International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:5, 2015

Improved FP-growth Algorithm with Multiple Minimum Supports Using Maximum Constraints

Elsayeda M. Elgaml, Dina M. Ibrahim, Elsayed A. Sallam

***Abstract*—**Association rule mining is one of the most important fields of data mining and knowledge discovery. In this paper, we propose an efficient multiple support frequent pattern growth algorithm which we called "MSFP-growth" that enhancing the FP-growth algorithm by making infrequent child node pruning step with multiple minimum support using maximum constrains. The algorithm is implemented, and it is compared with other common algorithms: Apriori-multiple minimum supports using maximum constraints and FP-growth. The experimental results show that the rule mining from the proposed algorithm are interesting and our algorithm achieved better performance than other algorithms without scarifying the accuracy.

***Keywords*—**Association Rules, FP-growth, Multiple minimum supports, Weka Tool.

I. Introduction

DATA mining, also known as knowledge discovery in databases representational techniques for discovering knowledge patterns hidden in large databases. Many data mining approaches are being used to extract interesting knowledge from such huge data like association rule mining. Data mining is seen as an increasingly important tool by modern business to transform data into the business intelligence that giving the informational advantages. Data mining is currently used in the wide range of profiling practices, such as scientific discovery, marketing, fraud detection and surveillance.

Association rule mining searches for interesting relationships among items in a given data set. Association rule mining is used in many applications as economic and financial time series [1]. It is frequently used in the Market Basket analysis [2].

An association rule is an expression of the form X→Y, where X, Y are item sets. It shows the relationship between the items X and Y, There are two important basic measures for association rules, support (sup) and confidence (c): Support (sup.) of the rule is the fraction of the transactions that contain all items both in X and Y, i.e., sup (X→Y) = P(X ∪ Y), Confidence (conf.) is defined as the fraction of transactions containing X also containing Y, i.e., P (Y|X) =P(X ∪ Y)/P(X).

A rare association rule [14] is one of the association rule mining challenges refers to an association rule forming between both frequent and rare items that used for the knowledge in rare associations [3]. The user defines the minimum support criteria for the item to be frequent. If the itemset match the minimum support criteria, then it can be infrequent patterns. Some items with low support and have high confidence; The Mining of these items is called rare itemset mining. There is a rare item problem as in Real-world databases are mostly non-uniform in nature, containing both frequent and relatively infrequent (or rare) items [4], [5]. If the items' frequencies in a database vary widely, we encounter the following issues while mining frequent patterns under the single *minsup* framework as following:

i. If *minsup* is set too high, we will miss the frequent patterns containing rare items.
ii. If *minsup* is low, we will find frequent patterns that involve both frequent and rare items, however, this may cause combinatorial explosion, producing too many frequent patterns, because those frequent items will combine with many of them are meaningless. The same happens with a rare periodic-frequent pattern because it is difficult to mine rare periodic-frequent patterns with a single *minsup*. Hence, efforts have been made into mine periodic-frequent patterns using multiple *minsup* constraints, where *minsup* of a pattern is represented with the minimum item supports of its items (MIS).

The remainder of this paper is organized as follows: Section II explains mining association rules and describes several algorithms including Basic Apriori, multiple minimum supports using Maximum constraints (maximum constraints) and FP-growth. Section III presents the proposed algorithm (MSFP-growth). In Sections IV and V, the Implementation and results of evaluating performance of the four algorithms are discussed, respectively. The conclusion is presented in Section VI. Finally, the expected future work is drawn at Section VII.

II. Mining Association Rules Algorithms

The mining association rules generate all association rules that have support and confidence greater than or equal the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*). The problem of discovering all association rules can be decomposed into two sub problems:

(1) Finding all the frequent itemsets (whose support is greater than or equal *minsup*), also called large itemsets.

Elsayeda M.Elgaml is with Computers and Control Engineering Dept. Tanta University, Egypt. Work as software engineer in ICTP government funded project in Egypt. (phone: 002-01020699688; e-mail: elsayeda@ictp.edu.eg).

Dina M. Ibrahim is with Computers and Control Engineering Dept. Tanta University, Egypt. Work as Lecturer (phone: 002-01005743255; e-mail: dina.mahmoud@f-eng.tanta.edu.eg).

Elsayed A. Sallam is with Computers and Control Engineering Dept. Tanta University, Egypt. Work as Emeritus Professor (phone: 002-01155411019; e-mail: sallam@f-eng.tanta.edu.eg).

(2) Generating the association rules derived from the frequent itemsets.

