International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 4, Issue 4, 2017, PP 22-30
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0404003
www.arcjournals.org

Analyzing Working of FP-Growth Algorithm for Frequent Pattern Mining

Amanvir Kaur[1], Dr. Gagandeep Jagdev[2]

[1]*Research Scholar (M.Tech.), Yadavindra College of Engineering, Talwandi Sabo (PB)*
[2]*Dept. of Comp. Science, Punjabi University Guru Kashi College, Damdama Sahib (PB), India*

Corresponding Author:** ***Dr. Gagandeep Jagdev, *Dept. of Comp. Science, Punjabi University Guru Kashi College, Damdama Sahib (PB), India.*

Abstract: *Frequent pattern mining has always been a topic of curiosity among the section of researchers and practitioners having a concern with data mining. It has proved its acceptance as a key complication in the field of data mining. The practice relates to the finding of itemsets that frequently appear in plenty of data under consideration. It is a kind of finding correlations and relationships between items involved in the data set. One may be interested in finding the frequent itemsets over the sliding window or in the complete data stream. In this research paper, the objective is to find all the itemsets that satisfy the minsup threshold. These itemsets are referred as frequent itemsets. Often the computational requirements for exploring the frequent itemsets are too expensive. The research paper elaborates the working of FP-Growth tree and formation of conditional FP-Tree via an example.*

Keywords: *Conditional FP-Growth, FP-Tree, itemsets, transactions*

1. Introduction

The fastest and most popular for frequent pattern mining is FP-growth algorithm. It works on prefix tree representation of the transactional database under study and saves a considerable amount of memory. The FP-growth algorithm is a kind of recursive elimination scheme [1, 2]. The itemsets, subsequences, or substructures that appear in a particular data set with a frequency no less than defined as the threshold by the user is defined as frequent patterns. For instance, a set of items like milk and bread often appears together in a transaction data and can be considered as a frequent itemset. Pattern mining can be applied to graphs, strings, spatial data, sequence databases, streams, transaction databases, etc. The list of frequent patterns is mentioned as under [3].

- **Frequent Itemset** − It refers to a set of items that frequently appear together, for example, milk and bread.
- **Frequent Subsequence** − A sequence of patterns that occur frequently such as purchasing a camera is followed by the memory card.
- **Frequent Sub Structure** − Substructure refers to different structural forms, which may be combined with itemsets or subsequences.

FP-Growth algorithm is the most popular algorithm for pattern mining. It is based on divide and conquer strategy. Compress the database providing frequent sets and divide this compressed database into a set of conditional databases, each related to a frequent set and apply data mining on each database.

2. Detailed Working of FP-Growth Algorithm

The FP-Growth algorithm allows the discovery of frequent itemset without generating candidate itemset. It is a two-step approach mentioned as under [4, 5].

1) Firstly a compact data structure is built which is referred as FP-tree. It is built using two passes over the data-set.

2) Traverse through FP-Tree and extract frequently occurring itemsets.

The steps involved in the working of the FP-Growth algorithm are mentioned as under [10, 11].

- The nodes resemble items and have a counter.
- One transaction is read at a time and mapped to a path.
- The use of fixed order is made so that paths are overlapped when items are shared among transactions, provided they have the same prefix. In such cases, counters are incremented.
- Pointers are maintained between the nodes having the similar item to create the singly linked list as shown by dotted lines in Fig.
- As the overlapping of paths increases so does the compression. In such cases, FP-Tree fits in the memory.
- Finally, frequent itemsets are extracted from the FP-Tree.

Consider a transaction data set as shown in Fig. 1 containing 10 entries each specified with a unique TID (Transaction ID).

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

Fig1. *The figure shows the transaction data set under consideration*

Fig. 2 shows the TID 1 mapped to the path.

Read transaction 1: {a; b}

Create 2 nodes a and b and the path null -> a -> b. Set counts of a and b to 1.

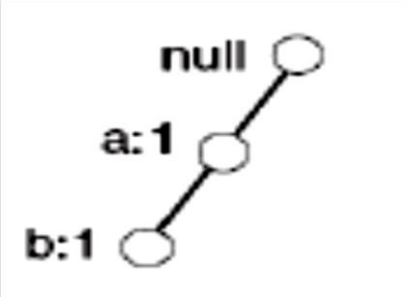

Fig2. *The figure depicts the TID 1 mapped to the path*

Fig. 3 shows the TID 2 mapped to the path.

