An improvement of FP-Growth association rule mining algorithm based on adjacency table

Ming Yin[1], *Wenjie* Wang[1,*], *Yang* Liu[1], and *Dan* Jiang[2]

[1]School of Software and Microelectronics Northwestern Polytechnical University, Xi'an 710072, P.R China
[2]Chinese helicopter design institute, Tianjin, P.R China

Abstract. FP-Growth algorithm is an association rule mining algorithm based on frequent pattern tree (FP-Tree), which doesn't need to generate a large number of candidate sets. However, constructing FP-Tree requires two scansof the original transaction database and the recursive mining of FP-Tree to generate frequent itemsets. In addition, the algorithm can't work effectively when the dataset is dense. To solve the problems of large memory usage and low time-effectiveness of data mining in this algorithm, this paper proposes an improved algorithm based on adjacency table using a hash table to store adjacency table, which considerably saves the finding time. The experimental results show that the improved algorithm has good performance especially for mining frequent itemsets in dense data sets.

1 Introduction

Data mining is a process of obtaining potentially useful knowledge from data[1]. As an important part of data mining, association rule mining reflects the intrinsic relationship between complex itemsets [2]. Agrawal et al[3, 4] proposed the Boolean association rule proplem and the corresponding Apriori algorithm. Considering the disadvantages of the Apriori algorithm, J. Han et al. proposed the FP-Growth algorithm using the FP-Tree to generate frequent itemsets [5]. It compresses the transaction itemsets into FP-Tree to store the association information of itemsets with FP-Tree and generates frequent itemsets using the FP-Tree[6]. Although the algorithm requires two database scans, and doesn't need to generate candidate sets [7], it needs to create FP-Tree that contains all the itemsets, which requires lots of memory. If frequent itemsets in the database is too many and the memory can't load the mapping information of all the items in the FP-Tree, the algorithm won't be effective [8]. Besides, scanning the transaction database twice also makes the performance of the algorithm low.

This paper proposes an improved FP-Growth algorithm based on adjacency table which draws on the idea of graphs. After scanning the itemsets in the transaction database, we adopts a storage method combing the adjacency table with the hash table, which can remove itemsets that are less than the minimum support as soon as possible and avoid generating all nonempty subsets of the long largest frequent itemsets. The algorithm makes full use of the established adjacency table, and only needs to scan the original transaction database once. It has the advantages of fast running speed, small memory consumption and low complexity.

* Corresponding author: wenjie@mail.nwpu.edu.cn

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

The rest of this paper is organized as follows: In Section 2, related works are discussed.The section 3, we proposes the improvement of the FP-Growth algorithm based on adjacent table and the mining process of frequent itemsets. Section 4, we analyse the time performance between FP-Growth algorithm and the improved one. In Section 5, we do experiments to compare the performance of FP-Growth algorithm with the improved one on various itemsets. In last section, we present our conclusions and future work.

2 Related works

Association rule mining is an important data analysis method and data mining technology [9]. Although Agrawal et al. proposed the Apriori algorithm, the algorithm uses iterative process for the data subset and uses the candidate itemsets produced earlier to generate frequent itemsets later, which results in low efficiency of the algorithm and being difficult to be used in the mining of massive data [10,11]. In response to the disadvantages of the Apriori algorithm, J. Han proposed the FP-Growth algorithm to generate frequent itemsets[5]. It compresses the transaction itemsets into FP-Tree to store the association information of itemsets with FP-Tree and generates frequent itemsets using the FP-Tree[12].

The algorithm introduces an data structures including three parts. The first part is the header-table. It is used to record the frequency of occurrences of all itemsets and then sort descendly by the frequency recorded; the second is FP-Tree, which maps the original itemsets to FP-Tree in memory and maintains the association information between itemsets; the third is the list of nodes. The frequent itemsets in all header tables are the node lists' heads which respectively point to the position of frequent itemsets in the FP-Tree[13].Although the algorithm requires two database scans, and does not need to generate candidate itemsets[14], it needs to create FP-Tree that contains all the itemsets. If frequent itemsets in the database is too many and the memory can't load the mapping information of all the itemsets in the FP-Tree, the algorithm won't be effective[15]. Besides, scanning the database twice also makes the algorithm inefficient.The DMFIA algorithm is an improvement based on the FP-Growth algorithm, which reduces the frequency of database scans, but still adopts the FP-Tree storage structure and the traversal method,whice has to search many layers and generate a lot of candidate itemsets at each layer leading to the low efficiency of the algorithm[16].