A rule is an implication X=>Y, for the rules to be strong; they must satisfy the support threshold (minsup) and the confidence threshold (minconf). the following algorithms solve this problem. Some of the well-known algorithms for association rule mining are Apriori and FP-growth, various authors have compared the existing algorithms and various others have proposed new algorithms to remove the drawbacks of older ones. In this paper, we are comparing the algorithms using single *minsup* like Apriori and FP-growth with another which using multiple using multiple *minsup* constraints like algorithm explain in Section B below and the proposed algorithm (MSFP-growth).

A. Apriori Algorithm

Apriori algorithm is an influential algorithm for mining frequent itemsets which has been proposed in [6], [7]. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemsets properties. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. First, the frequent 1-itemset is found, this is denoted by L1, which is used to find the frequent 2-itemset L2 and so on.to improve the efficiency of the level-wise generation of frequent itemsets, a property called Apriori property is used to reduce the search space. This property states that all nonempty subsets of a frequent itemset must also be frequent. A two-step process is used to find Lk-1 from Lk:

1) The join step: To find Lk, a set of k-itemsets is generated by joining Lk-1 with itself. This set of candidate itemsets is denoted Ck.
2) The prune step: Ck is a superset of Lk, that is, its members may or may not be frequent, but all the frequent k-itemsets are included in Ck. A scan of the database is done to determine the count of each candidate in Ck, those who satisfy the minsup is added to Lk. To reduce the number of candidates in Ck, the Apriori property is used. An example of Apriori algorithm is found in [7]. This algorithm inherits the drawback of scanning the whole databases many times. It also takes the much time, space and memory to the candidate generation process. Based on the algorithm, many new algorithms were designed with some modification or improvements.

B. Mining Association Rules with Multiple Mining Supports Using Maximum Constraints

Reference [9] proposed mining association rules with non-uniform minimum support values in 1999. This algorithm is an extension of Apriori algorithm which allowed users to choose different *minsup* to different items according to its natural frequency. They also defined the MIS as the lowest minimum supports among the items in the itemset. This is not always useful because it would consider some items that are not worth to be considered because one of the items in this itemset, its *minsup* was set too low. In some cases it makes sense that the *minsup* must be larger than the maximum of the minimum supports of the items contained in an item set [8].

In [8], the authors proposed an algorithm that gives items different minimum supports. The maximum constraint is adopted in finding frequent item sets [16], [18], [22]. That is, the *minsup* denoted by MIS for an item set is set as the maximum of the user specified minimum supports of the items contained in the item set. Under the constraint, the characteristic of level-by-level processing is kept, such that the original Apriori algorithm can be easily extended to find the frequent item sets. The algorithm first finds all the frequent 1-itemsets defined as (L1) for the given data transactions by comparing the support of each item with its predefined *minsup*. After that, candidate 2-itemsets C2 can be formed from L1. Note that, the supports of all the frequent 1-itemsets including each candidate 2-itemset must be larger than or equal to the maximum of their user specified *minsup*. This feature provides a good pruning effect before the database is scanned for finding large 2-itemsets.The algorithm then finds all the large 2-itemsets L2 for the given transactions by comparing the support of each candidate 2-itemset with the maximum of the user specified *minsup* of the items contained in the item set. The same method is repeated until all frequent item sets have been found. An example of the algorithm is found in [8].

C. FP-growth Algorithm

Reference [10] proposed a new tree structure, called a FP-tree, which is an extended prefix-tree structure for sorting compressed and crucial information in 2000. Consequently, the FP-growth method is a FP-tree based mining algorithm for mining frequent patterns. The Fp-growth approach is based on divide and conquer strategy for producing the frequent itemsets. FP-growth is mainly used for mining frequent itemsets without candidate generation shown in [10] to remove the drawbacks of the Apriori algorithm. Major steps in FP-growth are:

Step 1. It firstly compresses the database showing frequent itemset into FP-tree. FP-tree is built using 2 passes over the dataset.

Step 2. It divides the FP-tree into a set of conditional database and mines each database separately, thus extract frequent item sets from FP-tree directly.

The important step of FP-growth algorithm is the process to construct the FP- tree [15], which needs to scan the transaction itemset twice: scan the database of transaction T once to find out the frequent 1-itemset L, then arrange the support count in descending order to get the L1 then take the "Null" as the root node when scan the transaction itemset for the second time, then construct the FP-tree base on L1.

The next step is mining the frequent itemsets from the FP-tree after constructing the original FP-tree. The steps as following:

1) Produce conditional pattern base for every node in the FP-tree.
2) Build the corresponding conditional FP-tree from the conditional pattern base.

3) Mine the conditional FP-tree recursively and increase the frequent itemset belong to it at the same time by producing the involved frequent itemset immediately if the conditional FP-tree only contains one path. Otherwise, increase the minimum item of support count if the suffix pattern.

Then construct the conditional pattern base and conditional FP-tree (The conditional pattern base is the entire branch which takes (E) as their leaf node in the FP-tree. The conditional FP-tree of is a new FP-subtree taking the conditional pattern base as its transaction and constructed in the same way of the original FP-tree An example of the algorithm is found in [11].

