IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.6, Special Issue.11, Dec 2018 E-ISSN: 2347-2693

A Performance Analysis of Improved_Eclat Algorithm in Association Rule Mining

V. Priya[1*], S.Murugan[2]

[1,2]Dept. of Computer Science, Nehru Memorial College, Puthanampatti, India

Available online at: www.ijcseonline.org

Abstract— In mining frequent Itemsets, Eclat algorithm is an important one. But it has some inefficiency. We proposed an algorithm called Improved_Eclat which is a new improved eclat method with high efficiency in the searching process to reduce the running time using two dimensional pattern tree. By comparing Improved_Eclat with Eclat , Eclat-opt and Bi-Eclat, hereby it is proved that the Improved_Eclat has the highest efficiency in mining associating rules from various databases.

Keywords— Association rules, Eclat, increased search approach, increased two- dimensional pattern trees.

I. INTRODUCTION

Association rule mining is one of the most important data mining problems. Apriori algorithm [1], and FP-growth algorithm [2] are the most standard algorithms for mining association rules. Apriori algorithm is based on the approach of breadth-first search, which generates a (K+1)-itemsets form frequent k-itemsets, until no more frequent itemsets can be found. Apriori needs to scan the database many times, its efficiency is limited by the number of candidate Itemsets. Many people proposed their improved Apriori algorithms [3-6], such as DHP [7], DIC [8], MFI-TransSW [9], and so on. Different from Apriori, FP-growth algorithm is based on the approach of depth-first search, it does not need to generate candidate Itemsets instead, it compresses datasets into a FP-tree and obtains frequent patterns using an FP-tree-based pattern growth mining method. On the base of FP-growth, many improved algorithms was proposed[10-12], such as TD-FP-Growth [13], H-Mine[14],IFP-growth [15], and so on.

These algorithms are all based on the horizontal data format. There are many other algorithms based on the vertical data format, such as Eclat and so on. The efficiency can be improved using vertical data format when compared with horizontal data format. We will propose the vertical data format algorithm.

TID	Itemset
1	1,2,3
2	1,3,4
3	2,4,5

Horizantal Format

Items	TID Set
1	1,2
2	1,3
3	1,2
4	2,3

Vertical Format

Fig 1. Data representation formats.

In this paper can be organized as follows. In Section II, Review on the several improved Eclat Algorithms. In Section III, We give the proposed Algorithm, Improved_Eclat and its process, In Section IV, We Present the Experimental Results, and Conclusions are given in Section V.

II. RELATED WORK

ZAKI et al[18] proposed Eclat_Diffsets algorithm which adopts Boolean matrix and Diffsets to reduce the memory usage and increase the efficiency. The algorithm adopts Boolean matrix to store the itemset and TID_set, and uses binary operation to determine the intersection, which can improve the efficiency of intersection. It can effectually reduce the size of memory required to store intermediate results when the database is solid.

Xiong et al [19] proposed HEclat algorithm, the hash Boolean matrix to polish up the calculation of intersection. The algorithm uses hash Boolean matrix to store the TID_set of itemsets. It does not need to compare the TIDs of two itemsets one by one, but need to use bitwise "AND" operation on the two Boolean matrix. The technology of hash Boolean matrix can recover the competence only when the number of transactions of a database is not large. If the number of transactions is very large, it makes things worse.

Feng et al [20] proposed Eclat-opt algorithm, It uses double layer hash table, panel list of the set of itemset and TID misplaced threshold. These technologies can trim the candidate 3-itemset, decrease the search space as well as the time of making candidate itemsets, and haste up the design of intersection. In general, Eclat-opt algorithm is muchmore effective than other Eclat based algorithms.

traditional Eclat algorithm, the results of tests show that the Bi-Eclat algorithm increases improved performance.

The surviving enriched Eclat algorithms can shine up the effectiveness of original Eclat algorithm, but these algorithms still have some problems, such as large number of candidate itemset, incompetence of Itemsets connection and intersection.

III. METHODOLOGY

The proposed Improved_Eclat Algorithm, which is based on vertical data format. An Improved_Eclat using two-dimensional pattern tree and the TID_sets of itemsets in the vertical data format table are added into the pattern tree row by row. New frequent itemsets are generated by joining the new added itemset with the existing frequent itemsets in the pattern tree. The process of joining is based on the BFS. Firstly, the candidate and frequent 2-itemsets are generated, then make the candidate and frequent 3-itemsets, and etc, until all frequent itemsets in the pattern are linked with the new added itemset. The candidate itemsets are poised directly by the frequent itemset and new added itemset, without the operation of itemsets connection. Due to the breadth-first search approach, it can be used for cut the candidate itemsets completely. All redundant candidate itemsets will be curt.

