in the FP-tree. The FP-tree of Table1 is shown in Fig 1.

Item	*frequent*
c	4
f	4
a	3
b	3
m	3
p	3

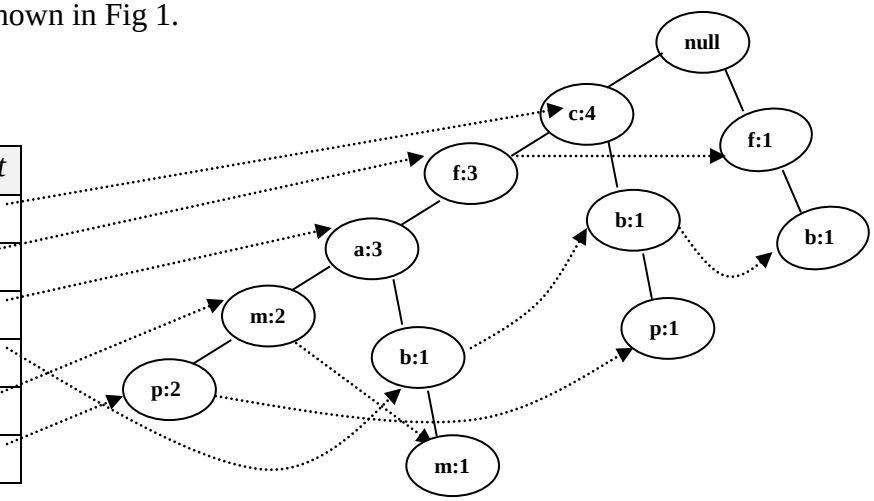

Fig.1 The FP-tree of Transaction Dataset in Table 1

After building the FP-tree, it will be mined by the FP-growth for each item included in the Header Table, starting from the end to the beginning of this table. For instanced, "p" item will be mined first as shown in Fig 1 and Table 2. After that, the prefixed path of each item is picked out in order to find its conditional pattern base. The conditional FP-tree are built using conditional pattern base and the frequent patterns are mined by pattern fragment growth. The conditional FP-tree is bulit using the same method of building FP-tree.

Table2: Dig FP-tree through creating conditional sub-pattern base.

item	*Conditional pattern base*	*Conditional FP-tree*	*Frequent pattern*
p	{(c,f,a,m:2),(c,b:1)}	{c:3}	pc:3
m	{(c,f,a:2),(c,f,a,b:1)}	{c:3,f:3,a:3,cf:3,ca:3,fa:3 ,cfa:3}	cm:3,fm:3,am:3,cfm:3, cam:3,fam:3, cfam:3
b	{(c,f,a:1),(c:1),(f:1)}	----------	----------
a	{(c,f:3)}	{c:3,f:3,cf:3}	ca:3,fa:3,cfa:3
f	{(c:3)}	{c:3}	Cf:3

4 The Proposed Method

It is known that FP-Growth algorithm requires scanning database twice, which improves the efficiency of mining the frequent patterns. This paper proposed a new algorithm which is called **A**ccelerator **F**requent **P**attern Growth (AFP growth), which scans the database only once to obtain the mining results. AFP algorithm uses binary matrix and Frequent Itemsets Table (FIT) in order to get all frequent itemsets.The new proposed algorithm work as follows:

1- The algorithm (AFP-growth) scans the database once to create a binary matrix, as follows: if $I_j \in D_i$, we represent that in the binary matrix by sitting matrix(i,j) to 1. Unless 0.
2- In the second step, the items with frequency number less than minsup *are deleted* from binary matrix. At this stage, the pruning is done.
3- In the third step, a combination algorithm is used to generate patterns as following:

a. Add the first column in the binary matrix to Frequent Itemsets Table (FIT).
b. Apply a logical AND between second column (b) in the binary matrix with all columns (a) that are existing in Frequent Itemsets Table (FIT) if frequency >= minsup, then add new column(ab) to Frequent Itemsets Table (FIT). After that add this column (b) to Frequent Itemsets Table (FIT).

c. Go to next column (c) in binary matrix and apply a logical AND between them and all existing columns in (FIT). If the frequency >= minsup, then add new column to Frequent Itemsets Table (FIT) and also add this column to frequent itemsets table.
d. Repeat the previous steps for all columns in binary matrix.
e. The Frequent Itemsets Table (FIT) will finally contain all frequent itemsets.