III. Mining Enhancement FP-Growth Algorithm with Multiple Minimum Supports Using Maximum Constraints: The Proposed Algorithm

Based on the algorithms explained in Section II, a new algorithm is proposed for enhancement FP-growth Algorithm by making infrequent child node pruning step so that the size of the MIS-tree result less than or equivalent to the FP-tree constructed by FP-growth approach then mining the proposed Algorithm but with specifying different *minsup* to each individual item. The proposed algorithm is an efficient tree-based algorithm for mining association rules with multiple minimum supports using Maximum Constraints (MSFP-growth).

A. The Proposed Algorithm

The MSFP-growth is an extension of single *minsup* based FP- growth approach to multiple *minsup* values [12], [13], [17]. This approach includes two steps. They are a construction of MIS-tree [23] and mining frequent patterns from the MIS-tree using conditional pattern bases. This approach assumes that the MIS values for each item will be choosed by the user; the MIS- tree is constructed as:

1. The items are sorted in descending order of their MIS values (L1).
2. A root node of the tree is constructed by labeling with" null".
3. For each transaction the nodes represent the items, level of nodes in a branch is based on the sorted order and the count of each node is set to 1.
4. Update the frequencies of the items which are present in the transaction by incrementing the frequency value of the respective item by 1 and each node along a common prefix is incremented by 1, the nodes for the items following the prefix are created linked to them and their values are set to 1.

To facilitate tree structure, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. From the item frequencies, the respective support values are calculated using the lowest MIS value among all the items, any frequent pattern will have support greater than or equivalent to the lowest MIS value among all the items. Otherwise, the tree-pruning process is performed on the item header and MIS-tree to remove the items that have support is less than the lowest MIS value. After tree pruning, tree-merging process and mining the MIS-tree result.

The proposed growth approach involves three steps constructing the MIS-tree, Infrequent child node pruning step and Mining frequent patterns from MIS-tree.

1) Constructing MIS-tree

The construction of MIS-tree in MSFP-growth algorithm described as follows. There are input parameters to these algorithm transaction dataset (Trans), Itemset (I) and minimum item support values (MIS) of the items. Using these input parameters, the MSFP-growth creates an initial MIS-tree. Next, starting from the last item in the item-header table (i.e., item having lowest MIS value), it perform tree-pruning to remove the infrequent items from the item-header table and MIS-tree. The result is an efficient MIS-tree, shown in Fig. 1.

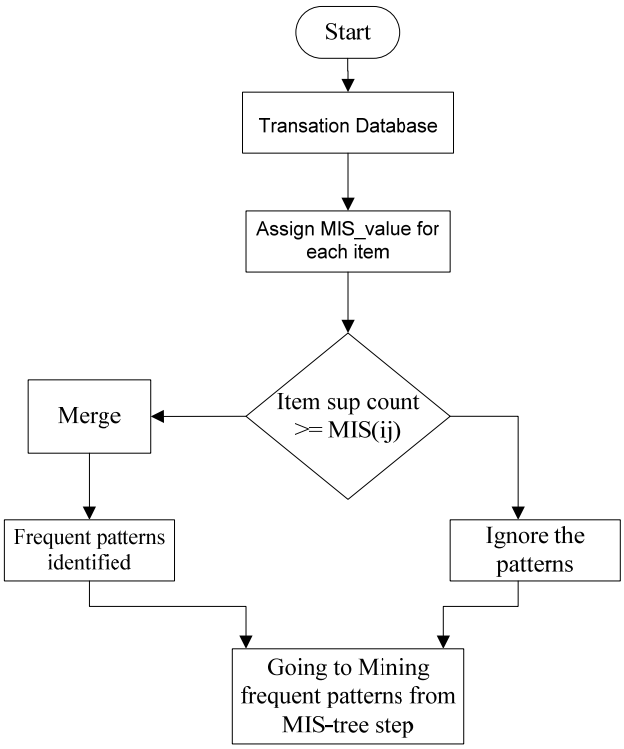


Fig. 1 Flow Diagram of MIS-Tree

2) Infrequent Child Node Pruning

The MSFP-growth approach skips the construction of conditional pattern bases for the infrequent items. So in the efficient MIS-tree, the child nodes belong to infrequent items have not any importance because its prefix paths (conditional pattern bases) are not used. So, if we can't prune the child nodes belonging to infrequent items making the MIS-tree result less than or equivalent to the FP-tree constructed by the FP - growth approach so in the MIS-tree, "infrequent child node pruning" is performed such that every branch ends with the node of frequent item. Pruning should be performed only on the child nodes belonging to infrequent items.