This paper proposes an improved FP-Growth algorithm. After scanning the itemsets in the data, we adopts a storage method combing adjacency table with hash table, which can remove itemsets quickly that are less than the minimum support .The algorithm makes full use of the established adjacency table, and only needs to scan the original database once. It has the advantages of fast running speed, small memory consumption and low complexity.

3 Improvement of FP-Growth algorithm based on adjacent table

The FP-Growth algorithm scans the database shown in table1 twice, figure .1 shows that how the transaction database converted into the FP-Tree. However, for large-scale data sets, the algorithm has shortcomings of memory and computational, making the algorithm inefficient [17].

Table 1. transaction database.

TID	Items
T100	I_2,I_3,I_5
T200	I_6, I_2
T300	I_3,I_1,I_4

T400	I_4,I_2,I_3,I_1,I_5
T500	I_3,I_5,I_4
T600	I_5,I_6

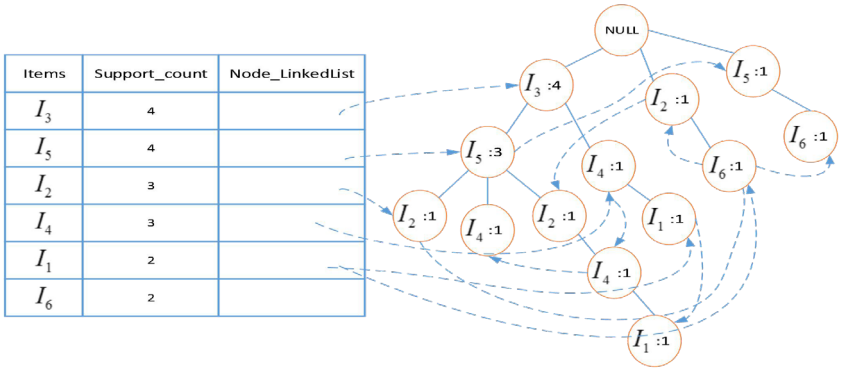

Fig.1. The transaction database is converted into the FP-Tree.

3.1 Generation of adjacency table

Takingthe database intable 1 as an example, theitems ineach itemsets can be considered related to each other and form a complete graph. Once the same two items are associated, the weight of the edge is incremented by one. The weight of the final edge is the association frequency. After the first scan of the database, the formed association relationship graph is shown in figure 2.

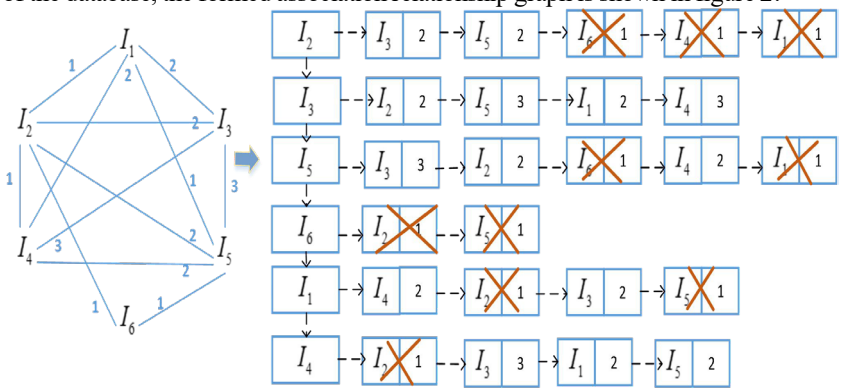

Fig. 2. The formed association relationship diagram and the generated adjacency table.

3.2 The mining of frequent itemsets

Using minimum support to remove unrelated items and adopting the adjacency table's vertices, adjacency points, for example, we removed the itemsets whose support countings are one and then can get a single-dimensional frequent itemsets as follow: {(I_1,I_3:2);(I_1,I_4:2);(I_2,I_5:2);(I_4,I_3:3);(I_4,I_5:2);(I_5,I_3:3)}.

Some single-dimension frequent itemsets are subsets of three frequent itemsets. By the subset continue to mine adjacency table, we can obtain three frequent itemsets: {(I_1, I_3,I_4:2),(I_4,I_3,I_5:2),(I_2,I_3,I_5:2)}Similarly, three frequent itemsets are subsets of the four frequent itemsets, and so on. The algorithm not end until all the frequent itemsets are mined.The improved FP-Growth algorithm steps are as follows:

Scan the transaction database to generate the adjacency table:

