International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

Present State-of-The-Art of Association Rule Mining Algorithms

N.Satyavathi, B.Rama, A.Nagaraju

Abstract*: A Data mining is the method of extracting useful information from various repositories such as Relational Database, Transaction database, spatial database, Temporal and Time-series database, Data Warehouses, World Wide Web. Various functionalities of Data mining include Characterization and Discrimination, Classification and prediction, Association Rule Mining, Cluster analysis, Evolutionary analysis. Association Rule mining is one of the most important techniques of Data Mining, that aims at extracting interesting relationships within the data. In this paper we study various Association Rule mining algorithms, also compare them by using synthetic data sets, and we provide the results obtained from the experimental analysis.

Keywords: Association Rule, Apriori, Pattern-Growth, Support.

I. INTRODUCTION

Data mining is a technique of digging knowledge or patterns from a large amount of data. Different kinds of patterns that can be mined are Characterization and Discrimination, Cluster Analysis, Classification and Prediction, Association Analysis, Evolutionary Analysis. Association Analysis is also called Association Rule mining, an important technique which mines interesting relationships within the data. Various algorithms have been developed for mining association rules (Satyavathi et al., 2015) .

A.Association Rule Mining Problem

An association rule is represented as X=>Y, where X, Y are called items or item sets;X is called as antecedent and Y is called Consequent.For every association rule, two important measures are defined: Support(s) and Confidence(c). Naturally, size of the database will be very large and there is a chance of getting many numbers of Association rules, some of them may not be interested to the user. Hence, support and confidence are thresholds are predefined by the user before the mining process, to drop those rules which are not interested or useful.

Revised Manuscript Received on October 30, 2019.

* Correspondence Author

N.Satyavathi,CSE department,Vaagdevi college of Engineering, Warangal,India. Email: satyanadendla15@gmail.com

Dr.B.Rama, Computer Science, Kakatiya University, Warangal, India. Email: rama.abbidi@gmail.com

Dr.A.Nagaraju, Computer Science,School of mathematics, statistics and computational science,Central University of Rajasthan, Ajmer, India. Email: nagaraju@curaj.ac.in

Suppose X=>Y is an Association rule. Support(s) is the percentage or fraction of records that contain both the items X & Y to the total number of records in the database. Support(s) can be calculated as follows:

Support(X=>Y) = (No. Of records containing both X & Y)/ (Total No. of records in Database)

For example Support(X=>Y) = 0.1%, means only 0.1% percent of transactions contain both the items X and Y.

Confidence(c) is defined as the percentage or fraction of records that contain both the items X & Y to the total number of records containing X. Confidence of an association rule can be calculated as follows:

Confidence(X=>Y) = (No. of records containing X &Y)/ (No. Of records containing X)

For example confidence of an association rules X=>Y is 60%, means that there is a 60% chance of buying X & Y items together in the future.

The problem of Association rule mining is to find out association rules that satisfy the predefined measures: support & Confidence. Association Rule mining process involves two

Steps: 1) Finding frequent item sets that satisfy predefined minimum support. Frequent item sets are defined as the set of items that frequently occur together in the given transaction.

2) Generating association rules that satisfy predefined minimum confidence. Association rule mining process is shown in the fig.1.

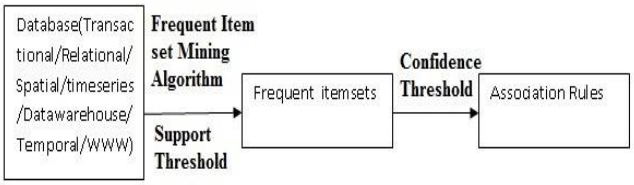

Fig. 1. The process of Association Rule Mining

Generating association rules from frequent item sets is a direct method. Hence, most of the researchers concentrated on developing an efficient algorithm for finding frequent item sets. In 1994, Agrawal and Srikant presented an efficient algorithm called Apriori for mining frequent item sets. Later many algorithms based on Apriori have been developed.Apriori algorithm uses candidate set generation approach to mine frequent item sets.Candidate set generation approach is costly, when database contains more number of transactions.Hence to overcome this approach a Pattern-Growth mining method called “FP-Growth was proposed.

Later many approaches based on FP-Growth are developed for mining Frequent itemsets.In this paper we provide overview of each apriori-based association rule mining algorithms and Pattern-Growth algorithms, we compare the algorithms by using synthetic data sets, and we provide the results obtained.