Read transaction 2: {b; c; d}

Create 3 nodes for b, c and d and the path sets to null -> b -> c -> d. Set counts to 1. Although TID 1 and TID 2 share b, but paths will remain disjoint as they don't share a common prefix. A link needs to be added between the b's.

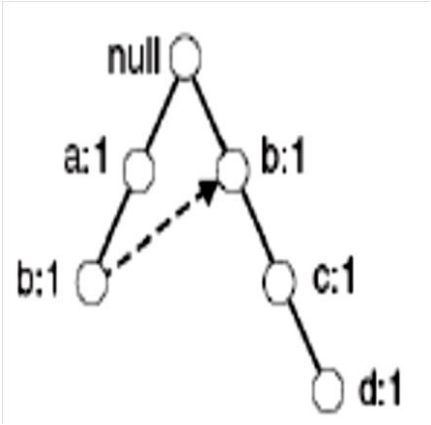

Fig3. *The figure shows the TID 2 mapped to the path*

Fig. 4 shows TID 3 mapped to the path.

Read transaction 3: {a; c; d; e}

The transaction 3 shares the common prefix item 'a' with the transaction 1, therefore the path of TID 1 and TID 3 will overlap and the frequency count for node 'a' will increase by 1. Further links need to be added between the 'c' and 'd'.

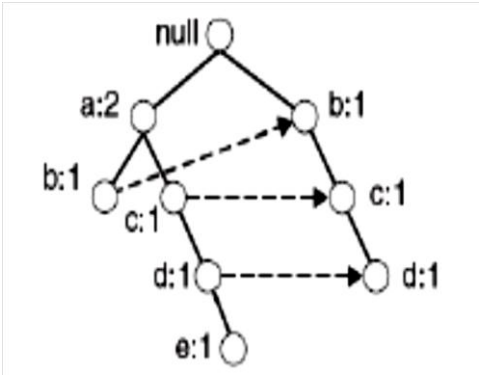

Fig4. *The figure shows the TID 3 mapped to the path*

This process is continued until all transactions are mapped to a path in the FP-tree as shown in Fig. 5.

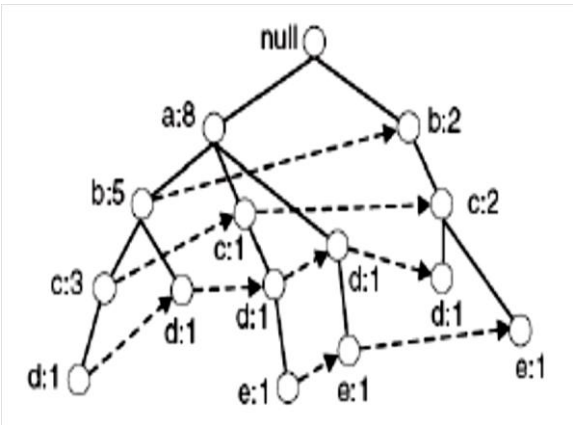

Fig5. *The figure shows the FP- Tree formed after performing all 10 transactions*

In comparison with uncompressed data, the FP-Tree usually have a smaller size as many transactions share items i.e. prefixes [6, 7]. In case of best case scenario, all transactions are supposed to contain the same set of items and results in 1 path in the FP-Tree. But in case of worst-case scenario, each transaction has a unique itemset, i.e. no items in common. The size of the FP-Tree is at least as large as the original data and the storage requirements are higher as there is need to store pointers to the nodes. The ordering of items has a deep impact on the size of the FP-Tree. Often ordering by decreasing support is preferred but it does not always guarantee the smallest tree.

Next one has to generate the frequent itemsets from the FP-Tree. In doing so, the bottom-up approach is adopted. Considering the FP-Tree obtained in Fig. 5, first there is a need to look for frequent itemsets ending in e, then de, etc. Similarly thereafter d, then cd, and so on.

The Fig. 6 shows the prefix path sub-tree for 'e' used to extract frequent itemsets ending in 'e'.

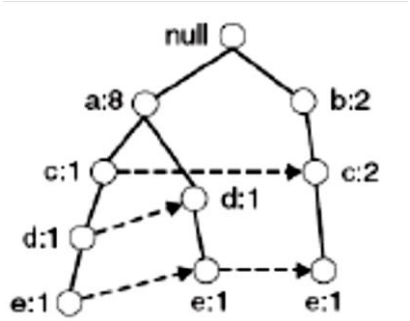

Fig6. *The figure depicts the path containing node 'e'*

Fig. 7 shows the prefix path sub-tree for 'd' used to extract frequent itemsets ending in 'd'.