(1) HashMap<String, HashMap<String,Integer>> GraphMap;//Define an adjacency table
(2) HashMap<String,Integer> frequent;
(3) while (read each transaction set I of transaction database! = null) {
(4) HashMap<String,Integer> list;//declare list to store adjacent points and weights
(5) Take the items in the array String arr[] in turn as the vertices of the adjacency table;
(6) if(GraphMap.containsKey(top));
(7) If there is, then go to the set of adjacency points of the vertex that is the association relationship set, list = GraphMap.get(top);
(8) else if it does not exist, it initializes the vertex's set of adjacency points;
(9) list= new HashMap<String,Integer>();}
(10) The loop traverses the items in arr[] in turn to store non-top items into the "list";
(11) list.put(arr[j],(list.containsKey(arr[j]) ? 1 + list.get(arr[j]) : 1));
(12) GraphMap.put(top, list);}
Mine adjacency tables to generate frequent item collections:
(1) The loop obtains the vertices of the GraphMap in turn. Then it gets adjacent frequent itemsets according to the vertices, and removes items less than min_sup;
(2) The obtainedvertices, form a single-dimensional frequent itemset with adjacent points;
(3) frequent.put(item, weight_count);//if not in the frequent itemsets
(4)//Frequent itemsets (A, B) and (B, A) are considered to be the same frequent itemsets;
(5)Traverse the adjacent points in GraphMap and declare ArrayList<Integer> Weight
(6)//Weight array is used to store the weight of adjacent points that is the association frequency;
(7) If the set of adjacency points of the vertex contains items in "frequent", we take this vertex $I_i \cup item_j$, and take the minimum min in the array Weight_sort[] as the frequent number;
(8) if(frequent $\not\subset (I_i \cup item_j)$) then frequent.put($I_i \cup item_j$, min);
(9) Repeat steps (6)-(9) until the return value of step (8) is null;
(10) return frequent;//Final itemsets mining is completed;

4 Time complexity analysis

This paper compares the time performance of the FP-Growth algorithm with an improved algorithm based on adjacency table. The following are the symbols used for performance analysis. n: the number of transactions in the entire database; n_1: The number of itemsets in the FP-Tree corresponding to the original database that is the number of leaf nodes in the FP-Tree; n_2: The average number of items in per itemset I; tr_i: Time to read transaction i from the original database; tl_i: Time of FP-Growth counting frequency of each item in the database; gl_i: Time of improved FP-Growth counting frequency of each item in the database; ts_i : Time consumption of sorting each itemset by head table order; tin_{si}: Time to insert each item into FP-Tree;gin_i: Time to insert each item into the adjacency table; tf_i: Time to get frequent itemsets from FP-Tree; gf_i : Time to get frequent itemsets from the adjacency table; T_{FP} : Time to eventually find all frequent itemsets from the original database using FP-Growth algorithm; Tg: Time to find all frequent itemsets from the original database using improved FP-Growth algorithm.

$$T_{FP} = \sum_{i=1}^{n} (tr_i + tl_i + ts_i + tin_{si} + tf_i) \quad (1)$$

When FP-Tree is close to the binary tree, time complexity of FP-Growth is lowest. Formula (1) above is approximately equal to Formula (2).

$$T_{FP} = \sum_{i=1}^{n} (tr_i + tl_i + ts_i) + \sum_{i=1}^{n} \log_2(i) + n1\log_2(n_1)) \quad (2)$$

$$T_g = \sum_{i=1}^{n} (tr_i + gin_i + gf_i + gl_i) \quad (3)$$

Since this paper adopts hash tables to design adjacency tables, the most time improved FP-Growth algorithm cost is approximately equal to Formula (4) below.

$$T_g = \sum_{i=1}^{n} (tr_i + gr_i) + o(n) + n_2 o(n_2) \quad (4)$$

When constructing an FP-Tree, it is necessary to sort each itemset by the order of the head table.Besides, counting the frequency of each item needs to traverse the header table.

On the contrast, when constructing the adjacency table, each itemset doesn't have to be sorted but traverse the hash table simply, so Formula (5) and Formula (6) can be as following:

$$\sum_{i=1}^{n} (tr_i + tl_i + ts_i) > \sum_{i=1}^{n} (tr_i + gr_i) \quad (5)$$

$$\sum_{i=1}^{n} \log_2(i) = \log_2 n! \quad (6)$$

Formula (7) can be deduced by Stirling's approximation:

$$\ln n! > \left(n + \frac{1}{2}\right) \ln n - n \quad (7)$$

Since the average number of items n2 of each transaction set I is less than the number of leaf nodes n1 in the FP-Tree, the following can be obtained by Formula (4), (5), (6), and (7):$o(n) + n_2 o(n_2) < \sum_{i=1}^{n} log_2(i) + n_1 log_2(n_1)$), So $T_{FP} > T_g$.