The rest of the paper is organized as follows: Describes the apriori algorithm, provides the overview of all apriori-based algorithms, gives the characteristic comparison of various apriori-based algorithms, performance comparison by using different synthetic data sets, Later describes the Pattern-Growth algorithm, provides the overview of all Pattern-Growth algorithms, gives the characteristic comparison of various Pattern-Growth algorithms discussed and provides performance comparison by using different synthetic data sets and then concludes the paper.

II. Overview of Apriori Algorithm

Apriori algorithm [1] was first proposed by Agrawal R and Srikant R for mining frequent item sets. Apriori uses candidate generation method and also pruning technique to find the frequent item sets. The first database is scanned to get the support count of each item. Items whose support count is less than predefined minimum support are pruned from the database. Items whose support is equal to or greater than predefined minimum support are called as candidate item sets or frequent 1- item sets. Using these frequent 1 –item sets all possible 2-item sets are generated by using join operation. Once again database is scanned to get the support count of each possible 2-item set. Item sets are pruned whose support count is less than predefined minimum support, remaining item sets are called as frequent 3-item sets or candidate 3-item sets. This process is continued until no new item sets are generated. Later on, many algorithms have been developed based on Apriori.

III. Apriori-based Association Rule mining algorithms

Apriori algorithm is found to be efficient in mining frequent item sets. Many improvements have been made to the Apriori and many other algorithms have developed with little modifications. Such algorithms are discussed below.

A. Apriori-Tid Algorithm

Apriori-TID [1] is a variant of the Apriori algorithm. It was proposed by Rakesh Agrawal and Ramakrishnan Srikant in the same article in which Apriori is proposed. A method used by AprioriTid is different when compared with Apriori but outputs produced by the two algorithms are same.

B. Apriori-Hybrid Algorithm

Apriori-Hybrid algorithm [1] is also proposed by Rakesh Agrawal and Ramakrishnan Srikant in the same article in which Apriori is proposed. In the earlier passes Apriori works better than Apriori-Tid but in later passes Apriori-Tid works better than Apriori. Hence, a hybrid algorithm called Apriori Hybrid can be designed that works as Apriori in the earlier passes and as Apriori-Tid in later passes. Apriori-Hybrid is very efficient in case of large databases.

C. Direct Hashing and Pruning Algorithm

This Direct Hashing and Pruning [2] is proposed by Jong Soo Park, Ming- Syan Chen and Philip S.Yu. The following procedure is followed in this technique: The algorithm finds frequent 1- item sets by scanning the given database. At the same time, possible frequent 2-item sets are generated and hashed them to a hash table. Now, this hash table is used to reduce the number of candidate item sets and to find the final set of frequent item sets.

D. Dynamic Item Set Counting Algorithm

Dynamic item set counting [3] is developed by Sergey Brin, Rajiv Motwani, Jeffry D.Ulman and Shalom Tsur. The basic algorithm is as follows: Given database is divided into k number of partitions. In the first partition, start counting the 1-item sets. Next, in the second partition, start counting 1-itemsets and also start counting the 2-item sets by making use of 1-itemsets obtained from the first partition. . Now in the third partition, start counting the 1-item sets and also start counting 2-item sets, 3-item sets by using the results obtained from first and second partitions.

E. Partition Algorithm

Partition algorithm [4] was proposed by Ashok Savasere, Edward Omiecinski, Shamkant Navathe.The main idea behind the algorithm is it divides the given database into equal size partitions and apriori property is used for each partition and finds large item sets in each partition. To get the final set of frequent item sets for the given database performs union operation on large item sets obtained from each partition. The efficiency of the algorithm depends on database size, partition size and number of local large item sets generated.

F. Eclat (Equivalence CLAss Transformation) MaxEclat, Clique, MaxClique, TopDown and ApprClique algorithms

Zaki. M [5] proposed six efficient algorithms : Eclat (Equivalence CLAss Transformation), MaxEclat, Clique, MaxClique, TopDown and ApprClique for fast mining of frequent item sets.

The main idea of these algorithms is as follows:

1) Frequent 1- item sets and frequent 2-item sets are computed by using a vertical tid-list database format and frequent 2-item sets are computed by simply performing tid-list intersections.

2) On the set of frequent 2-item sets, sub lattices are generated by using any one of the relation: prefix-based equivalence relation or maximal-clique-based pseudo equivalence relation.

3)As a final step to generate frequent item sets, these sub lattices are processed one at a time by making use of any one of the search procedure: Bottom-up/Top-Down/Hybrid.

Depending on the relation used in generating sub lattices and search strategy, Mohammed Zaki categorized the algorithms into 6 types, as shown in table 1.

