Research Journal of Computer and Information Technology Sciences _____ISSN 2320 – 6527
Vol. **12(2),** 1-5, December **(2024)** *Res. J. Computer and IT Sci.*

Performance Analysis of Mining Frequent Itemsets Based on Tree Structure Algorithm using Synthetic Dataset

Tusharkumar S. Patel[1*] and Harshad B. Bhadka[2]
[1]Faculty of Technology & Engineering, C. U. Shah University, Wadhwan City, Gujarat, India
[2]RDI Centre, C. U. Shah University, Wadhwan City, Gujarat, India
tusharit85@gmail.com

Available online at: www.isca.in
Received 28th July 2023, revised 13th February 2024, accepted 15th September 2024

Abstract

The most important data mining problem is mining of association rules. There are mainly two sub-problems, finding all frequent itemsets which is above threshold and finding association rules from generated frequent itemsets. The efficiency of algorithms is dependent on three factors: the candidates generation process, the structure is used and the implementation. All the previously available algorithms for mining frequent itemsets from Synthetic dataset are not efficient and scalable. The main aim of this paper is to presents a newly discovered Frequent Itemset Tree (FI-Tree) data structure. It is used for stowing frequent itemsets and its associated Transaction ID sets. In several data characteristics, MFIBT have a unique feature is that it has runs speedy. Large-scale experiments had been conducted and performance compared between several algorithms, a result shows that MFIBT better performs in terms of memory consumption and execution time on synthetic dataset. Also it is highly scalable in mining frequent itemsets from synthetic dataset.

Keywords: Association Rules, Frequent Itemset Mining, Data Mining.

Introduction

The most important data mining problem is mining of association rules. In mining of association rules, extract the association relationship between a set of products. To extract rules from set of relationship have two sub-problems. First is finding all frequent itemsets which is above threshold and second is finding association rules from frequent itemsets. To finding all frequent itemsets sub-problem is main role in association rule mining.

Frequent pattern finding is crucial explored field. Finding frequent patterns have mainly three direction (1) frequent itemset mining (2) sequential pattern mining and (3) sub-structure mining. Frequent itemsets are items exists in a dataset with itemset support is greater than a user-specified threshold. Frequent itemsets are used in different knowledge mining problems that find most important itemsets from datasets like associated rules, classifiers, sequences, correlations, clusters etc. Out of which associated rules is crucial research problem.

Frequent itemsets are itemsets that exists in a dataset which have criteria greater than or equal to a user specified minimum value criteria. For example, digital camera and memory card items set that exists frequently together in a transaction dataset is a frequent itemset. Frequent itemset mining finds relationship between items in a given synthetic dataset. It is used in different applications such as store layout, cross-marketing, customer shopping habits and various decision-making problems.

We newly discovered a Frequent Itemset Tree (FI-Tree) data structure. It is used for stowing frequent itemsets and its associated Transaction ID sets. In several data characteristics, MFIBT have a unique feature is that it has runs speedy. It has strong performance in different kinds of dataset, better than the pre-existing available methods in various parameters and its scalability is high for finding from synthetic dataset.

In this paper, next section presents frequent itemset mining related work. After that, list out the steps of MFIBT algorithm for frequent itemset finding. Then analyze performance of algorithms for frequent itemset finding. In the end, conclusion is denoted.

Related work

In 1993, Agrawal et al. was first proposed method for frequent itemset mining from transactional dataset in the shape of mining associated rules. It examine customers shopping styles by mining association rules between various set of frequent items. For example, if customers are purchasing computer, out of which how many percentages of customers are interested to buy printer and type of printer on the one time to the store in market? This kind of knowledge can used to improve trades by proper decision making, cross marketing and store layout[1].

There are thousands of research articles exists on frequent itemsets finding methods with several types of improvements and implementations. Scalability of algorithms to handles variety of different datasets and different mining problems in a diversity of applications. The Apriori algorithm is a level wise searching method and requires number of database scans on horizontal layout based synthetic dataset. Eclat algorithm[2] works on vertical layout based synthetic dataset, which is better in an execution time but requires large space of main memory.

