International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org || PP. 24-31

Research of Improved FP-Growth (IFP) Algorithm in Association Rules Mining

M.Sinthuja[1], Dr. N. Puviarasan[2], Dr. P.Aruna[3]
[1, 3](Department of Computer Science and Engineering,Annamalai University, Tamil Nadu)
[2](Department of Computer and Information Science, Annamalai University, Tamil Nadu)

Abstract: *Exploring frequent itemset from huge transactional database has been the most time consuming process of association rule mining.Up-to-date, various algorithms have been popularized in the area of frequent itemset generation. The FP-growth algorithms are the most familiar algorithms. FP-growth algorithm adopts tree structure for storing information producing in longer runtime. FP-growth algorithm faces the problem in complexity of space and time complexity. To develop the efficiency of the FP-growth algorithm, Improvised FP-growth (IFP) algorithm was introduced. In this paper, the benchmark databases considered for comparison are Chess, Connect and Mushroom. It was found out that the IFP-Growth algorithm outperforms FP-growth algorithms for all databases in the criteria of runtime and memory usage.*

Keywords: *Association Rule Mining Algorithm; Data Mining; FP-Growth; IFP-Growth; Minimum Support; Pruning*

I. Introduction

The era of automatized data collection applications and advanced database systems have given rise to tonnes of data for research in data warehouses, repositories and databases [1] [3]. Data should lead to useful information. The concept of datamining is used to extract useful and interesting information such as regularities, patterns, limitations and rules in databases. Data mining generates information about formerly accurate independent itemsets and their connection in a big database.

The items that are represented often in a database is called frequent itemsets. Frequent item sets are a group of items that occur often like notebook and pen in a transaction database. Frequent pattern mining bats for robust relationships in a data set. Frequent item sets may be transactional or relational.

This data industry has given birth to lot of data mining applications that can be applied in retail shopping for product showcasing and in internet for search of common words in pair. These applications work out for all items which have specific attributes-patient/symptoms, restaurants, /menu, internet/keywords, basket/products, railway/timings etc.

Market basket analysis tells about the products that are often bought in pair and it might provide data for promotion strategies. For example, purchase of a shampoo is accompanied with the purchase of conditioner. Promotion of shampoo might increase the sale of the conditioner. Promotion of both the items in combination might increase the profit.

II. Related Work And Contribution

Association rule learning is a predominant and thoroughly studied method for examining rules. R.S. Agrawal et al independently introduced Association rule mining [2] [3] [4]. Discovery of frequent item sets is the most pivotal aspect of data mining. Here frequent itemsets are sets of items that are purchased together frequently in a transaction. Mining frequent itemsets are improved by numerous new methods and an agile data structure is also introduced. Numerous algorithm for frequent itemset mining is discussed elaborately and the performance can be seen in the literature survey of frequent itemset mining [2] [13].

Apriori is one of the earlier algorithms which find frequent itemsets from big transactional databases [2]. It is a process to pinpoint frequent particular items in the transactional database. It in a later stage it extends them to larger itemsets that occur frequently. The frequent itemsets found by Apriori is used to explore association rules. This can be used in the application domain market basket analysis.

It is notable that the average search space for all frequent itemset is huge. Instead of exploring and calculating the support of the itemsets which are certainly impractical, numerous solutions have been introduced [1]. Partitioning and sampling methods of algorithms were introduced. The study includes algorithms like Apriori-TID [20], Apriori-Hybrid [5], Partition [6], Sampling [7], CARMA [8], ECLAT [9], Top-Down [10], FP-Growth [11] among others.

UT-Miner [11] is an advanced Apriori algorithm suited for sparse data. Sparse data are mostly varied and unique. It has a better performance with a structure of array. It is unreliable in terms of runtime and memory usage.

Predictive Apriori algorithm is another form of the association rule mining algorithms [12]. It is different from Apriori algorithm in both support and confidence. In this algorithm support and confidence are united into a single combination called predictive accuracy. {Support, Confidence}=> Predictive Accuracy. This algorithm performs with an increasing support threshold for a number of best rules.