Table –I: Algorithms for fast mining of frequent item sets

Algorithm	Relation used for generating sub lattices	Search strategy used for enumeration
Eclat	Prefix-based equivalence relation	Bottom-up search
MaxEclat	Prefix-based equivalence relation	Hybrid search
Clique	maximal-clique-based pseudo equivalence relation	Bottom-up search
MaxClique	maximal-clique-based pseudo equivalence relation	Hybrid search
TopDown	maximal-clique-based pseudo equivalence relation	TopDown
ApprClique	maximal-clique-based pseudo equivalence relation	TopDown

IV. CHARACTERISTIC COMPARISON OF APRIORI-BASED ASSOCIATION RULE MINING ALGORITHMS

Apriori-based algorithms are compared by considering various characteristics like database format used, the data structure used, search procedure followed in finding frequent item sets and a total number of scans required during the process and shown in table2.

Table- II: Characteristic comparson of Apriori-based algorithms

Algorithm	Database format	Database structure	Search Procedure	Scans (Where n is the size of longest frequent item set)
Apriori	Horizontal	Array	Bottom-up	n
Direct Hashing and Pruning	Horizontal	Hash tree	Bottom-up	n
Dynamic Item set Counting	Horizontal	Prefix tree	Bottom-up	≤ n
Partition	Vertical	Array	Bottom-up	2
EClat	Vertical	Array	Bottom-up	≥3
Max Eclat	Vertical	Array	Bottom-up & Top-Down	≥3
Clique	Vertical	Array	Bottom-up	≥3
MaxClique	Vertical	Array	Bottom-up & Top-Down	≥3
TopDown	Vertical	Array	Top-Down	≥3
AprClique	Horizontal	Hash trees	Top-Down	Only once

V.PERFORMANCE COMPARISON OF APRIORI-BASED ALGORITHMS

Performance of various Apriori-based algorithms is compared with respect to the Minimum support (%) and Execution Time (seconds) using synthetic datasets shown in table3 , the results obtained are shown in a tabular form and also shown in a graphical form.

Table- III: Dataset Description

Dataset	Number of Transactions	Average Transaction size
T10.I2.D100K	100,000	10
enterprise WEB	6200(5weeks data)	-
T20.18.D400K	400,000	20

A.Comparison of Apriori, AprioriTid, AprioriHybrid

Performance of Apriori, AprioriTid, and AprioriHybrid is tested using T10.I2.D100K dataset. Results obtained are recorded in Table4 and graphically shown in Fig. 2.

Table- IV: Minimum Support Vs Execution Time

Minimum support (%)	Execution time		
	Apriori	*AprioriTid*	*AprioriHybrid*
0.25	19.70	38.59	14.07
0.33	18.72	37.01	13.09
0.50	15.14	34.04	11.12
0.75	14.38	17.80	10.36
1.00	10.99	14.81	8.58
1.50	5.20	7.21	6.40
2.00	4.42	5.43	4.42

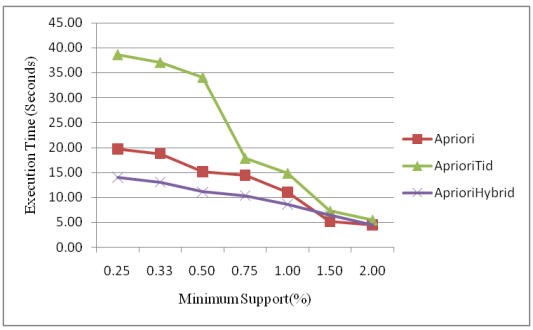

Fig.2. Performance comparison of Apriori,Apriori-Tid and Apriori-Hybrid

Fig. 2 shows the performance of AprioriHybrid relative to Apriori and AprioriTid for the T10.I2.D100K dataset. AprioriHybrid performs better than Apriori and AprioriTid but with 1.5% support, AprioriHybrid does a little worse than Apriori.

B. Comparing the performance of Apriori and DHP

Performance of Apriori and DHP is analyzed by using the data from logs of the WEB server of an enterprise WEB.

Five weeks visit records are extracted and 6200 user transactions are obtained. Performance of Apriori and DHP with respect to minimum support and execution time is shown in table 5 and graphically shown in Fig. 3.