3) Mining Frequent Patterns from MIS-Tree

The process of mining the MIS-tree in proposed algorithm is almost same as mining the MIS-tree in FP- growth but, the variance between the two approaches is that before generating conditional pattern base and conditional MIS-tree for every item in the header of the Tree, MIS is specified for each item, then check if each item's sup-count is greater than or equals its predefined MIS and If an suffix item is not a frequent pattern then the construction of conditional pattern base and conditional MIS-tree are skipped and generate the frequent itemset .After finding the frequent itemsets check if the rule's conf is greater than or equals *minconf* as Fig. 2.

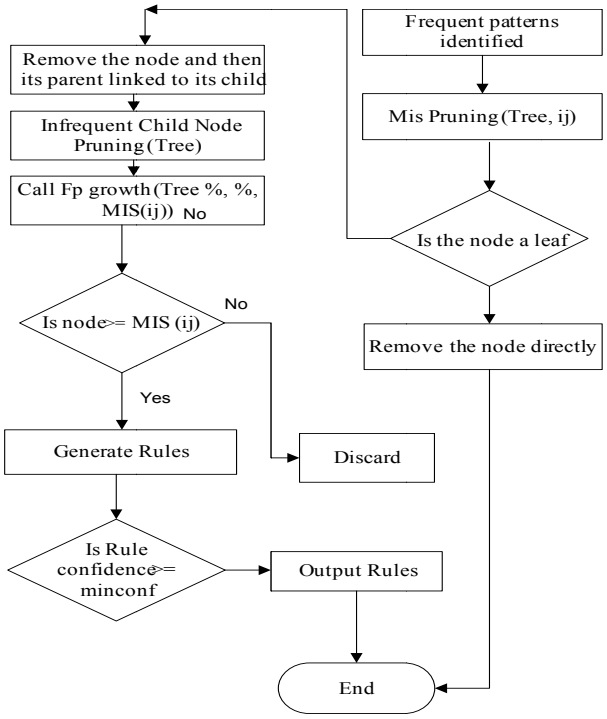


Fig. 2 Flow Diagram for Mining Frequent Pattern from MIS-tree

B. Example for the Proposed Algorithm

In the example illustrated in Figs. 3, 4 the itemsets are generated according to the steps mentioned above; if the *minconf* specified is 50%, three frequent patterns are generated by using proposed algorithm as shown below:

(a) Fig. 3: Construction of Initial MIS-tree.
(b) Fig. 4: Mining Frequent Pattern from MIS-Tree, The pruning step and conditional closure property then generate the output rules.

IV. Implementation

The phases of experimental results for the comparison between the four algorithms mentioned in the paper are:

A. Weka Tool

These association rule mining algorithms were implemented using Weka open source tool [19]; the software is freely available at [20] which is written in Java language. There are two algorithms built in Weka tool: Apriori and FP-growth). We implement the two remaining algorithms, maximum constraints and MSFP-growth, and then adding them to Weka tool.

TID	ITEMS
1	a, b
2	a, e, f
3	c, d
4	a, b, h
5	c, d
6	a, c
7	a, b
8	e, f
9	c, d, g
10	a, b
11	a, b
12	a, c
13	a, b
14	b, e, f, g
15	c, d
16	a, b, d
17	c, d
18	a, c
19	a, b, e
20	c, d

Transaction Table

ITEMSET	MIS
a	10
b	8
c	10
d	6
e	3
f	3
g	3
h	2

ITEMSET	COUNT
a	12
b	9
c	9
d	7
e	4
f	3
g	2
h	1

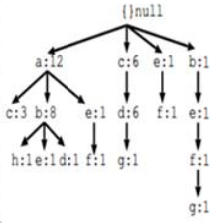
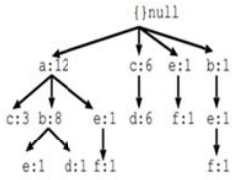
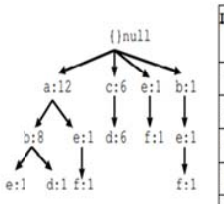


Fig. 3 Construction of Initial MIS-tree

ITEMSET	COUNT	MIS
a	12	10
c	9	10
b	9	8
d	7	6
e	4	3
f	3	3
g	2	3





ITEMSET	MIS	Cond. pattern bases	Cond. MIS-tree	frequent patterns
f	3	{{a,e:1},{e:1},{b,e:1}}	<e:3>	{{e,f:3}}
e	3	{{a,b:1},{a:1},{b:1}}	-	-
d	6	{{a,b:1},{c:6}}	<c:6>	{{c,d:6}}
b	8	{{a:8}}	<a:8>	{{a,b:8}}
c	10	-	-	-

Fig. 4 Mining Frequent Pattern from MIS-tree, the pruning step and conditional closure property

B. Dataset and Data Preprocessing Step

There is a dataset used to test the four algorithms. The dataset generated from Adventure Works DW has the following characteristics:

- Number of transactions: 21,255
- Number of individual Items: 37
- CSV File size: 5.58 MB

Data preprocessing step is decomposed into two steps; first, the database is processed to be in the form of Transactions (TID, Items). Second, saving it in a CSV file according to the Weka formatting.