Using the transaction database in Table 1 as an example, the AFP algorithm scans the database once, and generate binary-TDB, if the number of column for one item such as x is equal to i, and this item is seen in the transaction j, then the TDB (i , j) is equal to one, otherwise its value is zero. The items with frequency number less than minsup should be removed from the TDB. In fact, at this step, the first pruning is done.

Table 3 shows the binary-TDB Matrix.

TID	*a*	*b*	*c*	*f*	*m*	*p*
100	1	0	1	1	1	1
200	1	1	1	1	1	0
300	0	1	0	1	0	0
400	0	1	1	0	0	1
500	1	0	1	1	1	1
frequency	3	3	4	4	3	3

The second step of the proposal algorithm is to create Frequent Itemsets Table (**FIT**). The first column **a** in binary matrix is considered a frequent and added column to (FIT).

a
1
1
0
0
1

Fig. 2 Frequent Itemsets Table (FIT) in first step.

For the second column b in the binary matrix, a logical AND operation is applied between this column (item) and all previous columns in FIT, for example, for all previous columns in FIT (a), the AND logic is done with b. According to Fig. 3: $\sum$ (ab) = 1 $\not\geq$ minsup, then set ab as not frequent itemsets and therefore it will not be add to FIT.

a
1
1
0
0
1

AND

b
0
1
1
1
0

=

ab
0
1
0
0
0

Fig. 3 Checking (ab) pattern with minsup=3

According to Fig. 3, the Frequent Itemsets Table becomes:

A	*b*
1	*0*
1	*1*
0	*1*
0	*1*
1	*0*

Fig. 4 Frequent Itemsets Table (FIT) in second step

For the third column **c** in the binary matrix, the logical AND operation is applied again between this column (item), and all previous columns in **FIT**, which are **a** and **b**. The AND logic operation between **a** AND **c is** shown in Fig. 5. The $\sum$ (ac) = 3 $\geq$ minsup, then, new column (**ac**) is added to (**FIT**).

a
1
1
0
0
1

AND

c
1
1
0
1
1

=

ac
1
1
0
0
1

Fig. 5 Checking (ac) pattern with minsup=3

But $\sum$ (bc) = 2 $\ngeq$ minsup, in this instance the FIT becomes:

a	*B*	*ac*	*c*
1	*0*	*1*	*1*
1	*1*	*1*	*1*
0	*1*	*0*	*0*
0	*1*	*0*	*1*
1	*0*	*1*	*1*

Fig. 6 Frequent Itemsets Table (FIT) in third step

Similarity, for the fourth column **f** in the binary matrix, the logical AND operation is applied between this column (item), and all previous columns in FIT. In this case, the previous columns in FIT are *a*, *b*, *ac* and *c*. So AND is applied between **f** and (*a*, *b*, *ac*, *c*).

According to Fig. 7, the $\sum$ (af) =3 $\geq$ minsup, then, a new column (af) is added to the Frequent Itemsets Table (FIT).

a
1
1
0
0
1

AND

f
1
1
1
0
1

=

af
1
1
0
0
1

Fig. 7 Checking (af) pattern with minsup=3

but $\sum$ (bf) = 2 $\not\geq$ minsup, and according to Fig. 8, $\sum$ (acf) = 3 $\geq$ minsup and $\sum$ (cf) = 3 $\geq$ minsup. Then, new columns (acf) and (cf) are added to (**FIT**).

ac
1
1
0
0
1

AND

f
1
1
1
0
1

=

acf
1
1
0
0
1

Fig. 8 Checking (acf) pattern with minsup=3

in this instance the Frequent Itemsets Table **FIT** become.

a	*b*	*ac*	*C*	*af*	*acf*	*Cf*	*f*
1	*0*	*1*	*1*	*1*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*	*1*	*1*
0	*1*	*0*	*0*	*0*	*0*	*0*	*1*
0	*1*	*0*	*1*	*0*	*0*	*0*	*0*
1	*0*	*1*	*1*	*1*	*1*	*1*	*1*