Table –V: Minimum Support Vs Execution Time

Support Threshold (%)	Execution time Ratio	
	Apriori	***DHP***
0.5	1642.50	703.91
1	1407.61	631.80
1.5	1295.68	597.89
2	1205.82	575.61
3	1147.62	531.86
5	1023.69	485.83
6	954.30	471.84

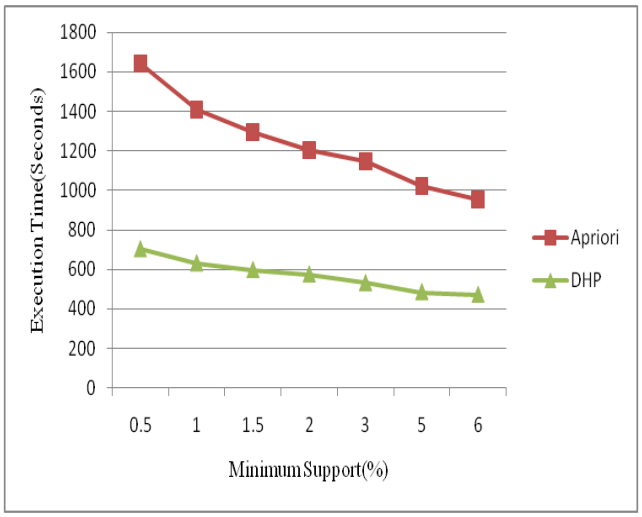

Fig. 3. Performance of Apriori and DHP with respect to Minimum Support (%) & Execution time (Seconds)

Performance of DHP is better than Apriori in all cases. At 0.5 minimum support value there is a lot of difference in the execution time of Apriori and DHP.

C.Comparison of Apriori and DIC

Apriori and DIC were run on synthetic dataset and results obtained are recorded in table6 and graphically shown in Fig.4.

Table –VI: Minimum support (%) Vs Execution Time (seconds)

Support Threshold (%)	Execution time	
	Apriori	***DIC***
0.005	270.82	212.2
0.007	198.15	144.71
0.01	135.72	108.14
0.012	118.22	111.33
0.015	55.8	66.14
0.017	52.1	52.1
0.02	43.13	51.73

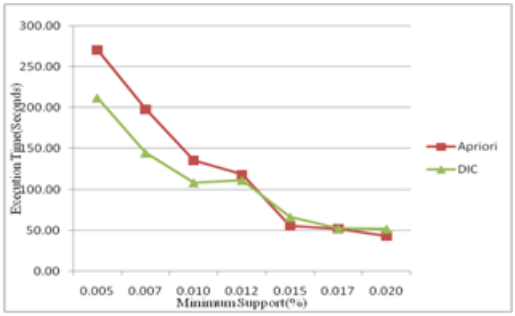

Fig.4. Performance comparison of Apriori and DIC

Performance of DIC is faster when compared to Apriori in almost all cases, with the support of 0.005 DIC is running 30% faster than Apriori. But with a support of 0.015, the performance of Apriori is better when compared with the performance of DIC.

D.Comparison of Apriori,Partition,Eclat,Clique,MaxEclat,MaxClique

Performance of Apriori, Partition, Eclat, Clique, MaxEclat, and MaxClique is compared by using T20.18.D400K dataset and the results obtained are recorded in table 7 and graphically shown in Fig.5.

Table- VII: Execution time (Seconds) of various algorithms with respect to Minimum Support (%)

Algorithm	Minimum Support(%)			
	0.25	***0.5***	***0.75***	***1***
Apriori	684.39	372.34	198.58	63.82
Partition	400.70	308.51	216.31	205.67
Eclat	315.60	205.67	148.93	131.20
Clique	301.48	198.58	148.93	131.20
MaxEclat	269.50	180.85	145.39	134.75
MaxClique	241.13	170.21	141.84	120.56
TopDown	585.10	407.80	173.75	134.75
ApprClique	627.66	453.90	273.05	74.46

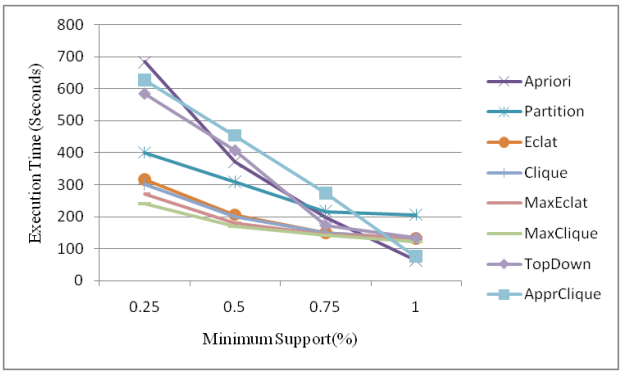

Fig.5. Performance comparison using T20.18.D400K dataset

Performance of MaxClique is much better in all the cases. Clique performs slightly better than Apriori, Partition, Top Down, MaxEclat is better than Eclat. MaxClique is faster when compared to all the algorithms.