Another achievement in the frequent pattern mining is FP-Growth algorithm. An effective algorithm called FP-Growth which forms a frequent pattern tree construction called FP-Tree [7]. In comparison to Apriori algorithm, candidate patterns are not searched and database is scanned only twice.

III. Proposed Research Work Improvised Fp-Growth Algorithm

In this research paper, an algorithm of Improvised FP-growth is introduced which is an advanced form of FP-growth algorithm. The structure of FP-growth algorithm has only two attributes of item name and count. In the proposed IFP-growth algorithm the structure of the node is modified. It has four attributes of item name, count, node link and flag. Here, node link plays an act of connect nodes with the similar item. It helps to search for particular node quickly. Thus the proposed algorithm quickly traverses the tree in relation to FP-growth algorithm. It reduces the runtime and memory of IFP-growth algorithm. IFP-growth algorithm shortens the database into IFP-tree. It also maintains the data between itemsets. IFP-growth is a divide and- conquer algorithm that uses two stages: build and mine. Structure of proposed IFP-growth algorithm is as follows:

An FP-tree is of three attributes namely:

i. Root node
ii. Child node
iii. Header Table

Attributes of each node are:

i. Item Name
   The names of the item are stored in this fie
ii. Count
   Number of times the item occur are updated in this field.
iii. Node Link
   Node-link that points the node that contains the same item name
iv. Flag
   The branch information are noted in this field.

Each entry of the item in header table consists of two Characteristics:

i. Item-name
ii. Head of node-link

Transaction Database is displayed in Table 1. The database is scanned early and each item in the database is counted. Items are assembled in descending sequence of support count. Items with lesser support value are pruned.

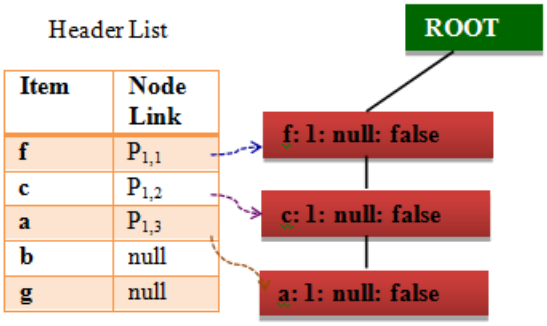
Table I: Transaction Database

TID	Item bought	Ordered item
100	d, c, a, f	f, c, a
200	g, c, a, f, e	f, c, a, g
300	b, a, c, f, h	f, c, a, b
400	g, b, f	f, b, g
500	c, b	c, b