FP-Growth algorithm[3] works on projected layout based synthetic dataset, which is better than all discussed above algorithms because no candidate generates but links store in main memory. Also studies on the Split and Merge (SaM)[4] algorithm for frequent itemset generation.

An itemset contains set of items. A k-itemset is a itemset have k items and it is commonly denoted by L_k. The total number of transactions that exists the itemset is called support count of the itemset. Apriori and FP-Tree (AFPT) algorithm is a mixer of Apriori and FP-Growth methods. Its working mechanism is faster as compared to Apriori and FP-Growth methods[5].

The frequent itemset mining issue is total number of itemset generated. If user specified minimum support threshold is lower then generation of itemsets are very huge. Then pruning uninteresting itemsets in process is among the main problem in frequent itemset finding.

Methodology

We provide a concise procedure of the Mining Frequent Itemset Based on Tree structure (MFIBT) for generating frequent itemsets discussed below:

A newly discovered data structure is Frequent Itemset Tree (FI-Tree). It is used for stowing frequent itemsets and its associated Transaction ID sets.

Now, frequent itemset mining MFIBT algorithm steps are given below: i. Suppose no more memory required, frequent 1-itemsets, transaction dataset and also main memory is exists for producing 2-itemsets candidate based on frequent 1-itemset. Scan dataset one time and produce frequent 1-itemsets with the parallel generate transaction sets, which exists the Itemset. ii. Produce 2-itemsets candidate based on frequent 1-itemset only. iii. Resulted 2-itemsets candidate node count is less than user specified minimum threshold with the help of FI-Tree data structure, it will be eliminated. Now at the second level, FI-Tree contains only frequent 2-itemsets. Iv. Similarly, approve the itemset by scan the frequent itemsets and its associated Transaction ID sets for each frequent 3, 4… n–itemset.

Performance Analysis

To analyze performance of algorithms, experiments were conducted on Intel® corei3™ CPU, 2.13 GHz and 3 GB of RAM computer with Microsoft Windows 7 Home Basic Version 2009 Service Pack 1 operating system. All algorithms were coded using JAVA language. Two synthetic dataset T10I4D100K and T40I10D100K provided by the FIMI repository[6,7]. The N was set to 1,000. Where, N is the number of distinct items in datasets. Parameters of the synthetic datasets are shown in Table-1. In the dataset T10I4D100K, |T|=10, |I|=4 and |D|=100K. In the dataset T40I10D100K, |T|=25, |I|=10 and |D|=100K. Table-2 shown datasets and their properties.

Table-1: Parameters of the synthetic datasets.

|T|	Average number of items per transaction
|I|	Average length of a frequent itemset
|D|	Number of transactions

Table-2: Datasets and their Properties.

Datasets	Type	No. items	Average length	No. transactions	Size (KB)
T10I4D100K	Sparse	1000	10.1	100000	4026
T40I10D100K	Sparse	1000	39.6	100000	15213

Figure-1 display our test results for T10I4D100K dataset using the execution time vs. different minimum support. The same results denoted by Table-3. Also peak memory in Mbytes requirements shown in Figure-3 and Table-4. We can see that the MFIBT is around 37.8 times faster than Apriori, 1.6 times faster than SaM algorithm and 1.5 times faster than AFPT algorithm with minimal support at 0.25%. Figure-2 display our test results for T10I4D100K dataset using the execution time vs. passes (minsup=0.25%).