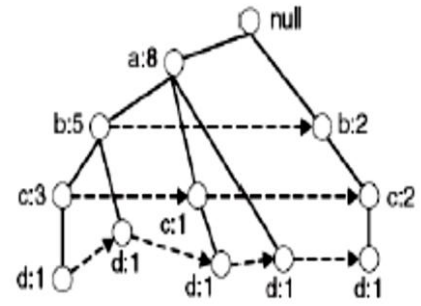

Fig7. *The figure depicts the path containing node'd'*

Fig. 8 shows the prefix path sub-tree for 'c' used to extract frequent itemsets ending in 'c'.

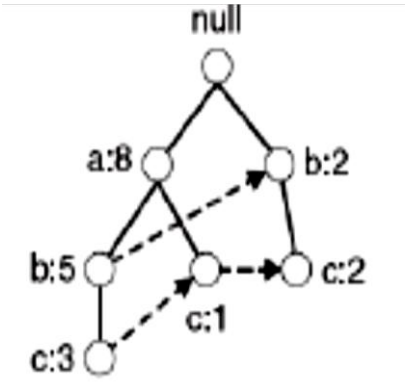

Fig8. *The figure depicts the path containing node 'c'*

Fig. 9 shows the prefix path sub-tree for 'b' used to extract frequent itemsets ending in 'b'.

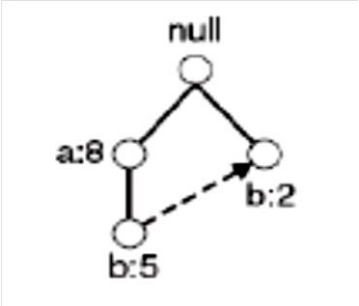

Fig9. *The figure depicts the path containing node 'b'*

Fig. 10 shows the prefix path sub-tree for ‘a’ used to extract frequent itemsets ending in ‘a’.

Fig10. *The figure depicts the path containing node ‘a’*

Now this practice is further expanded and the prefix path sub-tree for ‘e’ is used to extract frequent itemsets ending in ‘de’, ‘ce’, ‘be’, and ‘ae’ as shown in Fig. 11.

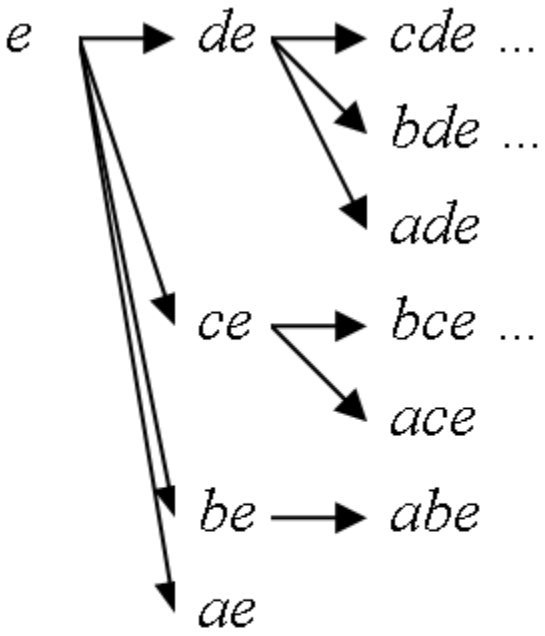

Fig11. *The figure depicts the frequent itemsets ending in ‘de’, ‘ce’, ‘be’, and ‘ae’*

This process is further carried out for ‘cde’, ‘bde’, and ‘ade’ and so on.

Now assume the minimum threshold value as 2 denoted as ‘. *‘minsup’*. Now *minsup=2* and extract all the frequent itemsets containing ‘e’. Fig. 12 shows the prefix path sub-tree for ‘e’.