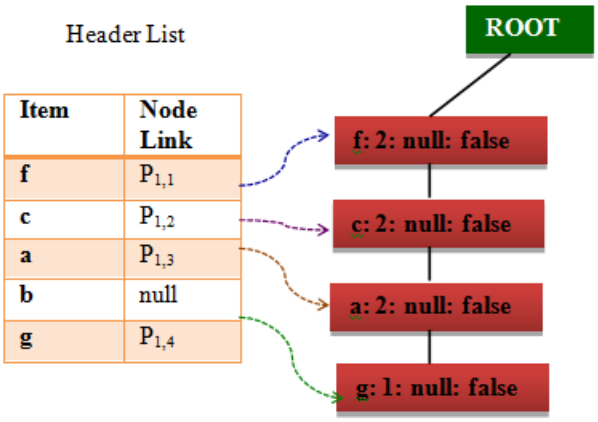
Table II: Prioritize the items

Items	Support
{f}	4
{c}	4
{a}	3
{b}	3
{g}	2
{d}	1
{e}	1
{h}	1

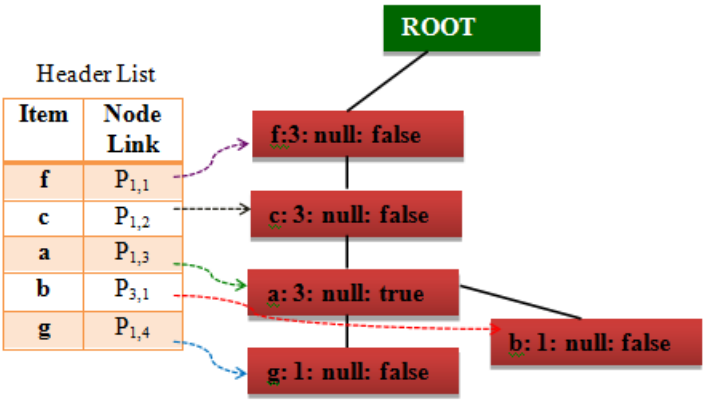
Step 1: Insertion of itemset {f, c, a}



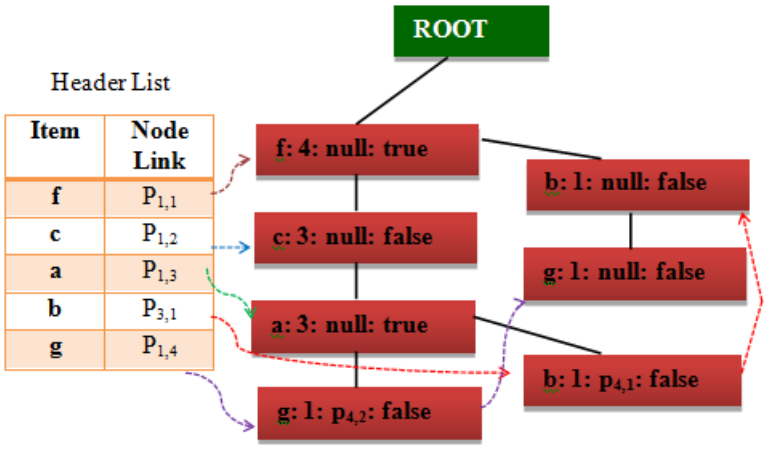
Step 2: Insertion of itemset {f, c, a, g}