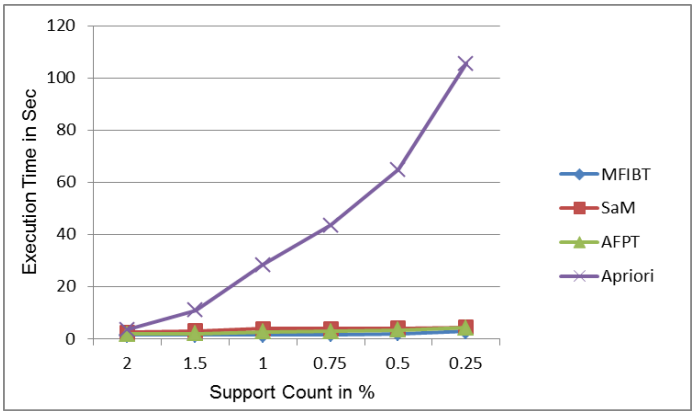

Figure-1: Total execution time of T10I4D100K dataset.

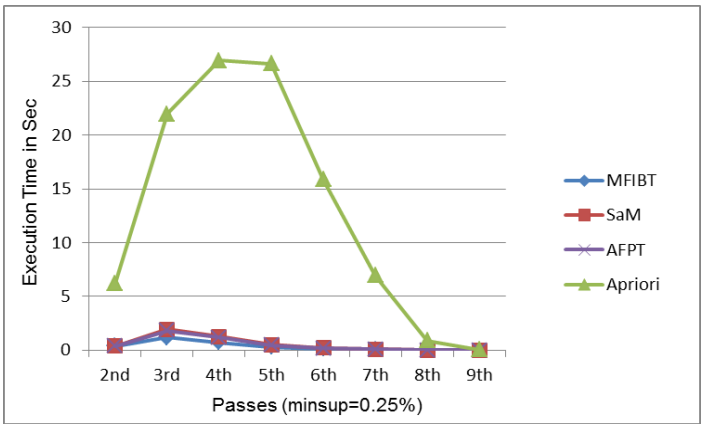

Figure-2: Execution time of T10I4D100K dataset.

Table-3: Total execution time of T10I4D100K dataset.

Support (in %)	Total Execution time in second			
	MFIBT	SaM	AFPT	Apriori
2	1.404	2.497	1.856	3.419
1.5	1.417	2.934	1.909	10.92
1	1.441	3.791	2.59	28.32
0.75	1.515	3.824	2.808	43.49
0.5	1.872	3.96	3.26	64.867
0.25	2.793	4.427	4.138	105.517

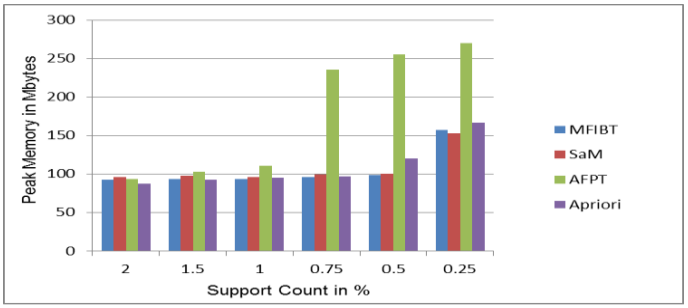

Figure-3: Peak memory required for T10I4D100K dataset.

Table-4: Peak memory required for T10I4D100K dataset.

Support (in %)	Peak Memory in Mbytes			
	MFIBT	SaM	AFPT	Apriori
2	92.36	96.07	93.46	87.75
1.5	93.28	97.4	102.69	92.25
1	93.56	95.73	110.43	94.7
0.75	96.35	99.38	235.2	96.75
0.5	98.29	100.55	255.18	119.79
0.25	157.18	152.95	270.24	166.38

Figure-4 display our test results for T40I10D100K dataset using the execution time vs. different minimum support. The same results denoted by Table-5. Also peak memory in Mbytes requirements shown in Figure-6 and Table-6. We can see that the MFIBT is around 2.1 times faster than Apriori, 1.4 times faster than SaM algorithm and 1.1 times faster than AFPT algorithm with minimal support at 0.25%. Figure-5 display our test results for T40I10D100K dataset using the execution time vs. passes (minsup=0.25%).