C. The Comparison Procedures between Single and Multiple Supports Algorithms

The comparison between any mining association rule algorithms is either made on a number of rules generated by each algorithm or on their processing time. To compare the processing time, same algorithm parameters should be used which are the *minsup* and *minconf*. But using the same *minsup* causes confusion when comparing an algorithm that takes one *minsup* value and another that takes multiple *minsup* criteria. If the output of the single and multiple supports algorithms is the same, it means that both had equivalent parameters. The procedures show in the flow chart of [Fig. 3] are used to specify a *minsup* to each item in order to unite the output of single and multiple supports algorithm. This will make comparing the processing times is based on a reliable aspect by uniting the output; this procedure has been tested in [21].

In Fig. 5, the following steps are taken to decide what *minsup* should be specified for each individual item when comparing FP-growth with MSFP-growth.

The comparison procedures are:

1. Take the rules generated by FP-growth algorithm with a specific minsup.
2. Get the sup-count of each itemset.
3. Calculate the values of MIS of each itemset.
4. For each item in the itemset, specify a minsup equals to MIS.
5. For each rule, determine what itemsets are contained in each rule.
6. To fulfill the condition which says that in any itemset the sup-count of each item must be greater than or equal MIS, so in backward method MIS must be less than or equal sup-count of each itemset to allow the itemset to be generated.
7. After calculating the values of MIS, we can specify the same value of MIS as a minsup for each item in the itemset.
8. In this way each item could be specified more than one minsup; if this happens the minimum minsup should be specified.

Finally, some items weren't specified any mix-up which means that they were not included in the rules generated so, they should be specified a *minsup* greater than their sup-count to be excluded from frequent itemset.

Example to Explain the Comparison Procedures shown in Fig. 5:

The rules generated by Basic FP-growth Algorithm when applied on AdventureWorksDW where the *minsup* was set to 2% and *minconf* is set to 50%:

[Water Bottle=t, Mountain200=t]: 589 ==> [Mountain Bottle Cage=t]: 589

The following itemsets must be generated first. The itemsets and their support count are:

Mountain-200, Water Bottle = 589

Mountain-200, Mountain Bottle Cage= 725

Water Bottle ==> Mountain Bottle Cage = 1623

Mountain-200, Water Bottle, Mountain Bottle Cage = 589

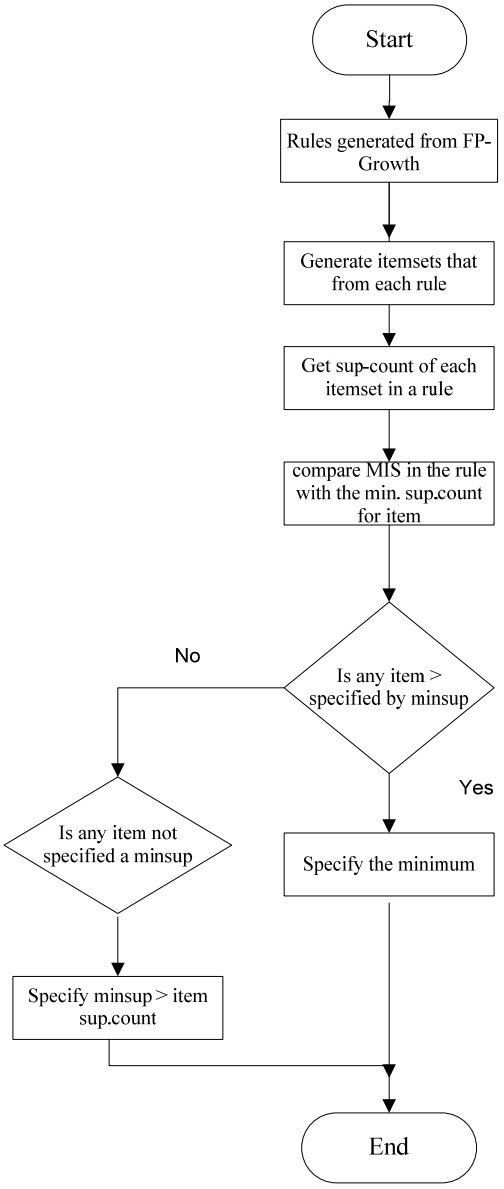


Fig. 5 Comparison Procedures

For the three items, their max predefined support should not exceed any of the support counts specified above, so if their specified *minsup* is equal to their least number = 589, we guarantee that those four itemsets are generated and so this rule will be generated.

minsup (Mountain-200) = 589

minsup (Water Bottle) = 589

minsup (Mountain Bottle Cage) = 589

The comparison procedures are applied when comparing Apriori with maximum constraints and when comparing FP-growth with MSFP-growth.