VI. OVERVIEW OF PATTERN-GROWTH ALGORITHMS

Pattern algorithms are developed to overcome the drawback of Apriori & Apriori-based algorithms. The main idea of Pattern-Growth Algorithms is to use divide and conquer approach for mining frequent item sets, compress the given database by using some efficient data structure and then continue mining frequent item sets from the compressed structure.

VII. PATTERN-GROWTH ALGORITHMS

Pattern Growth algorithms[6],[7] are efficient and scalable when compared to Apriori-based algorithms. First developed Pattern Growth algorithm is "FP-Growth". Later many algorithms are developed which are discussed in next section.

A.FP-Growth Algorithm

FP-Growth algorithm [8] mines frequent item sets without candidate generation approach. The main idea of FP-Growth is as follows: 1) constructs FP-tree (Frequent pattern tree) for the given database. Header Table is maintained for the FP-tree. 2) Consider items from the Header Table (from the last item), find conditional pattern base, construct conditional pattern tree for each item.3) Finally Frequent item sets are mined from the conditional pattern tree.

B. H-MINE

The main idea of HMine algorithm [9] is, for the given database HStruct structure is constructed. Mining is performed on HStruct structure to find frequent item sets. H-mine is efficient and scalable at mining very large databases.

C.Patricia mine

A compressed patricia trie [10] is used to represent the database. Patricia trie is a modification of standard trie.To find the frequent item sets, patricia trie is traversed using a technique called "item-guided traversal".

D.RElim

This algorithm is similar to H-mine.RElim [11] finds frequent items,starting with a preprocessing step and then recursively eliminates least frequent item.RElim uses a "linked list" data structure in the process of mining frequent item sets.

E. PPV

PPV[12] is an vertical mining algorithm.It uses a tree structure called PPC-tree(Preorder Postorder Coded-tree) to represent the database.The tree is traversed using preorder and postorder techniques and each node in the tree is assigned a pre-order and post-order code,based on which,each frequent item can be represented by Node-list.

F. Prepost

PrePost [13] uses a datastructure called "N-List" for representing frequent items. The difference between PPV and Prepost is,PPV uses candidate generation approach for mining frequent item sets but Prepost mines frequent item sets without candidate generation approach.

G. NSFI (N-list and subsume based algorithm for mining Frequent item sets)

It is an enhanced version of Prepost algorithm. This algorithm creates N-List using a hash table and uses an enhanced N-intersection algorithm for mining frequent item sets. NSFI algorithm [14] outperforms Prepost in terms of runtime and memory usage.

H. FIN (Fast mining of frequent item sets using Node-sets)

FIN[15] algorithm is used for fast mining of Frequent item sets.FIN approach is: Given database is pre-processed and infrequent items are removed .POC- tree (Pre order coded tree) is constructed for the pre-processed data. From which Frequent-1 item sets and frequent 2-iem sets are determined. Set Enumeration tree is constructed for each frequent 2-item set to find frequent k-item sets.

VIII. CHARACTERISTIC COMPARISON OF PATTERN-GROWTH ALGORITHMS

Pattern-Growth algorithms are compared by considering various characteristics like database format used, the data structure used, search approach followed in finding frequent item sets and time required during th
e process, shown in table8.

Table- VIII: Characteristic comparison of Pattern-Growth Algorithms

Algorithm	Data structure used	Approach
FP-Growth	FP tree	Divide and Conquer
H-Mine	H-Struct	Divide and Conquer
Patricia mine	Particia trie	Divide and Conquer
RElim	Singly Linked List	Divide and Conquer
PPV	PPC tree, Node-list	Divide and Conquer
Prepost	PPC tree, N-List	Divide and Conquer with subsume index concept
NSFI	PPC tree, N-List	Divide and Conquer
FIN	POC tree, Nodeset	Divide and Conquer

IX. PERFORMANCE COMPARISON OF PATTERN-GROWTH ALGORITHMS

Performance of various Pattern-Growth algorithms is compared with respect to the Minimum support (%) and Execution Time(seconds) using various datasets shown in table 9, the results obtained are shown in a tabular form and also shown in a graphical form.

Table- IX: Dataset Description

Dataset	Number of Transactions	Number of attributes	Source
T40I10D100k	100,000	4	Artificially Generated
Chess	28056	6	UCI Machine Learning Repository

T10I4D100K	100,000	4	Artificially Generated
T25I20D100K	100,000	4	Artificially Generated
Mushroom	8124	22	UCI Machine Learning Repository
connect	67557	42	UCI Machine Learning Repository

A. Comparison of FP-Growth and HMINE

Performance of FP-Growth and HMINE is tested using T4010D100k dataset. Results obtained are recorded in Table 10 and graphically shown in Fig. 6.