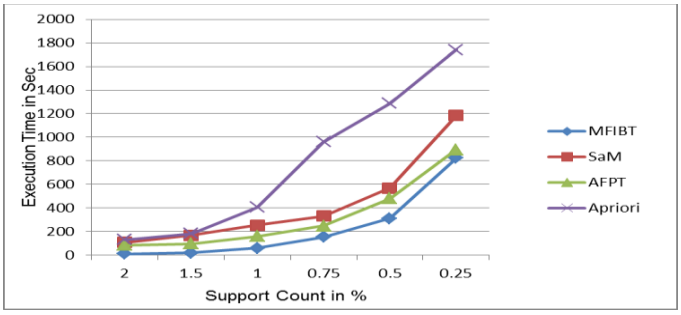

Figure-4: Total execution time of T40I10D100K dataset.

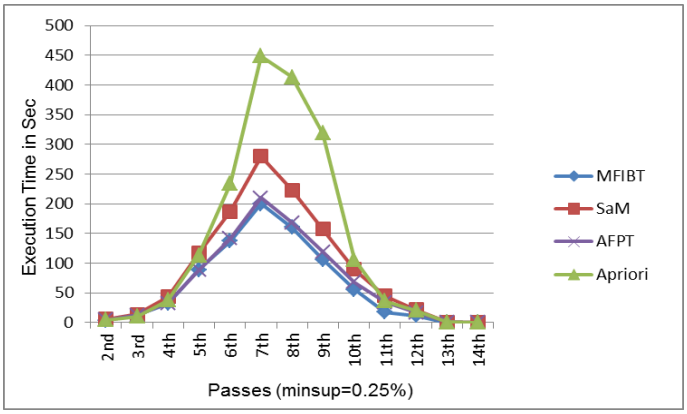

Figure-5: Execution time of T40I10D100K dataset.

Table-5: Total execution time of T40I10D100K dataset.

Support (in %)	Total Execution time in second			
	MFIBT	SaM	AFPT	Apriori
2	10.249	108.35	84.894	131.208
1.5	19.94	169.536	98.112	181.412
1	59.796	254.994	159.594	405.163
0.75	153.926	330.96	248.658	962.26
0.5	311.414	569.67	479.502	1286.03
0.25	825.102	1183.44	894.852	1743.13

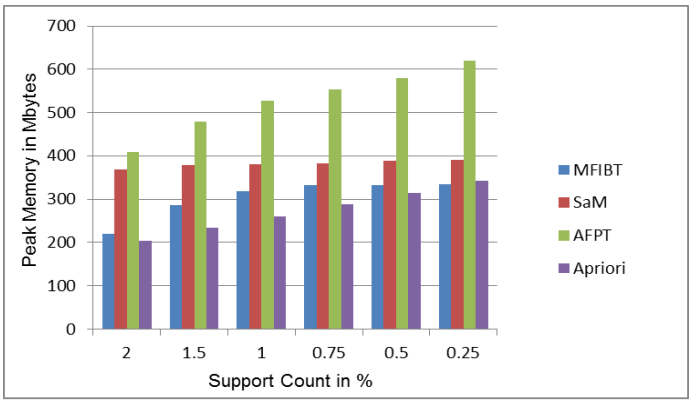

Figure-6: Peak memory required for T40I10D100K dataset.

Table-6: Peak memory required for T40I10D100K dataset.

Support (in %)	Peak Memory in Mbytes			
	MFIBT	SaM	AFPT	Apriori
2	219.57	368.5	408.96	203.61
1.5	286.11	378.12	478.4	233.5
1	318.07	380.75	526.46	260.6
0.75	331.57	382.84	552.72	287.7
0.5	332.88	388.02	579.81	314.8
0.25	334.84	391.51	619.97	341.9

Figure-7 display our scalability test results for T10I4 dataset using the relative time vs. number of transactions (minsup=0.75%). The same results denoted by Table-7.