V. Performance Evaluation and Simulation Results

A. Processing Time

The four algorithms mentioned in the paper are tested and evaluated for time and accuracy. Following the procedures mentioned in the above section, specify a *minsup* for each item to test the multiple supports algorithms and then generate rules at constant *minconf* and different values for *minsup*. The processing time of each algorithm when applied on AdventureWorksDW is illustrated in Table I and the comparison graph is illustrated in Fig. 6.

TABLE I
Processing Times of Apriori, Max. Constraints, FP-Growth and MSFP-Growth by AdventureWorksDW Dataset

Apriori equiv. Support count (%)	Apriori Time (Sec)	Max. Constraints Time (Sec)	FP-growth Time (Sec)	Proposed Algorithm (MSFP-growth) Time (Sec)
0.2	7	5	0.5	0.2
0.175	8	6	0.6	0.3
0.15	9	7	0.7	0.4
0.125	10	8	0.8	0.5
0.1	14	11	0.9	0.6
0.075	15	13	1	0.7

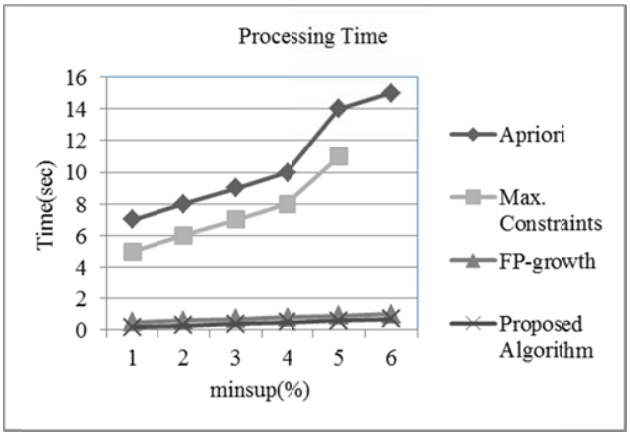


Fig. 6 Comparison of processing time between Basic Apriori, max constraints, FP-growth and (MSFP-growth) Proposed Algorithm of AdventureWorksDW

In Fig. 6 MSFP-growth takes the least amount of time among other algorithms and Basic Apriori takes the longest time. The time increases when the *minsup* decreases because the number of individual itemsets and rules generated increase.

B. Accuracy Test and Interesting Measurements

After building a mining model, the validity of the model should be tested. The data must be randomly separated into two separate datasets (training and testing). The training dataset is used to build the model, and the testing dataset is used to test the accuracy of the model. This is a part of the software engineering cycle to test many algorithms that solve the same problem, then test their efficiency in solving the problem.The AdventureWorksDW dataset is randomly separated to test the four algorithms mentioned in this paper.

The four algorithms are applied in the training and testing datasets at different values of *minsup* with constant *minconf*. Follow the comparison procedures mentioned in Section IV, to specify the *minsup* that should be given to each individual item while testing the multiple supports algorithm. The specification is a percentage of the *minsup* assigned to the original dataset, so the algorithms do not generate the same number of rules and the time is calculated as the time taken to generate each rule. After performing the tests, the results are collected to calculate the accuracy of each algorithm by applying the concept of accuracy index where:

$$\text{accuracyindex} = \frac{\sum_n accuracy}{\max(accuracy)} \quad (1)$$

Equation (1) shows that if an algorithm has the accuracy of 100%, it is the highest accuracy among the other algorithms. It does not mean that it has 100% absolute accuracy.

The steps of the accuracy test are:

1) Divide the data into two datasets randomly.
2) Mine each dataset separately by using weka tool.
3) Compare the rules generated from both dataset and find their intersection, then calculate the percentage of this intersection.

Applying the time index on the results shown in Table I where:

$$\text{timeindex} = \frac{\sum_n time}{\max(time)} \quad (2)$$

Equation (2) shows that if an algorithm takes 100% time, it takes the longest amount of time among other algorithms. For example, after dividing the data into 90% and 10%. The 90% dataset generates 50 rules and the 10% dataset generates 60 rules. If the number of rules intersections are 20. Then the percentage on first part is 20/50*100=40% and the percentage on second part is 20/60*100=33.33%. Also, we execute the previous step for each of the 20% and 80%, and then 30% and 70%.

Table II shows the results of the accuracy and time indices of AdventureWorksDW and these results are illustrated in Figs. 7 and 8.