Fig. 9 Frequent Itemsets Table (FIT) in firth step

For the fifth column **m** in the binary matrix, the logical AND operation is applied between this column (item), and all previous columns in **FIT**, which are *a, b, ac, c, af, acf, cf* and *f*.
After applying AND operation, the **FIT** is generated and shown in Fig. 10.

a	*b*	*ac*	*C*	*af*	*acf*	*cf*	*F*	*am*	*acm*	*Cm*	*afm*	*acfm*	*cfm*	*fm*	*m*
1	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
0	*1*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
1	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*

Fig. 10 Frequent Itemsets Table (FIT) in fifth step

Finally, for column **p** in the binary matrix, the logical AND operation is applied between this column (item), and all previous columns in **FIT**, which are *a, b, ac, c, af, acf, cf, f, am, acm, cm, afm, acfm, cfm, fm* and *m*. After applied and logical the **FIT** becomes.

a	*b*	*Ac*	*c*	*af*	*acf*	*cf*	*F*	*am*	*acm*	*Cm*	*afm*	*acfm*	*cfm*	*fm*	*M*	*cp*
1	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*0*
0	*1*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*
1	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*

Fig. 11 Frequent Itemsets Table (FIT) in finally step

According to Fig. 11, the frequent itemsets mining (FIM) are:

1. Tow frequent itemsets are: ac:3, af:3, cf:3, am:3, cm:3, fm:3, cp:3
2. Three frequent itemsets are: acf:3, acm:3, afm:3, cfm:3,
3. Four frequent itemsets are: acfm:3

5 Results, Analysis and Discussions

In this section, we evaluate the performance of the proposed algorithm. We compared the performances of the proposed algorithm with FP-Growth algorithm. The experiments were performed on a 2.53 GHz Intel Processor with 4 GB memory, and run with the Windows 7, 32-bit operating system. The Java language, in NetBeans IDE 8.0 development environment was used to implement the proposed algorithm. Standard datasets (Mushroom, Retail, Pumsb, Census, T10I4D100K) were used, which were taken from FIMI(http://fimi.cs.helsinki.fi/).

Table 5 Characteristics of test datasets

datasets	*size*	*Transactions*	*Items*
Census	11 M	3,196	76
Mushroom	M0.56	8124	119
Retail	M3.97	88162	16469
T10I4D100K	3.93M	100000	870
pumsb	16.30M	49046	2113

Fig. 11-15 show that the performance of the proposed algorithm outperformed the standard mining method (FP-Growth) in terms of the computational time using all datasets for different minimum support thresholds.