Step 3: Insertion of itemset {f, c, a, b}



Step 4: Insertion of itemset {f, b, g}



Step 5: Insertion of itemset {c, b}

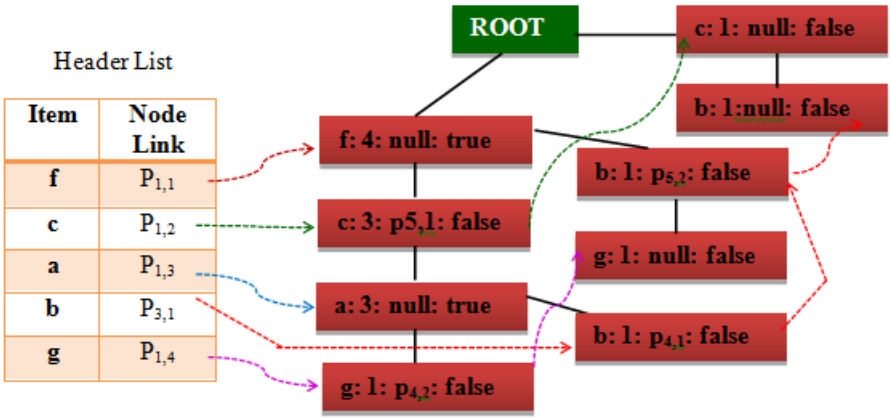


Fig. 1. Construction of IFP-tree

IFP-tree is constructed. Early, the proposed IFP-growth algorithm generates a node named 'root' with the tag value 'null'. For the second scan, it starts generating nodes for each transaction. For example, the first transaction of the database contains three items {f, c, a} based on the arrangement of descending order, building the early wing {(f:1) (c:1) (a:1)} in IFP-tree with three nodes. Here, item 'f' is the child of "root", item 'c' links to item 'f', item 'a' links to item 'c'. Now, the node link to 'null' is updated, as it is not related to any link and the flag value to 'false' is tagged, as it is not related to any branch nodes. Node link in the header table has to be updated for each item when an item is inserted in the IFP-tree. The second transaction contains four items {(f: 1) (c: 1) (a: 1) (g:1)} where four nodes are generated. Here, this branch shares the prefix 'f', 'c', 'a' with the existing path of transaction '01'. In this way the count of the nodes are incremented to '2'. For item 'g' new node is generated from item 'a'. The node link of item 'f',' c', 'a', 'g' to 'null' is updated as it does not have any link and fix its flag value. Third transaction of the database contains {f, c, a, b}. It is similar to that of previous transaction. Fourth transaction of the database contains {f, b, g}. In this scenario, the item 'f' shares its path from first transaction and increment its count to '4'. Node link value is noted down and flag value is fixed to true as it has branch node. For the insertion of item 'b' and 'g' a new node is generated from item 'f' where item 'b' is linked to item 'f' and item 'g' is linked to item 'b'. It is notable that the node link of item 'g' has to be updated to $P_{4,2}$and item 'b' to $P_{4,1}$ respectively along with an update in flag value. In case of fifth transaction the database contains {c, b}. In this scenario, new node has to be generated from root node as it does not share its path with existing transactions. Here, item 'c' is linked to 'root' and item 'b' is linked to item 'c'. The node links for all items are updated. The complete IFP-tree is portrayed in Fig 1.

Table III. Generated Frequent Patterns

Items	Conditional pattern base	ConditionalFP-tree	Frequent patterns generated
h	{f, c, b, a:1}	-	**No**
e	{f, c, a, g:1}	-	**No**
d	{ f, a, c:1}	-	**No**
g	{f,b:1} {f, c, a:1}	{f:2}	**{f, g:2}**
b	{c:1}{f:1} {f, c, a:1}	{f, c:2}	**{f, c, b:2}**
a	{f, c:3}	{f,c:3}	**{f,c,a:3}**
c	{f:3}	{f:3}	**{f, c:3}**
f	-	-	**No**

The system of exploring frequent itemsets from IFP-tree is from last to first using "Bottom Up" approach to explore frequent itemsets. Generated frequent itemsets is displayed in Table III. FP growth algorithm is great in achieving three important aspirations: database is scanned only two times and computational cost is decreased dramatically, Candidate itemsets are not generated. The search space is reduced by using divide and conquer approach. The limitation of FP-growth algorithm is crucial to use in incremental mining, as new transactions are added to the database.

IV. Experimental Results Analysis

Inthisexperiment,theperformanceofimproved FP-growthiscomparedwithFP-growthalgorithms.Inordertoshowthattheexperimentsarereasonable,theperformancebasedontwoimportantcriteria:runtimeandmemoryusage

A. Data Description

Theproposed algorithm iscarried outusingstandarddatasetsofvariousdomainsnamelyChess, Connect, Mushroom.TableIVdepictstheentireinformationofdatasetsusedfortheimplementationpurpose.

TABLE IV.DatasetDescription

Datasets	File Size (kb)	Tuples	Items	Type
Chess	335	3196	75	Dense
Mushroom	9105	8124	58	Dense
Connect	558	67557	116	Dense