Improved_Eclat Algorithm:

The core process of Improved_Eclat algorithm Initially, it scans the database and stores it into a table using vertical data format.Next, it establishes a null increased two-dimensional pattern tree and adds the TID_sets of itemsets in the vertical data format into the pattern tree row by row to produce new frequent itemsets.At last, all frequent itemsets can be created by picking up all nodes in the pattern tree.

Input:
- o D,Database of Transaction
- o MS, the minimum support threshold

Output:
- FIs ,All frequent

Itemsets Steps:

1. VM = CreateVMfromDB(D)
2. PT=CreateNullPatternTree
3. for i = 1 to length(VM) do
4. if length(VM[i].TID_sets) >= MS then
5. AddItemtoPatternTree(VM[i], PT, MS);
6. end
7. FIs=GetAllFrequentItemsetsfromPatternTree(Two-dimensionalPattern Tree: PT)

Fig 2. Improved_Eclat Algorithm.

The increased two-dimensional pattern tree is poised of nodes and a cover pointer array.

A node is clear as following:

TreeNode={itemset, TID_set, PtoF, PtoC, HP, IsValid}.

Every TreeNode includes the information of a frequent itemset, such as the itemset and the TID_set. Besides itemset and TID_set,

A TreeNode also has other four elements:

The element **"PtoF"** are pointers pointing to the nodes of its two fathers.The element **"PtoC"** are pointers pointing to the nodes of its children.The element **"HP"** are a slanting pointer, which is used for connecting the frequent itemsets with same length together.

The element **"IsValid"** is a boolean value, which is used for indicating whether the node can be combined with another node.

If an (*K+1*)-itemset *A* is joined by two *k*-itemset *B* and *C*, then *B* and *C* are the fathers of *A*, and *A* is the child of *B* and *C*. The pattern tree has multiple layers and the frequent item sets with same length belong to the same layer.

The layer pointer array is defined as following: Layer Pointers: array of Pointer. The elements of layer pointer array are pointers pointing to the first node of all layers. So we can find all frequent *1*-itemsets based on the first element of the layer pointer array, and find all frequent *2*-itemsets based on the second element, and so on. All frequent item sets can be found from the elements of the layer pointer array.

Initially, establishing sorted two-dimensional pattern tree is tree initialization, add a null node to tree as the ancestor node of all frequent 1-itemsets, and set the length of LayerPointers as zero. After initialization, every TID_set of frequent 1-itemset in the vertical data format table are added into the pattern tree as the nodes of the first layer. Supposing the new added itemset as *In*, which contains the transactions of TID_set (*In*), the process of adding a new itemset to pattern tree can be described as follows:

Firstly, form a new node "Node(*In*)" for *In*. The elements of Node(*In*) are:

Node(*In*).itemset=*In*,
Node(*In*).TID_set=TID_set(*In*),
Node(*In*).PtoF=nil,
Node(*In*).PtoC=nil,
Node(*In*).IsValid=true.

The HP of the last node in the first layer (frequent 1-itemset layer) is set to point to Node(*In*).Combine Node(*In*) with every node (Node(*Ix-1*)) in the first layer (frequent 1-itemset layer) to generate candidate 2-itemsets and calculate the support degree of the candidate 2-itemsets. If a candidate 2-itemset is frequent, a new node for the candidate 2-itemset

will be made, the element "itemset" of new node will be set as the mixture of *Ix-1* and *In*, and the element "TID_set" of new node will be set as the intersection of TID_set(*Ix-1*) and TID_set(*In*). Also, the element "PtoF" of new node will point to Node(I_{x-1}) and Node(*In*). Finally the HP of the last node in the second layer (frequent 2-itemset layer) is set to point to the new node. If a candidate 2-itemset is not frequent, it will need to be abandoned and the element "IsValid" of all children of Node(*Ix-1*) must be set as false.

Join Node(*In*) with every node (Node(*Ix-2*)) in the second layer (frequent 2-itemset layer) to generate candidate 3-itemsets. If the element "IsValid" of Node(*Ix-2*) is false, it cannot be joined with Node(*In*). Thus, we need to set the value of "IsValid" as true for further combination. If the element "IsValid" of Node(*Ix-2*) is true, it needs to be combined with Node(*In*) to generate a candidate 3-itemset, and to calculate the support degree of the candidate 3-itemset. If the candidate 3-itemset is frequent, a new node for the candidate 3- itemset will be made, the element "itemset" of new node will be set as the combination of *Ix-2* and *In*, the element "TID_set" of new node will be set as the intersection of TID_set(*Ix- 2*) and TID_set(*In*). Also, the element "PtoF" of new node will point to Node(*Ix-2*) and Node(*In*), finally set the HP of the last node in the thrid layer (frequent 3-itemset layer) to point to the new node. If a candidate 3-itemset is not frequent, it needs to be abandoned and the element "IsValid" of all children of Node(*Ix-2*) must be set as false.In the same way, Node(I_n) needs to be combined with all of the nodes in other layers to get all frequent itemsets.The process of adding a TID_set of frequent 1-itemset in the vertical data format table to the pattern tree .