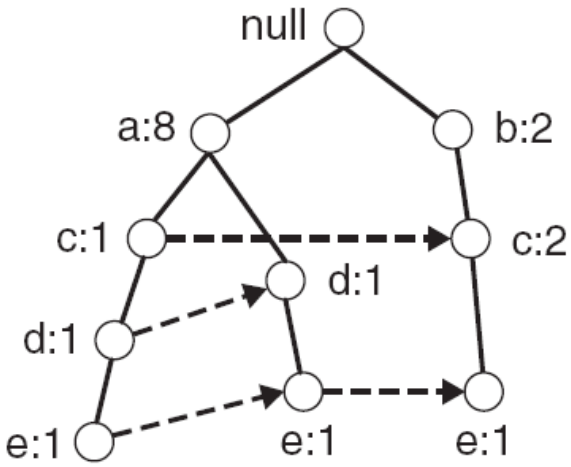

Fig12. *The figure shows the prefix path sub-tree for ‘e’*

Next step is to check whether ‘e’ is a frequent item and this can be done by adding the counts along the linked list i.e. dotted line. In case of ‘e’, the count value is 3 and is extracted as a frequent itemset.

Next find the frequent itemsets ending with e, i.e. ‘de’, ‘ce’, be’, and ‘ae’. For accomplishing this, we need conditional FP-Tree for ‘e’.

The conditional FP-Tree is constructed if only transactions containing a particular itemset are considered and thereafter the itemset is removed from the selected transactions [8, 9].

Fig. 13 shows the table for FP-Tree conditional on ‘e’. Firstly all the transactions in which ‘e’ is absent, is stroked off. Thereafter, from the left out transactions containing ‘e’, the item ‘e’ is stroked off.

TID	Items
~~1~~	~~{a,b}~~
~~2~~	~~{b,c,d}~~
3	{a,c,d,~~e~~}
4	{a,d,~~e~~}
~~5~~	~~{a,b,c}~~
~~6~~	~~{a,b,c,d}~~
~~7~~	~~{a}~~
~~8~~	~~{a,b,c}~~
~~9~~	~~{a,b,d}~~
10	{b,c,~~e~~}

Fig13. *The figure shows the table for FP-Tree conditional on ‘e’*

Now as per Fig.13, there is a need to update the prefix paths from ‘e’ inorder to reflect the number of transactions containing ‘e’. The item ‘a’ is set to 2 and items ‘b’ and ‘c’ should be set to 1 each as shown in Fig. 14.

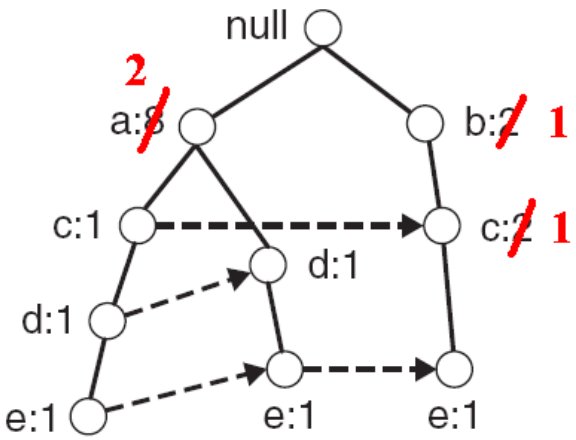

Fig14. *The figure shows the updated values of items ‘a’, ‘b’, and ‘c’*

Now remove the nodes having ‘e’ as shown in Fig.15 and Fig. 16.

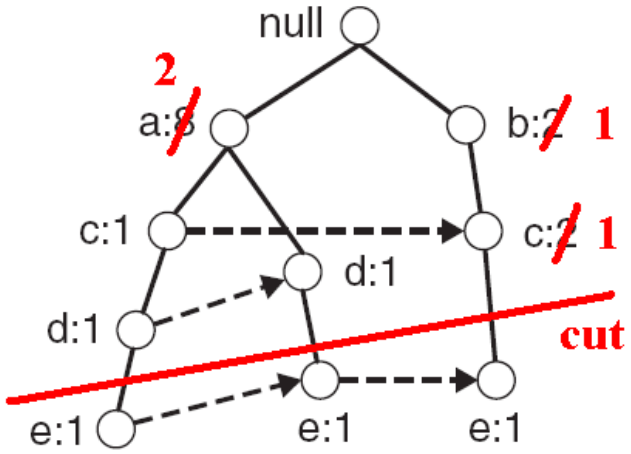

Fig15. *The figure shows the stroked of nodes containing 'e'*

Next, the removal of infrequent items from the prefix paths is initiated. 'b' has a support of 1 and there is only 1 transaction containing 'b' and 'e', hence 'be' can be declared infrequent and can be removed.