B. Runtime Assessment

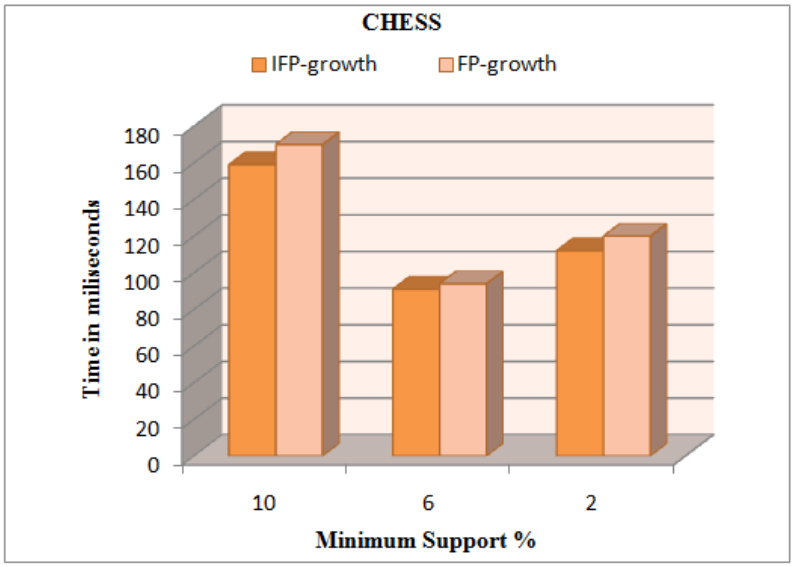


Fig. 2. Runtime of proposed IFP-growth algorithm

From theFig.2y-axisdisplayruntimeinmillisecondandx-axisdisplaythedifferent minimumsupportvalues.Itincludestheresult ofexecutiontimeof proposed IFP-growthandFP-growthalgorithmsonthedatasetChesswithdifferentminimumsupport.

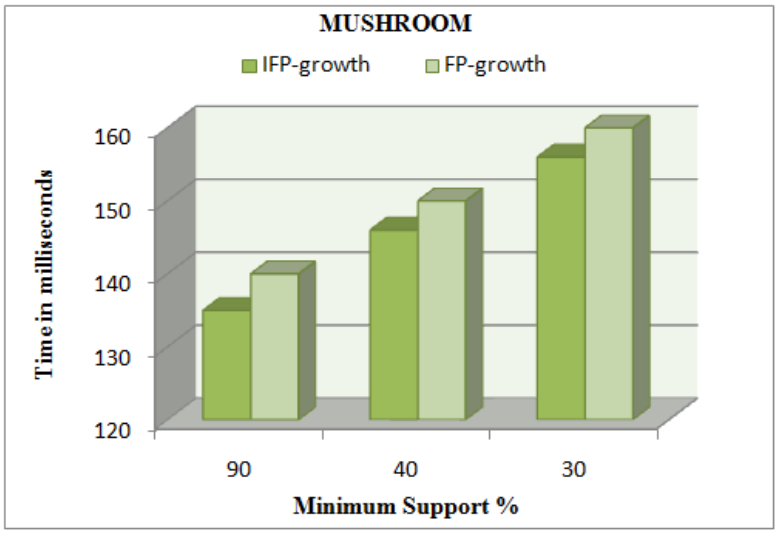


Fig. 3. Runtime of proposed IFP-growth algorithm

Runtimemeansthetotalexecutiontimebetweeninputandoutput.Here,theruntimeofIFP-growthis170msandFP-growthis150msforminimumsupport10.ItshowsthatIFP-growthisquickerthanFP-growthalgorithm.Astheminimum supportdiminishestheruntimeofFP-growthincreaseswhencomparedtoproposed IFP-growthalgorithm.

With reference to the Fig.3, itshowstheresultofrunningtimeof proposed IFP-growthandFP-growthalgorithmsonthedatasetMushroom withdifferentminimumsupport.TheruntimeofFP-growthis 160msandIFP-growthis156msforminimumsupport30. In some cases, the processing to time of FP-growth is more or less similar to the proposed IFP-growth algorithm. Even though the proposed IFP-growth algorithm is better than FP-growth algorithm.