Also peak memory in Mbytes requirements shown in Figure-8 and Table-8. We can see that the MFIBT is around 27.5 times faster than Apriori, 3.7 times faster than SaM algorithm and 2.2 times faster than AFPT algorithm with number of transactions at 250K.

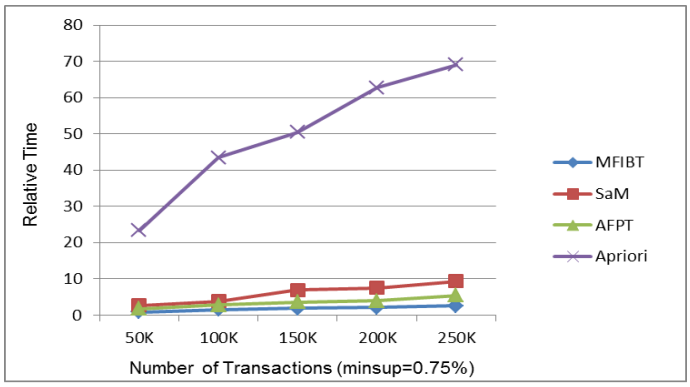

Figure-7: Relative time of T10I4 dataset.

Table-7: Relative time of T10I4 dataset.

Number of transactions (minsup=0.75%)	Relative Time			
	MFIBT	SaM	AFPT	Apriori
50K	0.795	2.715	1.732	23.32
100K	1.515	3.824	2.808	43.49
150K	1.85	6.875	3.554	50.45
200K	2.044	7.526	3.9	62.743
250K	2.508	9.377	5.446	69.035

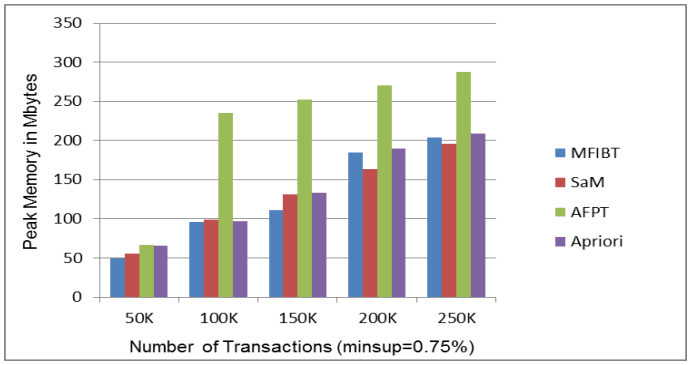

Figure-8: Peak memory required for T10I4 dataset.

Table-8: Peak memory required for T10I4 dataset.

Number of transactions (minsup=0.75%)	Peak Memory in Mbytes			
	MFIBT	SaM	AFPT	Apriori
50K	49.5	55.53	67.08	66.09
100K	96.35	99.38	235.2	96.75
150K	110.675	131.445	252.61	133.125
200K	185	163.51	270.02	189.5
250K	204.325	195.575	287.43	208.875

Figure-9 display our scalability test results for T40I10 dataset using the relative time vs. number of transactions (minsup=0.75%). The same results denoted by Table-9. Also peak memory in Mbytes requirements shown in Figure-10 and Table-10. We can see that the MFIBT is around 5.1 times faster than Apriori, 1.5 times faster than AFPT algorithm and 1.2 times faster than SaM algorithm with number of transactions at 250K.

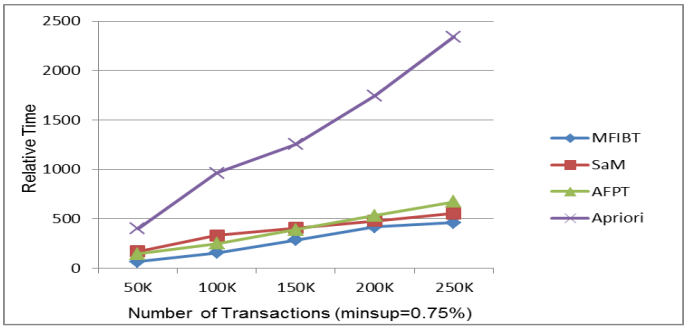

Figure-9: Relative time of T40I10 dataset.