Table –X: Minimum support (%) Vs Execution Time (milli seconds)

Support Threshold (%)	Execution time in milli seconds	
	FP-Growth	*HMINE*
5	1032	391
10	375	218
20	266	160
25	140	156

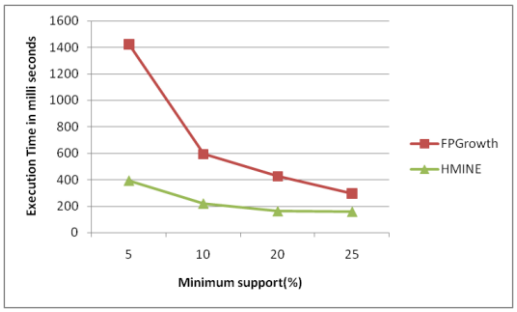

Fig.6. Performance comparison using T4010D100k dataset

From the experimental analysis, we can conclude that Performance of HMINE is better than FP-Growth. When minimum support value is 5%, there is a lot of variation in the running time of FP-Growth and HMINE.

B. Comparison of FP-Growth and Patricia mine

Performance of FP-Growth and Patricia mine is tested using "chess" dataset. Results obtained are recorded in Table 11 and graphically shown in Fig.7.

Table -XI: Minimum support (%) Vs Execution Time (seconds)

Minimum Support %	Execution Time in Seconds	
	Patricia mine	*FP-Growth*
20	7133.67	8003.26
25	6256.83	7189.23
30	5511.92	6245.78
35	4633.81	5567.47
40	3954.24	4953.24
45	3275.93	4210.85

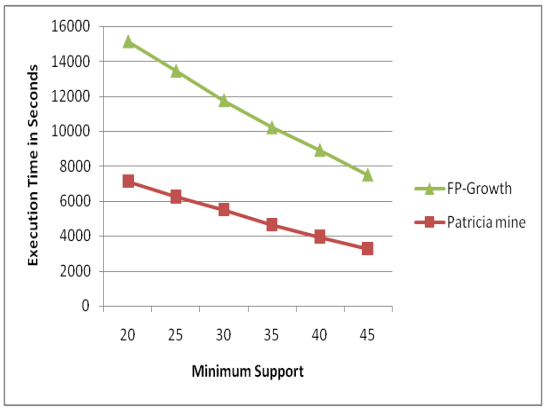

Fig. 7. Performance comparison using "chess" dataset

From the experimental analysis, we can conclude that Performance of Patricia mine is better than FP-Growth for all the support values.

C. Comparison of FP-Growth and *Recursive Elimination*

Performance of FP-Growth and Recursive Elimination algorithm is tested using T10I4D100K dataset. Results obtained are recorded in Table 12 and graphically shown in Figure 8.

Table-XII: Minimum support (%) Vs Execution Time (seconds)

Minimum Support (%)	Execution time in Seconds	
	Recursive Elimination	*FP-Growth*
5	0.465268	0.604799
10	0.200576	0.321492
15	0.131207	0.224248
20	0.103727	0.196747
25	0.0762267	0.173898
30	0.0719919	0.160361
35	0.0537936	0.146814
40	0.0449078	0.142579
45	0.0499542	0.133693
50	0.0410684	0.13411
55	0.0321825	0.134515
60	0.0320987	0.120978

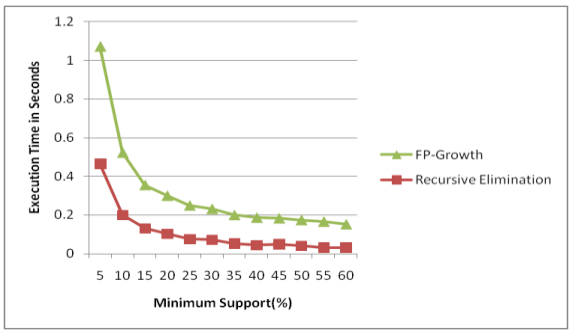

Fig.8. Performance comparison using T1014D100k dataset

From the experimental analysis, we can conclude that Performance of Patricia mine is better than FP-Growth for all the support values. When a support value is 5%, the performance of Recursive elimination algorithm is far better than FP-Growth.