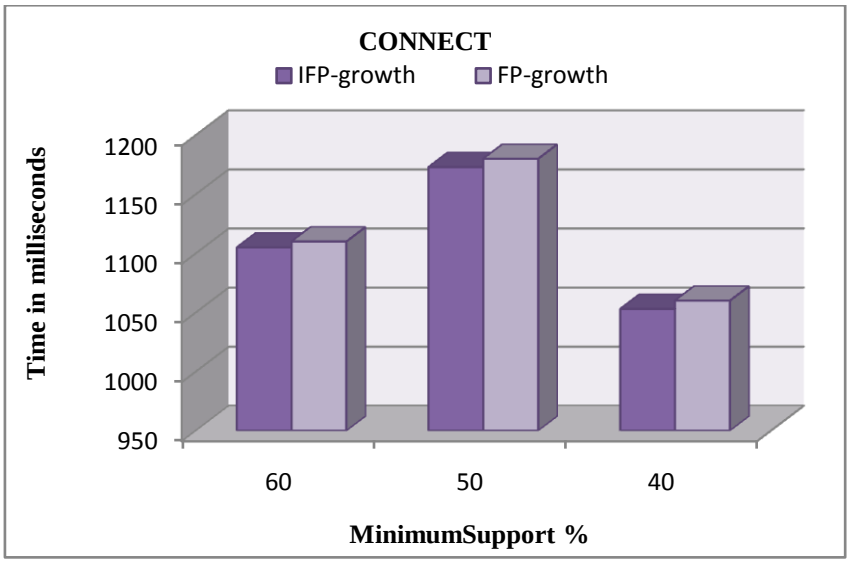


Fig. 4. Runtime of proposed IFP-growth algorithm

The graph illustrated in Fig 4 shows theruntimeof all algorithms for the dataset Connect.FP-growthis 1110msandIFP-growthis1105msfor the dataset Connect with minimumsupport60.ItshowsthatproposedIFP-growthisrapidthanFP-growthalgorithm.IFP-growthshowsit'sthe beston thedatasetsConnectforbothlowerandhigherminimumsupportwhencomparedtoFP-growthalgorithm.

C. Memory Consumption

Fig.5portraysthecomparativeresultsof IFP-growthandFP-growthalgorithmincaseofmemoryusage for the dataset Chess.Fromthegraphlongitudinalaxisshowsthememoryin MBand latitudinalaxisshowsthedifferentsupportthresholdvalues.ThememoryusageofFP-growthis 107.86 MBandproposed IFP-growthis101MB forminimumsupport 6 .ItshowsthatIFP-growthisrapidthanFP-growthalgorithm.IFP-growthrequireslessmemorythanFP-growthalgorithm while the minimum support is higher. When the minimumsupport is lower memoryrequirementforIFP-growthislesswhencomparedtoFP-growthalgorithm.

FromFig.6memoryusageofFP-growthalgorithmishigherascomparedtoIFP-growthalgorithmsincaseofbothhigherand lower minimumsupportforthedataset Mushroom. Here, the consumption of memory of proposed IFP-growth algorithm is 120MB and 123.9MB for FP-growth algorithm with the minimum support threshold values of 40.

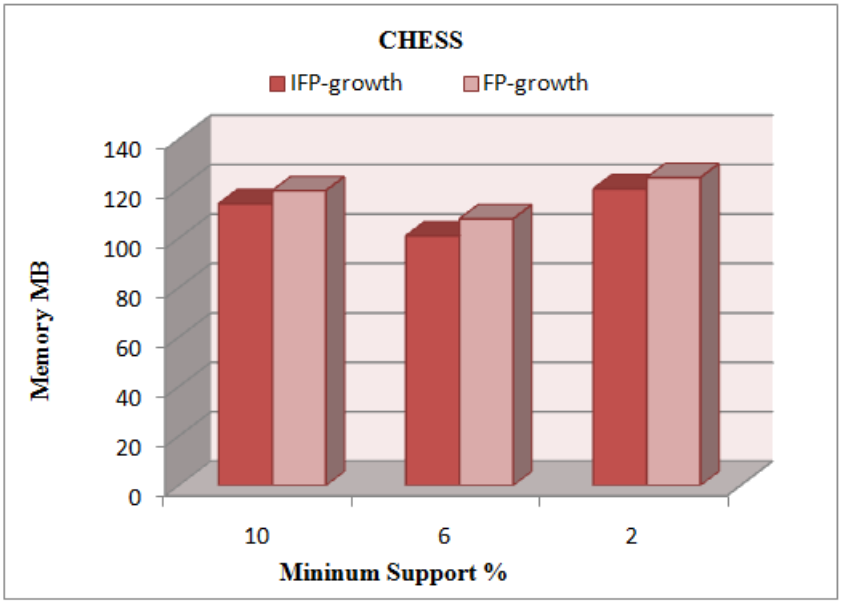


Fig. 5. Memory usage of proposed IFP-growth algorithm

From the graph it is come to know that theIFP-growthalgorithmgivesbetterperformanceascomparedtoFP-growthalgorithm.