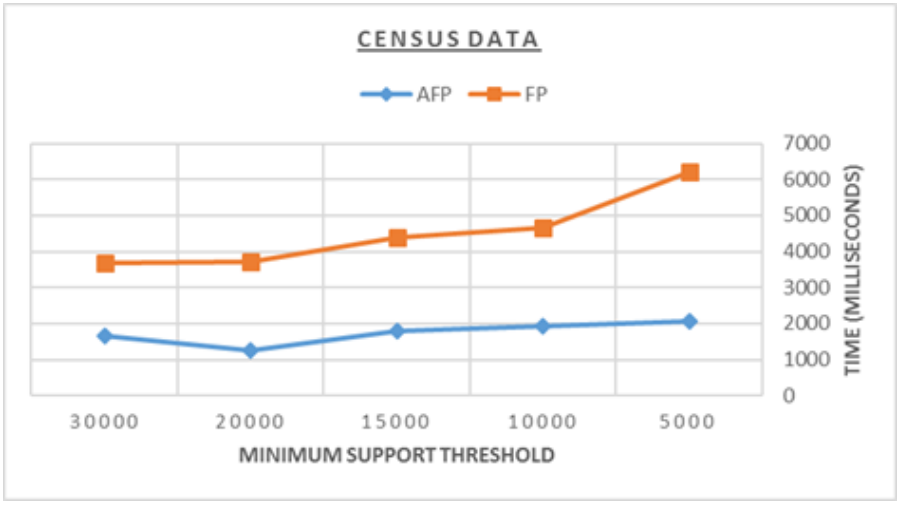

Fig. 11 census dataset

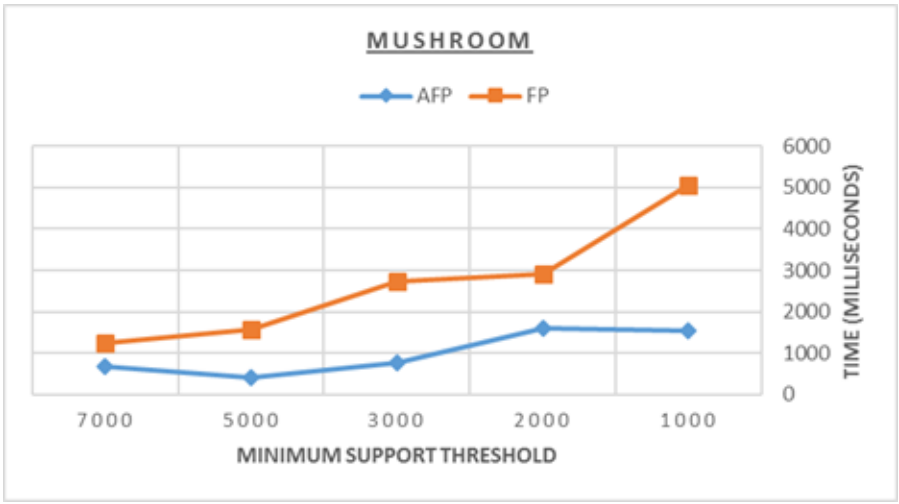

Fig. 12 Mushroom dataset

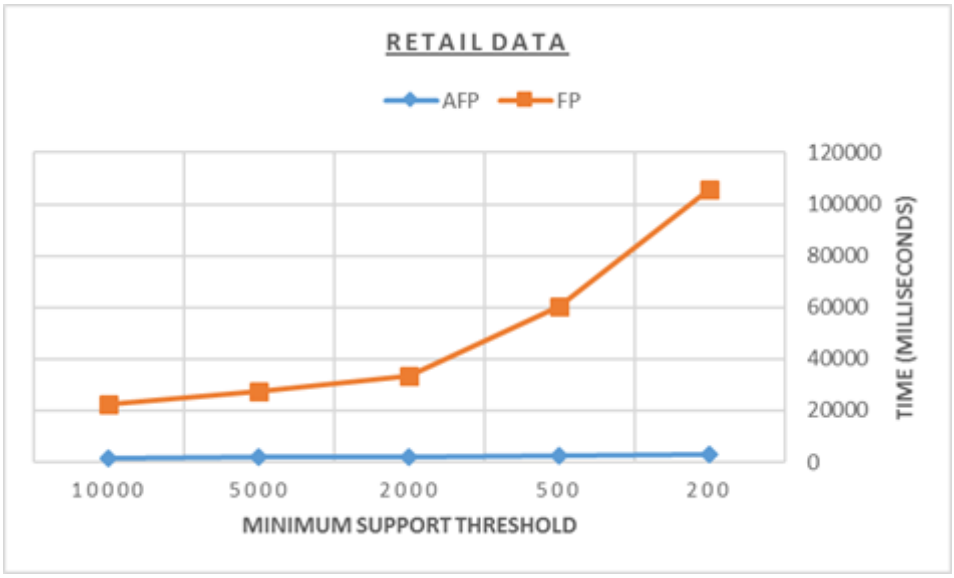

Fig. 13 Retail dataset

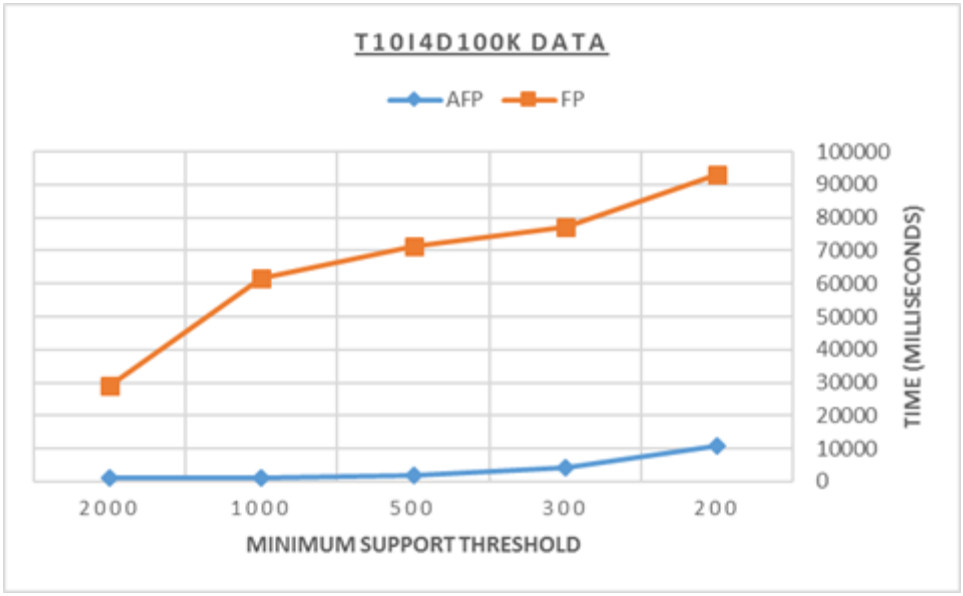

Fig. 14 T10I4D100Kdataset

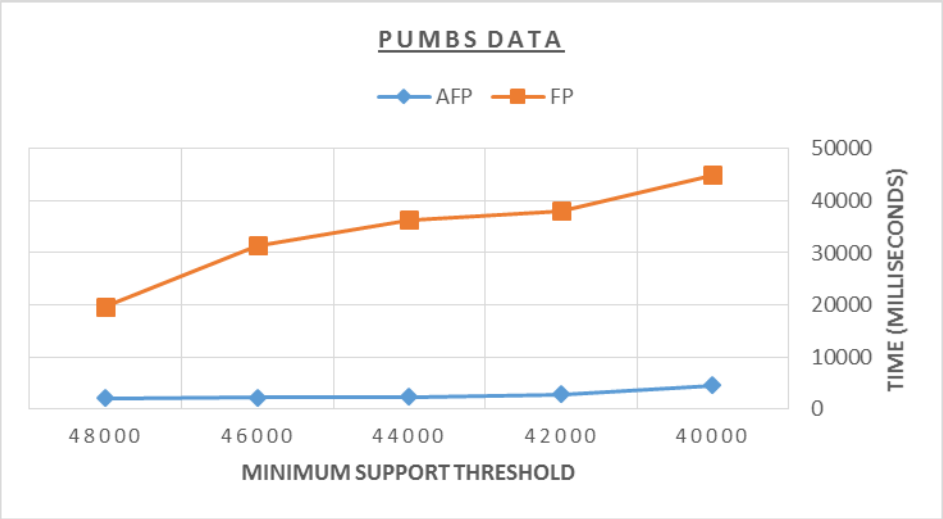

Fig. 15 Pumbs dataset

For each dataset, the experiment was conducted 30 times, and the average of the execution time was reported in Figures 11-15. Each data set used different values of minimum support.

As shown in Figures 13 and 15, when the transactions length is long, the data is considered as massive and the number of items is large. Therefore, FP-growth generates many tree branches which requires more memory and increases the time computation. This makes the performance of FP-Growth is not efficient when massive data are used. The results showed that the proposed method, AFP, has a better scalability.

As shown in the experimental results of T10I4D100K dataset and Pumbs dataset in Fig. 14-15, when the minimum support degree increases, the AFP-Growth algorithm runs much faster than the FP-Growth algorithm.

6 Conclusion

FP-Growth algorithm requires scanning database twice, which improved the efficiency of mining the frequent patterns comparing to Apriori algorithm and other methods. However, it still consuming more computational time due to building the conditional FP-trees. This paper proposed a new algorithm which is called Accelerator Frequent Pattern Growth (AFP Growth), which scans the database only once to obtain the mining results. AFP algorithm uses binary matrix and Frequent itemsets Table (FIT) in order to get all frequent itemsets. The experimental results showed that the performance of the enhanced algorithm outperformed the standard FP-growth algorithm.