Input:

- VM,Vertical Matrix
- PT,Two Dimentional Pattern Tree
- MS,Min-Sup

Output:

- A new Pattern Tree

Steps:

1. for i = 1 to length(LayerPointers) do
2. if i =1 then
3. newNode = AddNewNodetoTree(VM[i])
4. tmpNode = LayerPointers[i]
5. while tmpNode ≠ null do
6. if tmpNode.isValid = true then
7. NewCombineNode = CombineNewPattern(newNode, tmpNode)
8. if Support(newCombineNode) >= MS then
9. AddNodetoTree(newCombineNode)
10. else
11. SetAllChildrenFalse(newCombineNode)
12. end
13. else
14. tmpNode.isValid == true;
15. end if Support(newCombineNode) >= MS
16. tmpNode = tmpNode.HorizontalPointer
17. end while tmpNode ≠ null
18. end LayerPointers

Fig 3: AddItemtoPatternTree.

Taking the vertical format database, by setting the minimum support degree then process of building the increased two-dimensional pattern . Improved_Eclat can clip the candidate itemsets completely and reduce the number of candidate itemsets obviously under the prior knowledge. The algorithm adopts Boolean matrix to store the itemset and TID_set, and uses binary operation to determine the intersection, which can improve the efficiency of intersection and then calculating the intersection and support degree, it can clearly decrease the computational complexity.

IV. RESULTS AND DISCUSSION

In this paper, we compare Improved_Eclat with the Eclat focused algorithms. To show the usefulness of the Improved_Eclat algorithm, we associate it with an Eclat algorithm [16], and an Eclat-opt algorithm [20] and Bi-Eclat [21]. All testing were executed on a PC with Inter Core 1.8G and 4G main memory, running on Microsoft Windows 10 64-bit and all the programs are coded in VC++. We compare the Accuracy and processing time of Improved_ Eclat on the following Datasets.

The standard datasets Mushroom, Chess and Accidents, T10I4D100K, Kosarak, T40I10D100K, are used. The entire records from the above datasets are preserved as 100% .The whole records in the datasets aggregation and the metrics are

measured after processing and the results are for every 20% of data.

Accuracy:
Acuracy is the most important in data mining. In this scenario, Eclat-Opt secures the least average accuracy mark of 82.49% whereas the proposed Improved_Eclat algorithm locks the highest accuracy mark average of 87.63%. The very close result in accuracy of 87.15% is reached by the Bi-Eclat method. The measured values are presented in the graph is given in Figure 4.

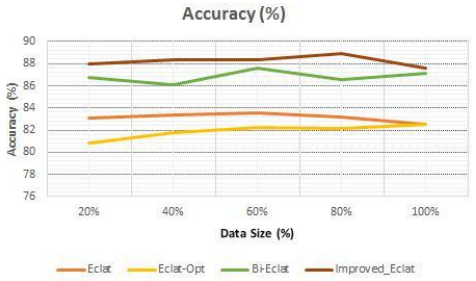

Fig 4: Graph for Accuracy (%).

Processing Time:
Processing time is one of the vigorous fight in data mining procedures. The execution time of remaining and proposed algorithms are stately here in millisecond units for higher care. The full transaction records are fragmented into 20% data slices and the processing time is measured after every data slice is processed. Therefore, 5 processing times are noted for every data mining procedure discussed here. The practical processing times are show in the graph is given in Figure 5.

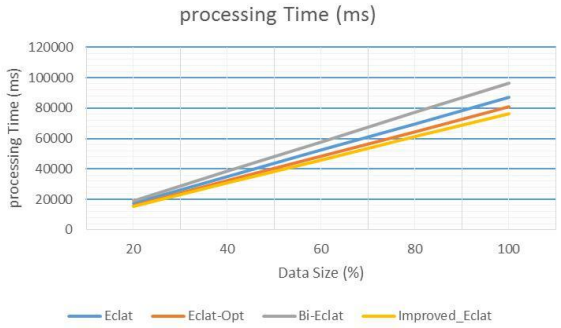

Fig 5. Graph for Processing Time (mS).