D. Comparison of FP-Growth and PPV

Performance of FP-Growth and PPV algorithm is tested using T25I20D100K dataset. Results obtained are recorded in Table 13 and graphically shown in Figure 9.

Table- XIII: Minimum support (%) Vs Execution Time (seconds)

Minimum Support %	Execution Time in Seconds	
	FP-Growth	*PPV*
3	742.107	635.218
4	717.409	573.212
5	650.061	553.989
6	625.495	508.01
7	435.276	349.844

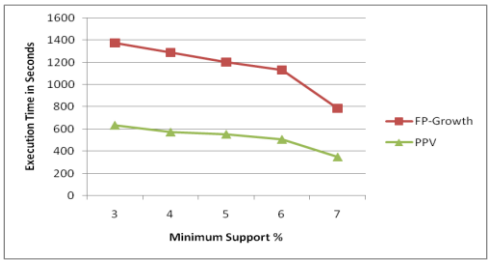

Fig.9. Performance comparison using T25I20D100K dataset

From the experimental analysis, we can conclude that Performance of PPV algorithm is better than FP-Growth for all the support values.

E. Comparison of Prepost and NSFI

Performance of Prepost and NSFI algorithm is tested using "Mushroom" dataset. Results obtained are recorded in Table 14 and graphically shown in Figure 10.

Table-XIV: Minimum support (%) Vs Execution Time (seconds)

Minimum Support %	Execution Time in Seconds	
	Prepost	*NSFI*
5	7.98913	4.72826
10	1.1413	0.652174
15	0.326087	0.163043

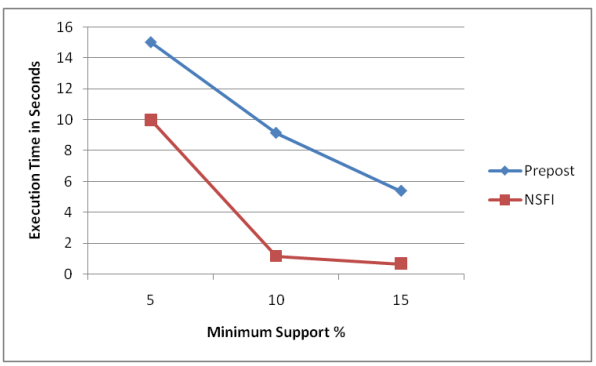

Fig.10. Performance comparison using "Mushroom" dataset

From the experimental analysis, we can conclude that Performance of NSFI algorithm is better than PrePost algorithm for all the support values.

F. Comparison of Prepost and FIN

Performance of Prepost and FIN algorithm is tested using Connect dataset. Results obtained are recorded in Table 15 and graphically shown in Figure 11.

Table- XV: Minimum support (%) Vs Execution Time (seconds)

Minimum Support %	Execution Time in Seconds	
	Prepost	*FIN*
45	37.7358	1.50943
50	20.3744	1.25786
55	11.0692	1.00629
60	6.03774	1.00629

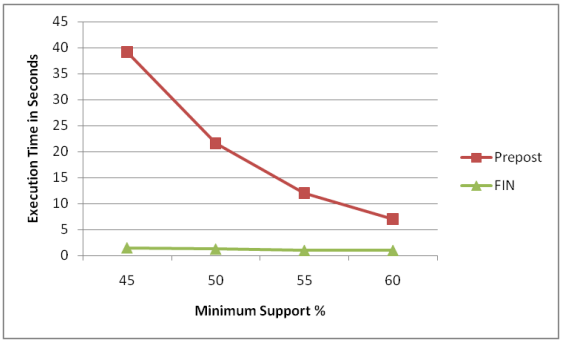

Fig.11. Performance comparison using "connect" dataset

From the experimental analysis, we can conclude that Performance of PPV algorithm is better than FP-Growth for all the support values. As the support value increases the difference in execution time is also reduced.

X. CONCLUSION AND FUTURE SCOPE

We have provided an overview of various Apriori-based and Pattern Growth algorithms used for mining frequent items, we also compared them by considering different characteristics and finally we analyzed their performance by using synthetic datasets. Summary of the in-depth analysis of Apriori-based and Pattern Growth algorithms is done which made a significant contribution to the search for efficient algorithm for mining frequent item sets.

In all, many algorithms have contributed to mine frequent item sets , however yet there are scopes to increase the efficiency of algorithms, development of new algorithms, and to reduce the time required in Execution.

REFERENCES

1. Agrawal, R. and Srikant, R. Fast Algorithms for Mining Association Rules, In Proceedings of the 20th International Conference on very Large Databases, Santiago, Chile, 1994, pp. 487-499.
2. J.S. Park, M. Chen, and P.S. Yu, An Effective Hash Based Algorithm for Mining Association Rules, ACM SIGMOD Int'lConf. Management of Data May 1995.