The experimental results of memory usage all algorithms for the dataset Connect are reported in Fig 7. FP-growth algorithm consumes lot of memory when compared to other algorithm. Meanwhile, IFP-growth algorithm consumes lower memory space when compared to FP-growth algorithm. It is efficient in consumption of memory. For the minimum support threshold value of 60 the IFP-growth consumes 99MB FP-growth consumes 103.59MB. It is clearly observed from the graph that the proposed IFP-growth algorithm is efficient than FP-growth algorithm.

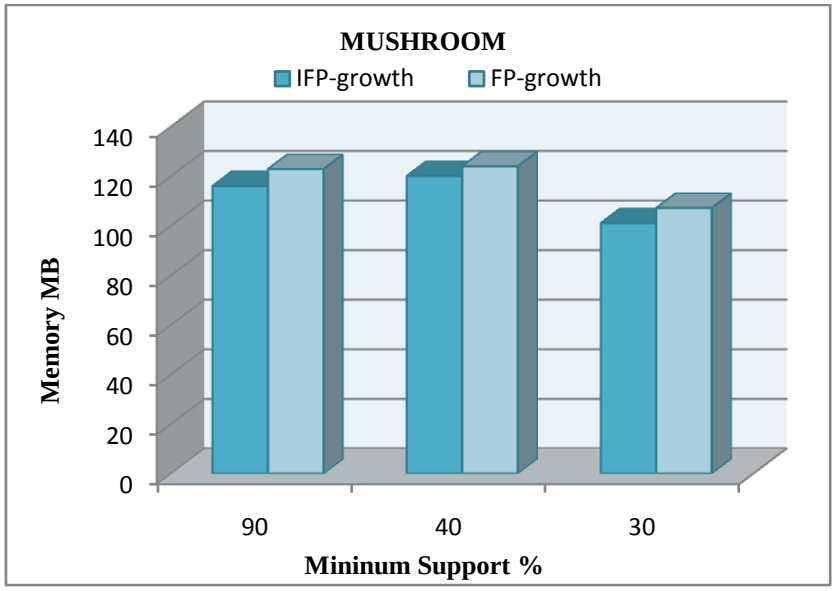


Fig. 6. Memory usage of proposed IFP-growth algorithm

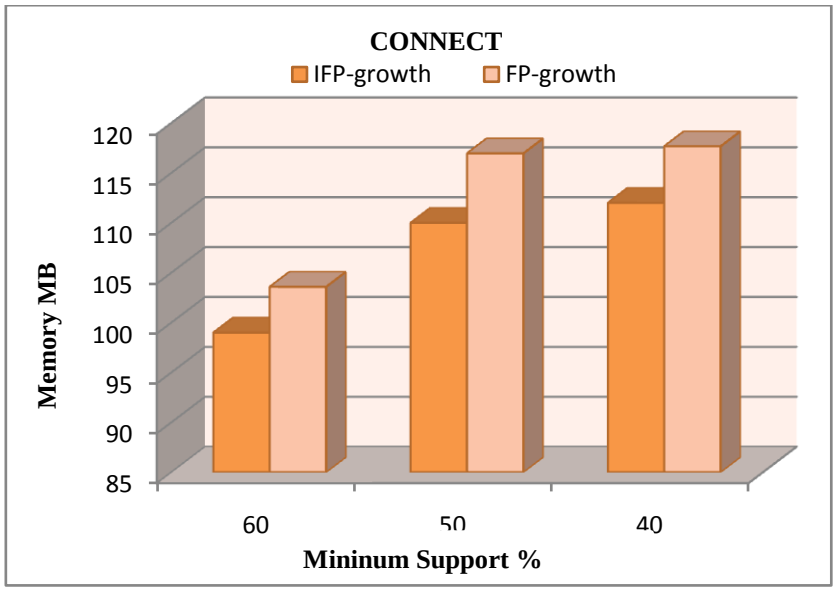


Fig. 7. Memory usage of proposed IFP-growth algorithm

V. Conclusion

In this research paper, improvised version of Frequent Patten growth (IFP) is proposed for exploring frequent itemsets. In the light of results observed above from the proposed Improvised Frequent Pattern growth algorithm. It has been finalized that the proposed Improvised Frequent Pattern growth algorithm is better than the FP-growth algorithm for large benchmark datasets like Chess, Mushroom and Connect for mining frequent itemsets. It is determined from the experimental evaluation that the proposed Improvised FP-growth is a suitable algorithm for dense datasets with respect to speed and memory consumption. It is more efficient for dense datasets. The proposed Improvised Frequent Pattern growth algorithm is found to be the best algorithm than FP-growth algorithm in terms of processing time and consumptionof memory.

Acknowledgment

The authors would like to thank the Department of Computer Science and Engineering, Annamalai University for providing the lab facilities. We gratefully acknowledge the funding agency, the University Grant Commission (UGC) of the Government of India, for providing financial support. Also, authors would like to thank to Dr. N. Puviarasan and Dr. P.Aruna for guiding to do new analysis for revising the article.

REFERENCES

[1] S. Langenfeld, *"CRM and the Customer Driven Demand Chain,"* vol. *2004*, 2004.