V. CONCLUSION

An Improved_Eclat algorithm, on the basis of increased search approach, adopts new technologies, such as two-dimensional pattern tree. In this method of construction of two-dimensional pattern tree, the prior knowledge can be used for extract the candidate itemsets fully and all redundant candidate itemsets will be abrupt and calculating the intersection and support degree, it can visibly reduction the computational complexity. The analysis and experimental shows that Improved_Eclat has the maximum performance in mining associating rules from various databases than Eclat, Eclat-opt and Bi-Eclat. In the future research, we will also concentrate on cloud offloading using Improved_eclat, and explore new techniques to improve the accuracy and running time using cloud environment.

REFERENCES

[1] R. Agrawal, T. Imilienski and A. Swami, *"Mining association rules between sets of items in large databases*", Proceeding of the ACM SIGMOD Int"l Conference on Management of Data, Washington DC, **(1993)**, pp. 207-216.
[2] J. Han, J. Pei and Y. Yin, *"Mining frequent patterns without candidate generation"*, Proceedings of the2000 ACM SIGMOD International Conference on Management of Data, **(2000)**, pp. 1-12.
[3] J. Dong and M. Han, *"BitTableFI: An efficient mining frequent itemsets algorithm"*, Knowledge-Based Systems, vol. 20, no. 4, **(2007)**, pp. 329-335.
[4] C. Aflori and M. Craus, *"Grid implementation of the Apriori algorithm*", Advances in Engineering Software, vol.38, no. 5, **(2007)**, pp. 295-300.
[5] F. Berzal, J. Cubero and N. Marin, *"TBAR: An efficient method for association rule mining in relational databases*", Data & Knowledge Engineering, vol. 37, no. 1, **(2001)**, pp. 47-64.
[6] D. C. Pi, X. L. Qin and N. S. Wang, *"Mining Association Rule Based on Dynamic Pruning*", Mini- Micro Systems, vol. 10, **(2004)**, pp. 1850-1852.
[7] J. Pork, M. Chen and P. Yu, *"An effective hash based algorithm for mining association rules*", ACM SIGMOD, vol. 24, no. 2, **(1995)**, pp. 175-186.
[8] S. Brin, R. Motwani and C. Silverstein, *"Beyond market baskets: generalizing association rules to correlations,"* ACM SIGMOD Conference on Management of Data, Tuscon, AZ, (**1997**), pp. 265–276.
[9] H. F. Li and S. Y. Lee, *"Mining frequent itemsets over data streams using efficient window sliding techniques*", Expert Systems with Applications, vol. 36, no. 2, **(2009)**, pp. 1466-1477.
[10] Y. F. Zhong and H. B. Lv, *"An Incremental Updating Algorithm to Mine Association Rules Based on Frequent Pattern Growth*", Computer engineering and Application, vol. 26, **(2004)**, pp. 174-175.
[11] R. Balazs, *"Nonordfp: An FP-Growth Variation without Rebuilding the FP-Tree*", Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementa-tions, **(2004)**.
[12] G. Gosta and J. F. Zhu, *"Fast Algorithms for Frequent Itemset Mining Using FP-Trees*", IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 10, **(2005)**, pp. 1347-1362.
[13] W. Ke, T. Liu, H. J. Wei and L. J. Qiang, *"Top down fp-growth for association rule mining"*, The 6th Pacific-Asia Conference, PAKDD 2002, Taipei, Taiwan, **(2002)**, pp. 334-340.
[14] J. Pei, J. W. Han and H. J. Lu, *"H-Mine: Fast and space-preserving frequent pattern mining in large databases*", IIE Transactions, vol. 39, no. 6, **(2007)**, pp. 593-605.
[15] K. C. Lin, I. Liao and Z. S. Chen, *"An improved frequent pattern growth method for mining association rules*", Expert Systems with Applications, vol. 38, no. 5, **(2011)**, pp. 5154-5161.

[16] M. Zaki, *"Scalable algorithms for association mining",* IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 3, **(2000)**, pp. 372-390.
[17] L. S. Thieme*, "Algorithmic Features of Eclat",* Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November, **(2004)**.
[18] M. J. Zaki and K. Gouda, "*Fast vertical mining using diffsets*", Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, **(2003)**, pp. 326-335.
[19] X. Z. Yang, C. P. En and Z. Y. Fang, "*Improvement of Eclat algorithm for association rules based on hash Boolean matrix"*, Application Research of Computers, vol. 27, no. 4, **(2010)**, pp. 1323-1325.
[20] F. P. En, L. Yu, Q. Q. Ying and L. L. Xing, "*Strategies of efficiency improvement for Eclat algorithm*",Journal of Zhejiang University (Engineering Science), vol. 47, no. 2, **(2013)**, pp. 223-230.
[21] Xiaomei Yu, Hong Wang, *"Improvement of Eclat Algorithm Based on Support in Frequent Itemset Mining",Journal ofComputers*", vol. 9, no. 9, **(2014)**, pp. 2116-2123