Acknowledgement

This work is supported by the Ministry of Higher Education (MOHE) and the Research Management Centre (RMC) at the Universiti Teknologi Malaysia (UTM) under the Research University Grant Category (VOT Q.J130000.2528.16H74).

References

[1]. Zeng, Y., et al., *Research of improved FP-Growth algorithm in association rules mining.* Scientific Programming, 2015. **2015**: p. 6.

[2]. Gole, S. and B. Tidke. *Frequent Itemset Mining for Big Data in social media using ClustBigFIM algorithm*. in *Pervasive Computing (ICPC), 2015 International Conference on*. 2015. IEEE.

[3]. Zhou, L. and X. Wang, *Research of the FP-growth algorithm based on cloud environments.* Journal of Software, 2014. **9**(3): p. 676-683.

[4]. Dhinakaran, D. and J.P. PM, *A Study on Data Mining: Frequent Itemset Mining Methods Apriori, FP growth, Eclat.* 2017.

[5]. Agrawal, R. and R. Srikant. *Fast algorithms for mining association rules*. in *Proc. 20th int. conf. very large data bases, VLDB*. 1994.

[6]. Agrawal, R., T. Imieliński, and A. Swami. *Mining association rules between sets of items in large databases*. in *Acm sigmod record*. 1993. ACM.

[7]. Badhe, V. and P. Richharia, *A Survey on Association Rule Mining for Finding Frequent Item Pattern.* 2016.

[8]. Addi, A.-M., A. Tarik, and G. Fatima. *Comparative survey of association rule mining algorithms based on multiple-criteria decision analysis approach*. in *Control, Engineering & Information Technology (CEIT), 2015 3rd International Conference on*. 2015. IEEE.

[9]. Bala, A., et al., *Performance Analysis of Apriori and FP-Growth Algorithms (Association Rule Mining).* International Journal of Computer Technology and Applications, 2016. **7**(2): p. 279-293.

[10]. Alghyaline, S., J.-W. Hsieh, and J.Z. Lai, *EFFICIENTLY MINING FREQUENT ITEMSETS IN TRANSACTIONAL DATABASES.* Journal of Marine Science and Technology, 2016. **24**(2): p. 184-191.

[11]. Nigam, B., A. Nigam, and P. Dalal, *Comparative Study of Top 10 Algorithms for Association Rule Mining.* 2017.

[12]. Shukla, V.S. and M.S. Itkar, *Improving Association Rule Mining By Defining A Novel Data Structure.* 2017.

[13]. Liu, Y., *Research on Association Rules Mining Algorithm Based on Large Data.* Revista de la Facultad de Ingeniería, 2017. **32**(8).

[14]. Pei, J., *Pattern-growth methods for frequent pattern mining*. 2002, Citeseer.

[15]. Grahne, G. and J. Zhu. *High performance mining of maximal frequent itemsets*. in *6th International Workshop on High Performance Data Mining*. 2003.
[16]. Liu, G., et al. *AFOPT: An Efficient Implementation of Pattern Growth Approach*. in *FIMI*. 2003.
[17]. Krupali, R. and D. Garg, *Survey on the Techniques of FP-Growth Tree for Efficient Frequent Item-set Mining.* International Journal of Computer, 2017.
[18]. Schlegel, B., R. Gemulla, and W. Lehner. *Memory-efficient frequent-itemset mining*. in *Proceedings of the 14th International Conference on Extending Database Technology*. 2011. ACM.
[19]. Sohrabi, M.K. and H. Hasannejad Marzooni, *Association rule mining using new FP-Linked list algorithm.* Journal of Advances in Computer Research, 2016. **7**(1): p. 23-34.
[20]. Leung, C.K.-S., R.K. MacKinnon, and F. Jiang. *Reducing the search space for big data mining for interesting patterns from uncertain data*. in *2014 IEEE International Congress on Big Data*. 2014. IEEE.
[21]. Nguyen, T.-N., L.T. Nguyen, and N.-T. Nguyen. *An improved algorithm for mining frequent Inter-transaction patterns*. in *INnovations in Intelligent SysTems and Applications (INISTA), 2017 IEEE International Conference on*. 2017. IEEE.