TABLE II
Accuracy and Time Indices of AdventureWorksDW

	Apriori	Max. Constraints	FP-growth	Proposed Algorithm
Time inedx %	100.00	97.77	80.45	81.40
Accuracy index %	97.89	98.25	99.50	100.00

Fig. 8 shows that the MSFP-growth (Proposed Algorithm) is faster than other algorithms. Basic Apriori takes the longest time and MSFP-growth takes the shortest. Fig. 7 shows that the accuracy of the Proposed Algorithm is the best among other algorithms and the difference between the accuracy of FP-growth and MSFP-growth is in the range of 0.5% .That means the MSFP-growth is faster than any other algorithms and the accuracy is the best when applied on dataset without

affecting the number of rules generated because it generates less itemsets due to the two conditions which is added to the algorithm (it uses the concept of multiple *minsup* that gives the user flexibility in specifying a *minsup* for each item and making infrequent child node pruning step).

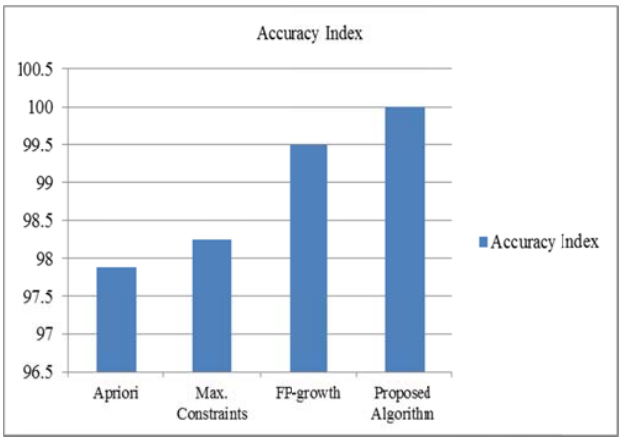


Fig. 7 Accuracy index of the four algorithms of AdventureWorksDW

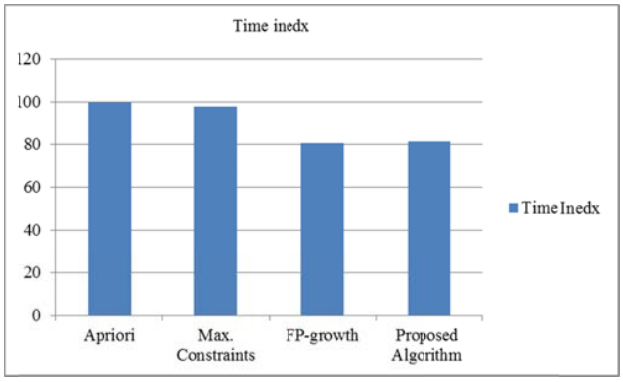


Fig. 8 Time index of the four algorithms of AdventureWorksDW

VI. Conclusion

This paper presents an algorithm which is an enhancement of the FP-growth algorithm by making infrequent child node pruning procedure for each infrequent itemset to reduce MIS tree size with the multiple minimum support criteria. The simulation results of the proposed algorithm (MSFP-growth) are faster than any other algorithms and the accuracy is the best when applied on dataset without affecting the number of rules generated. The new algorithm consumes less time than the FP-growth and the accuracy isn't affected in a bigger percentage when applied on a dataset that has a huge number of individual items. The rules generated from the algorithms are 100% interest when applied to AdventureWorksDW dataset. In this new algorithm the user is given the flexibility to specify a different *minsup* for each individual item. This option overcomes the problem of rare items that need to be mined too in [5]. The method presented is to produce the exact same rules from multiple supports algorithms as generated by single support algorithms. It can be used to compare any single support algorithm with a multiple supports algorithm.

VII. Future Work

In the future, it is recommended to test the algorithms with different database. Also, applying new values of *minconf* by the Weka tool on the multiple supports algorithms. These features test the reliability of the data mining technique in terms of accuracy and efficiency, which is more accurate than the method presented here. This should be done because the Weka tool does not support a multiple supports algorithms.