3. R. Motwani, J.D. Ullman, and S. Tsur S. Brin, "Dynamic Itemset Counting And Implication Rules For Market Basket Data," *ACM SIGMOD, International Conference on Management of Data*, vol. 26, no. 2, pp.55–264, 1997.
4. A. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", Proc. 21st Very Large Data Bases Conf., 1995.
5. M. Zaki, "Scalable algorithms for association mining",IEEE Transactions on Knowledge and Data Engineering, vol. 12, no.3, (2000), pp.372-390.
6. Satyavathi N., Rama B., Nagaraju A.,"Research Travelogue-Performance Evaluation of Index support Item Set Mining Algorithms",International Journal of Pure and Applied Mathematics,Volume 120 No. 6 2018, 3641-3651,ISSN: 1314-3395 (on-line version),url: http://www.acadpubl.eu/hub/Special Issue.
7. Satyavathi N., Rama B., Nagaraju A. (2017) Incremental Updating of Mined Association Rules for Reflecting Record Insertions. In: Satapathy S., Prasad V., Rani B., Udgata S., Raju K. (eds) Proceedings of the First International Conference on Computational Intelligence and Informatics. Advances in Intelligent Systems and Computing, vol 507. Springer, Singapore.
8. J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data May 2000.
9. Pei, J., Han, J., Nishio, S., Tang, S., and Yang, D. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proc.2001 Int.Conf.on Data Mining. 2001.
10. Pietracaprina A, Zandolin D (2003) Mining frequent itemsets using Patricia tries. In:Goethals B, Zaki MJ (eds) Proceedings of the FIMI 2003. Available via CEUR-WS.org.
11. C.Borgelt. Keeping Things Simple: Finding Frequent Item Sets by Recursive Elimination. Proc. Workshop Open Software for Data Mining (OSDM'05 at KDD'05, Chicago, IL), 66– 70. ACM Press, New York, NY, USA 2005.
12. Deng, Z. H., & Wang, Z. H. (2010). A new fast vertical method for mining frequent itemsets. International Journal of Computational Intelligence Systems, 3(6), 733–744.
13. Citation Deng Z H, Wang Z H, Jiang J J. A new algorithm for fast mining frequent itemsets using N-lists. Sci China Inf Sci, 2012, 55: 2008–2030, doi: 10.1007/s11432-012-4638-z.
14. Bay Vo, Tuong Le, Frans Coenen, Tzung-Pei Hong, Mining frequent itemsets using the N-list and subsume concepts, International Journal of Machine Learning and Cybernetics, DOI: 10.1007/s13042-014-0252-2.
15. Zhi-Hong Deng, Sheng-Long Lv: Fast mining frequent itemsets using Nodesets. Expert Syst. Appl. 41(10): 4505-4512 (2014).

AUTHORS PROFILE

N.Satyavathi is Assistant professor at vaagdevi college of Engineering, obtained her Bachelors and Masters Degree in Computer Science and Engineering from JNTUH in 2005 and 2010 respectively. She is PhD research scholar in the Department of CSE at JNTUH.

Dr B.Rama has received MSc degree in Computer Science from Kakatiya University, Warangal. She obtained a PhD in Computer Science from Sri Padmavathi Mahila Viswa Vidhyalayam, Tirupati. Since 2010 she has been at the computer science department of Kakatiya University, where she serves as an Assistant professor. She published around 40 articles in various journals and reputed conferences like IEEE & Springer. She has patents published 5 in number for her credit. She authored Scholarly and research articles on a variety of areas related to Data mining, cloud computing, Machine learning and received a best paper award presented in International Conference.

Dr.A.Nagaraju is an Assistant Professor in the Department of Computer Science, Central University of Rajasthan. He received M.Sc (Pure Mathematics) from the Central University of Hyderabad, M.Tech in Computer Science and Technology from University of Mysore, and Ph.D. in Computer Science and Engineering from Osmania University, Hyderabad. His research area includes Wireless Sensor Networks, Ad-hoc Networks, Energy-Efficient Scheduling algorithms in Highly Distributed Systems, Artificial Intelligence and Machine Learning applications in Data Mining and Network Security. He has published 40 international conference papers, and 15 peer-reviewed journals indexed by SCIE, Scopus, and DBLP. He worked as Co-ordinator for the Department of Computer Science from Aug-2016 to Aug-2019, Central University of Rajasthan.