5 Experimental results

To study the performance of the algorithm, this paper compares the FP-Growth algorithm with the improved one on sparse dataset and dense dataset under the same experimental environment. The sparse dataset averages 10 items per transaction set. In figure 3, the minimum support counting of each transaction database is 100; in figure 4, the number of transaction set is 500,000. The dense dataset averages 23 items per transaction set. In figure 5, the minimum support counting of each transaction database is 500; in figure 6, the number of transaction set is 30,000.

From the experimental results, it can be concluded that the effiency of mining the frequency itemsets using the FP-Growth algorithm improved obviously after improving the FP-Growth algorithm. Especially when dealing with maasive frequent itemsets, the effect is more prominent. The main reason lies in that the improved algorithm only needs to scan the transaction database once, and doesn't have to sort many itemsets by the support frequency. Further more,the fast lookup of the hash table also helps save more time, even in the case of massive frequent itemsets.

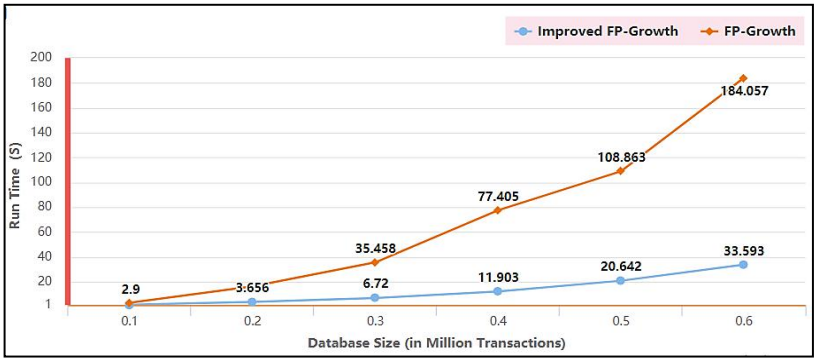

Fig.3. Comparison of effects on different numbers of sparse transaction items.

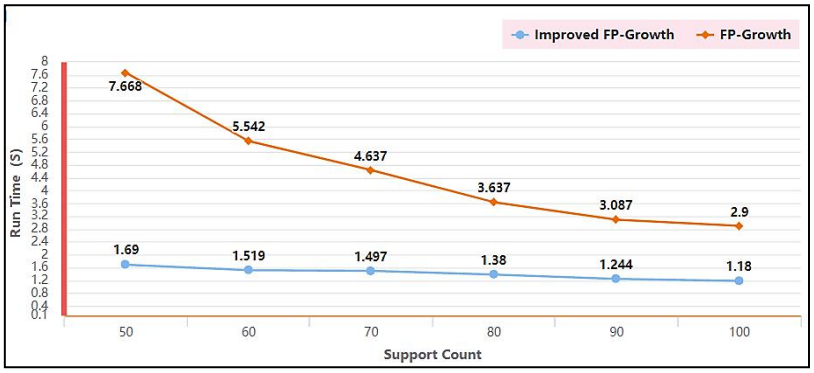

Fig.4. Comparison of different support counting for sparse transaction items.

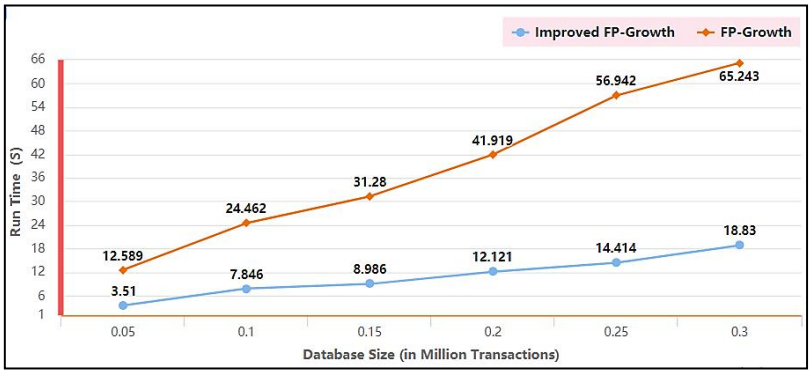

Fig.5. Comparison of effects on different numbers of dense transaction items.