References

[1] S. Brin, R. Motwani, J. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in SIGMOD-97, 1997, pp. 255-264.
[2] C. Aggarwal, and P. Yu "Online generation of association rule," in ICDE-98, (1998), pp. 402-411.
[3] R. Uday Kiran, P. Krishna Reddy , "An efficient approach to mine rare association rules using maximum items support constraints data Security and security data, " Springer, vol. 6121, pp. 84-95,2010.
[4] L. Troiano, G. Scibelli and C. Birtolo, "A fast algorithm for Mining rare itemsets," in Conf. 2009 the Ninth IEEE International Conference on Intelligent Systems Design and Applications, pp. 1149-1155.
[5] K. S. C. Sadhasivam, T. Angamuthu, "Mining rare Itemset with Automated Support Thresholds," Computer Science J., vol. 7, pp. 394-399, 2011.
[6] R. Agrawal, T. Imielinksi and A. Swami, "Mining association rules between sets of items in large database," in 1993The ACM SIGMOD Conf., pp. 207–216.
[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in 1994VLDB Conf., Santiago, Chile, Expanded version available as IBM Research Report RJ9839.
[8] Y. Lee, T. Hong and W. Lin, "Mining association rules with multiple minimum supports using maximum constraints," Elsevier Inc., 2005.
[9] B. Liu, W. Hsu and Y. Ma, "Mining association rules with multiple minimum supports," in 1999The International Conf. on Knowledge Discovery and Data Mining, pp. 337–341.
[10] J. Han, J. Pei and Y. Yin, "Mining frequent patterns without candidate generation," in 2000International Conf. on Management of data, vol. 29, pp. 1-12.
[11] Z. Sun, "Analysis and implementation of the algorithm of fp-growth," Guangxi Institute of Technology J., vol. 16, no. 3, pp. 64-67, 2004.
[12] R. U. Kiran, P. K. Reddy, "An improved multiple minimum support based approach to mine rare association rules," in 2009 IEEE Int. Conf. Communications Computational Intelligence and Data Mining, pp. 340-347.
[13] R. U. Kiran, P. K. Reddy , "An Improved Frequent Pattern growth Approach To Discover Rare Association rules," in 2009 International Conf. on Knowledge Discovery and Information Retrieval, pp. 43-52.
[14] S. Tsang, Y. S. Koh and G. Dobbie, "Finding Interesting Rare Association Rules Using Rare Pattern Tree," The University of Auckland, Springer-Verlag Berlin Heidelberg,2013.
[15] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, M. Hsu, , "Mining sequential patterns by pattern-growth: the prefixspan approach," *IEEE Trans.* Knowledge and Data Engineering 16, 1424–1440, 2004.
[16] R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints," KDD-97, pp. 67-73, 1997.
[17] Y. Hu, F. Wu, Y. Liao, "Sequential pattern mining with multiple minimum supports: a tree based approach", in 2010, The 2nd International Conf. on Software Engineering and Data Mining, Chengdu, China.
[18] J. Pie, J. Han, and L. Lakshmanan., "Mining frequent itemsets with convertible constraints", in 2001 IEEE ICDE Conf., p.433– 442.
[19] M. S. Saravanan, R.J. Rama, "Performance study of Association Rule Mining Algorithms for Dyeing Processing System Innovative Systems Design and Engineering", Vol 2, No 5, 2011.
[20] http://www.cs.waikato.ac.nz/ml/weka , Last acess : 6 march 2015

[21] M. Walaa, H. Ahmed and K. Hoda, "Combined Algorithm for Data Mining using Association rules," Ain Shams of Electrical Engineering J., Vol. 1.No. 1 ISSN: 1687-8582, 2008.
[22] P. Jyothi, V. D. Mytri, "A Fast association rule algorithm based on bitmap computing with multiple minimum supports using max constraints," International of Comp. Sci & Electronics J.,Volume1,Issue 2, IJCSEE,2013.
[23] F.A. Hoque , N. Easmin and K. Rashed, " Frequent pattern mining for multiple minimum supports with support tuning and tree maintenance on incremental database," Research of Information Technology J., 3(2): 79-90 ,2012.

Elsayeda M.Elgaml is software engineer in ICTP government funded project in Egypt scince 2009 .His Birth Place in Ismaili on February 15th 1988 .His research inetests are Data Mining, Algorithms and Web Technologies.

Dina M. Ibrahim, Lecturer at Computers and Control Engineering Department-Faculty of Engineering-Tanta University, Egypt. She was born in United Arab of Emarat at 1980, her B.Sc., M.Sc. and Ph.D. degrees taken from Computers and Control Engineering Department - Faculty of Engineering - Tanta University at 2002, 2008, and 2014, respectively. Dr. Dina Works as a Database administrator, then a consultant Engineer, and finally a Vice-Manager at Management Information Systems (MIS) Project – Tanta University, Egypt, from August 2008 until November 2014.

Elsayed A. Sallam, Emeritus Professor at Computers and Control Engineering Department-Faculty of Engineering-Tanta University, Egypt. His B. Sc. degree taken from Faculty of Engineering-ElMenofia University, Egypt at 1977. M.Sc. and Ph.D. degrees taken from University of Bremen, German at 1983, and 1987, respectively. Dr. Elsayed was head of the Computers and Control Engineering Department-Faculty of Engineering-Tanta University from 2008 till 2014. He works as a Manager at Management Information Systems (MIS) Project– Tanta University, Egypt, from August 2008 until December 2011. Then as a CIO - Tanta University, Egypt, from December 2011until November 2014.