Table-9: Relative time of T40I10 dataset.

Number of transactions (minsup=0.75%)	Relative Time			
	MFIBT	SaM	AFPT	Apriori
50K	64.681	168.57	142.968	401.086
100K	153.926	330.96	248.658	962.26
150K	285.867	405.39	390.018	1254.69
200K	417.807	479.82	531.378	1747.11
250K	459.747	554.25	672.738	2339.53

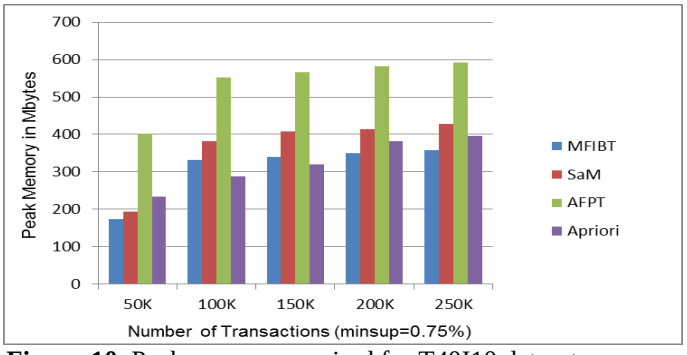

Figure-10: Peak memory required for T40I10 dataset.

Table-10: Peak memory required for T40I10 dataset.

Number of transactions (minsup=0.75%)	Peak Memory in Mbytes			
	MFIBT	SaM	AFPT	Apriori
50K	174.51	194	402.86	233.22
100K	331.565	382.84	552.72	287.7
150K	340.453	407.93	565.97	320.285
200K	349.34	413.02	583.22	382.87
250K	358.227	428.11	591.47	395.455

Conclusion

In this paper, we conduct the experiments on MFIBT algorithm to find frequent itemsets using a newly discovered Frequent Itemset Tree (FI-Tree) data structure. It is used for stowing frequent itemsets and its associated Transaction ID sets. In several data characteristics, MFIBT have a unique feature is that it has runs speedy. The experiments includes time required for execution, maximum memory requirement and scalability are evaluated for MFIBT, SaM, AFPT and Apriori algorithms using synthetic dataset and varying minimum support thresholds. Switching the user specified support value does not major affect the MFIBT algorithm in execution time and memory consumption.

Our experiments prove that the MFIBT algorithm has a magnitude order improved over the basic Apriori method on synthetic dataset and is better than the other three algorithms. Since MFIBT algorithm uses frequent itemset tree structure, execution time and memory consumption do not necessary increase as the number of transactions increases. Different experiments had been conducted and performance compared between several algorithms, a result shows that MFIBT better performs in terms of memory consumption and execution time on synthetic dataset which are T10I4D100K and T40I10D100K. Scalability of MFIBT algorithm is high in frequent itemset mining.

References

1. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB, 1215, 487-499.

2. Borgelt, C. (2003). Efficient implementations of apriori and eclat. In FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations (Vol. 90).

3. Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2), 1-12.

4. Borgelt, C. (2010). Simple algorithms for frequent item set mining. In Advances in Machine Learning II: Dedicated to the Memory of Professor Ryszard S. Michalski, 351-369. Berlin, Heidelberg: Springer Berlin Heidelberg.

5. Lan, Q., Zhang, D., & Wu, B. (2009). A new algorithm for frequent.

6. Goethals, B. (2003). Frequent itemset mining dataset repository. http://fimi. cs. helsinki. fi/data/.

7. Bayardo, R. (2014). Frequent itemset mining dataset repository. UCI datasets and PUMSB.