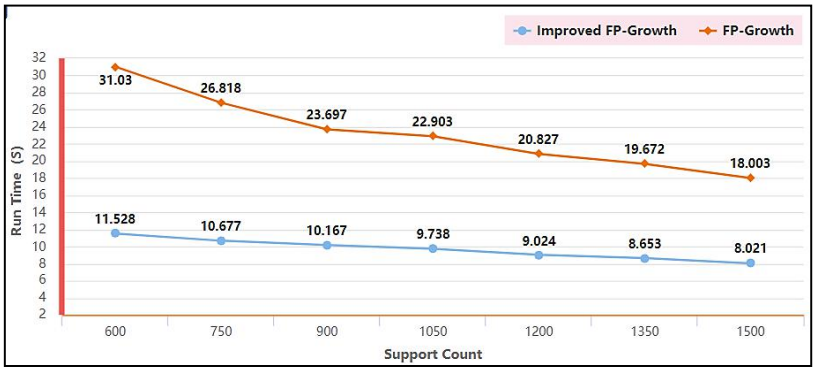

Fig.6. Comparison of different support counting for dense transaction items.

6 Conclusions and future scope

After studying the mining process of association rules of FP-Growth algorithm, this paper proposes an improved FP-Growth algorithm based on adjacency table.It significantly improves the performance of the algorithm to a certain degree. First of all, the improved algorithm only scans the transaction database once, which reduces the I/O operations greatly. Secondly, It doesn't require the establishment of a header table and a large number of sort operations. Finally, when mining frequent itemsets, the improved algorithm adopts the hash table for the fast lookup and does not need recursive mining. These have

considerably reduce the algorithm's time and memory consumption. Especially in dealing with dense transaction items, the improved algorithm shows high performance and is supposed to have great application value. The future work will perfect the FP-Growth algorithm combing the application, and study the improvement in parallelization.

References

1. Wu X, Kumar V, Quinlan J R, et al. Top 10 algorithms in data mining[J]. Knowledge & Information Systems, 2007, 14(1):1-37.
2. Sharma S, Bhatia S. Analysis of Association rule in Data Mining[C]. International Conference on Information and Communication Technology for Competitive Strategies. ACM, 2016:1-4.
3. Agrawal R. Fast Algorithm for Mining Association Rules in Large Databases[C]// Proc. Very Large Data Bases Conference. 1994.
4. Agrawal R, Imieliński T, Swami A. Mining Assocation Rules between Sets of Items in Large Databases[J]. Proc of Sigmod, 1993, 22(2):207-216.
5. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]. ACM SIGMOD International Conference on Management of Data. ACM, 2000:1-12.
6. Heaton J. Comparing dataset characteristics that favor the Apriori, Eclat or FP-Growth frequent itemset mining algorithms[C]. Southeastcon. IEEE, 2017:1-7.
7. Difallah D E, Benton R G, Raghavan V, et al. FAARM: Frequent Association Action Rules Mining Using FP-Tree[C]. IEEE, International Conference on Data Mining Workshops. IEEE Computer Society, 2011:398-404.
8. Hao J, He M. A Parallel FP-Growth Algorithm Based on GPU[C]. IEEE, International Conference on E-Business Engineering. IEEE Computer Society, 2017:97-102.
9. Chang H Y, Lin J C, Cheng M L, et al. A Novel Incremental Data Mining Algorithm Based on FP-growth for Big Data[C]. International Conference on NETWORKING and Network Applications. IEEE, 2016:375-378.
10. Tsai C F, Lin Y C, Chen C P. A new fast algorithms for mining association rules in large databases[C]. International Conference on Systems, Man and Cybernetics. IEEE, 2002:6 pp.
11. Sun D, Teng S, Zhang W, et al. An Algorithm to Improve the Effectiveness of Apriori[C].International Conference on Cognitive Informatics. IEEE, 2007:385-390.
12. Heaton J. Comparing dataset characteristics that favor the Apriori, Eclat or FP-Growth frequent itemset mining algorithms[C]. Southeastcon. IEEE, 2017:1-7.
13. Chen M, Gao X D, Li H F. An efficient parallel FP-Growth algorithm[C]. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. IEEE, 2009:283-286.
14. Difallah D E, Benton R G, Raghavan V, et al. FAARM: Frequent Association Action Rules Mining Using FP-Tree[C]. International Conference on Data Mining Workshops.IEEE, 2011:398-404.
15. Hao J, He M. A Parallel FP-Growth Algorithm Based on GPU[C]. IEEE, International Conference on E-Business Engineering. IEEE Computer Society, 2017:97-102.
16. Subbulakshmi B, Dharini B, Deisy C. Recent weighted maximal frequent itemsets mining[C]. International Conference on I-Smac. IEEE, 2017:391-397.
17. Willhalm T, et al. FPTree: A Hybrid SCM-DRAM Persistent and Concurrent B-Tree for Storage Class Memory[C]. International Conference on Management of Data. ACM, 2016:371-386.