[2] R. Agarwal, T. Imielinski, and A. Swami, "*Mining Association Rules Between Sets of Items in Large Datasets,"* in Proceedings of the ACM SIGMODConference on Management of Data. Washington, D.C.: ACM, 1993, pp. 207-216.
[3] J. Schafer, J. Konstan, and J. Riedl, "*Electronic Commerce Recommender Applications*," Journal ofData Mining and Knowledge Discovery, *vol. 5*, pp. 115-152, 2001.
[4] W. Lin, "*Association Rule Mining for Collaborative Recommender Systems*," in Computer Science.
[5] R. Agrawal and R. Srikant, "*Mining Sequential Patterns*," in Proceedings of the 11th IEEEInternational Conference on Data Engineering. Taipei, Taiwan: IEEE, 1995.
[6] A. Savasere, E. Omiecinski, and S. B. Navathe, "*An Efficient Algorithm for Mining Association Rules in Large Databases*," in Proceedings of the 21ndInternational Conference on Very Large Databases. Zurich, Switzerland, 1995, pp. 432-444.
[7] H. Toivonen, "*Sampling Large Databases for Association Rules,"* in Proceedings of the 22nd International Conference on Very Large Databases. Mumbai, India, 1996, pp. 134-145.
[8] C. Hidber, "*Online Association Rule Mining,"* in Proceedings ACM SIGMOD InternationalConference on Management of Data. Philadephia, Pennsylvania: ACM, 1999, pp. 145-156.
[9] M.J.Zaki, "*Scalable Algorithms for Association Mining*," IEEE Transactions on Knowledge and DataEngineering, *vol. 12*, pp. 372-390, 2000.
[10] J. Han, J. Pei, and Y. Yin, "*Mining Frequent Patterns Without Candidate Generation,"* in Proceedings of the2000 ACM SIGMOD International Conference onManagement of Data. Dallas,TX: ACM, 2000.
[11] J. Hipp, U. Guntzer, and G. Nakhaeizadeh, *"Algorithms for Association Rule Mining -- A general Survey and Comparision*," in Proceedings of ACMSIGKDD, *vol. 2*: ACM, 2000, pp. 58-64.
[12] M.Sinthujaet al, *"Comparative Analysis of Association Rule MiningAlgorithms in Mining Frequent Patterns*"International Journal of Advanced Research in Computer Science, *vol 8 (5),* May-June 2017,1839-1846.
[13] B. Goethals, "*Survey on Frequent Pattern Mining," vol. 2004*, 2003.