Fig. 16 shows the figure of final constructed conditional FP-Tree on 'e'.

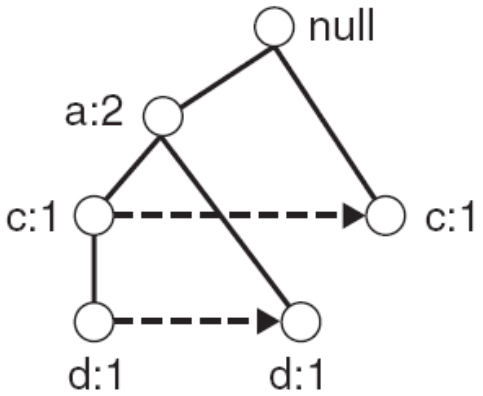

Fig16. *The figure shows the conditional FP-Tree on 'e'*

The conditional FP-Tree for 'e' can be used to find frequent itemsets ending in 'de', 'ce', and 'ae'. Here 'be' cannot be considered as 'b' is not in the conditional FP-Tree for 'e'. Proceed to find the prefix paths from the conditional tree on 'e' for 'de', 'ce', and 'ae'.

In case of 'de', e -> de ->ade are found to be frequent.

Fig. 17 shows the process adopted to find the conditional FP-Tree for 'de'.

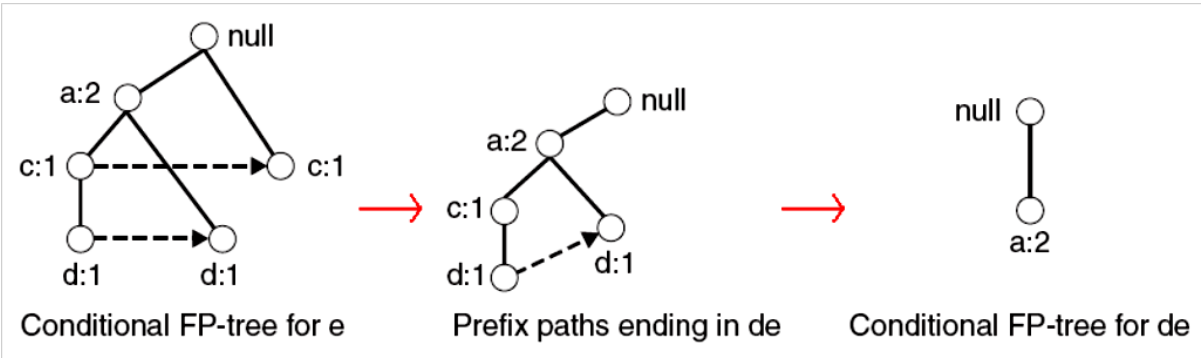

Fig17. *The figure shows the conditional FP-Tree for 'de'*

In case of 'ce', e ->ce ({ce} is found to be frequent).

Fig. 18 shows the conditional FP-Tree for 'ce'.

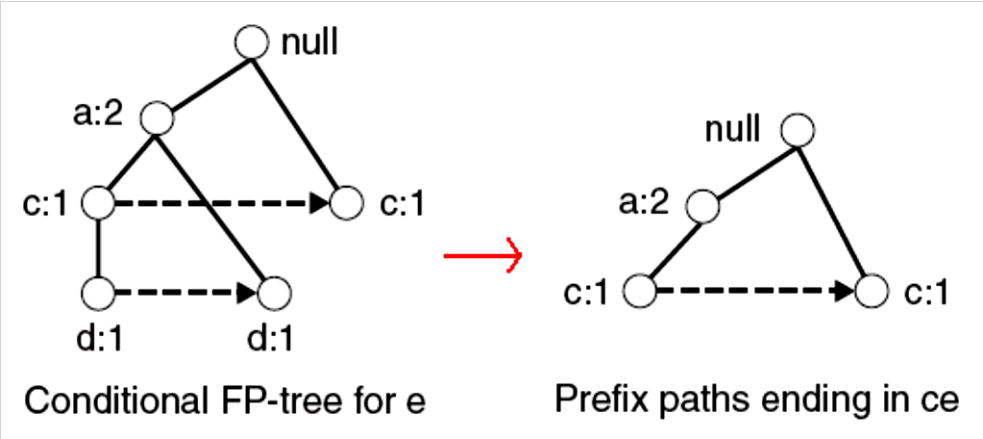

Fig18. *The figure shows the conditional FP-Tree for 'ce'*

And process continues in similar manner.

The final result obtained in regard to finding frequent itemsets is shown in Fig. 19.

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

Fig19. *The figure shows the final result obtained showing frequent itemsets*

3. Pros and Cons of FP-Growth

The pros and cons related to FP-Growth algorithm are mentioned as under.

Pros

- The major advantage of the FP-Growth algorithm is that it takes only two passes over the data set.
- The FP-Growth algorithm compresses the data set because of overlapping of paths.
- The candidate generation is not required.
- The working of the FP-Growth algorithm is much faster as compared to the Apriori algorithm.

Cons

- The FP-Growth algorithm may not fit into the memory.
- The FP-Growth algorithm is expensive to construct. It consumes time to build. But once it is done with construction, itemsets can be read off easily.
- Enormous time is wasted when support threshold is high as pruning can be practiced only on single items.
- The process of calculating the support can be carried out only after the entire data set is added to the FP-Tree.

4. Conclusion

Today there is no second thought about the fact that frequent pattern mining has broad applications which incorporate software bug detection, clustering, recommendations, classification, and several other problems. The major utility of frequent pattern mining is an intermediate tool for providing pattern-centered insights for diverse problems. The purpose of the paper was intended to provide the reader with the complete working of the FP-Growth algorithm with an appropriate example. The

paper also elaborated the advantages and disadvantages associated with the FP-Growth algorithm. Frequent pattern mining has been well utilized to explore the past and now it has reached the stage where its application can effectively predict the future.

REFERENCES

[1] Jia-Dong Ren, Hui-Ling He, Chang-Zhen Hu, Li-Na Xu, Li-Bo Wang Mining Frequent Pattern Based On Fading Factor In Data Streams, Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15 July 2009.

[2] http://120.105.184.250/lwcheng/data_mining/fp-growth/FPGrowth.pdf

[3] Meera Narvekar, Shafaque Fatma Syed "An optimized algorithm for association mining using FP tree" International conference on advanced computing technologies and application (ICACTA-2015).

[4] V. Ramya, M. Ramakrishnan "Mining Association Rules Using Modified FP-Growth Algorithm" international journal for research in emerging science and technology, E-ISSN: 2349-7610.

[5] Rakesh Agrawal Ramakrishnan Shrikant, "Fast Algorithms for Mining Association Rules", IBMAlmaden Research Center.

[6] Chen Wenwei. Data warehouse and data mining tutorial [M]. Beijing: Tsinghua University Press. 2006.

[7] Hand David, MannilaHeikki,SmythPadhraic. Principles of Data Mining[M].Beijing:China Machine Press,2002.

[8] Morzy T.: Eksploracjadanych.(http://www.portalwiedzy.pan.pl/images/stories/pliki/ publikacje/ nauka/ 2007/03/N_307_06_Morzy.pdf, (2010).

[9] Pasztyła A.: Analizakoszykowadanychtransakcyjnych – celeimetody. (http://www.statsoft.pl/ pdf/artykuly/ basket.pdf, (2010).

[10] RapidMiner 4.3. User Guide. Operator Reference. Developer Tutorial. (http://docs.huihoo.com/ rapidminer/rapidminer-4.3-tutorial.pdf, (2010).

[11] Skrzypczak P. : Modelowaniewzorcówzachowańklientów Delikatesów Alma przywykorzys taniuregułasoc jacyjnych, Master Thesis, UniwersytetEkonomiczny, Wrocław (2010)

AUTHOR'S BIOGRAPHY

Dr. Gagandeep Jagdev, is a faculty member in Dept. of Computer Science, Punjabi University Guru Kashi College, Damdama Sahib (PB). His total teaching experience is above 10 years and has above 108 international and national publications in reputed journals and conferences to his credit. He is also a member of editorial board of several international peer-reviewed journals and has been active Technical Program Committee member of several international and national conferences conducted by renowned universities and academic institutions. His field of expertise is Big Data, ANN, Biometrics, RFID, Cloud Computing, Cryptography, and VANETS.

Citation: *Amanvir Kaur & Dr. Gagandeep Jagdev (2017). Analyzing Working of FP-Growth Algorithm for Frequent Pattern Mining, International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 4(4), pp.22-30, DOI: http://dx.doi.org/10.20431/2349-4859.0404003*