Research of Optimized Algorithm about Mining Frequent Closed Itemsets

Ling Yuan, Ihab Alsammak, Wasan Itwee
School of Computer Science
Huazhong University of Science and Technology, Wuhan, China
{cherryyuanling, I201421219, I201521052}@hust.edu.cn

Abstract— ***Frequent closed patterns technique is used to extract smaller and most important frequent patterns. In existed algorithms, when finding frequent closed itemsets, a large amount of useless frequent itemsets will generated, which leads to high cost as well as ineffective. To solve this problem, we propose an optimized algorithm about mining frequent closed patterns without duplication. Several experiments have been implemented to compare the proposed algorithm with the existed algorithm. The results demonstrate that our proposed approach has better performance by reducing the number of frequent closed items without losing any information.***

Keywords- Data mining; Association rule mining; Frequent pattern; Frequent closed items; CLOSET algorithm

I. INTRODUCTION

The frequent itemsets play an important role in the data mining, which can find important patterns in databases, such as sequences, clusters, association rules, correlations, classier, and episodes. Usually, the number of extracted frequent itemsets are very big, but actually the number of most important itemsets is much smaller than those extracted itemsets. Therefore, it is very important to constrain the number of extracted frequent itemsets. Then we propose a more efficient mining algorithm to reduce the number of extracted patterns which is less important and interesting. So far, the researchers have proposed many frequent itemsets mining algorithms [1, 2, 3]. Some closed itemsets mining algorithms are designed for centralized datasets and their performance degrades for large datasets, that is, datasets with a large number of records [4]. There is no need to find associations and complete set of frequent itemsets, we just need to get the frequent closed itemsets and their corresponding rules.

In general, mining frequent closed itemsets in terms of capacity and power is the same with mining complete set of frequent itemsets. However, mining frequent closed itemsets can make the size fewer than frequent itemsets, and also increases the mining efficiency. Let us examine some examples, suppose we have a database containing only two transactions, *"{(K1, K2,. . . , K100), (K1, K2…….K50)}"*, the *Min_Sup = 1* and the *Min_Conf = 50%*. To find the association mining with mining frequent itemsets methods, it will generate $2^{100} - 1 \approx 10^{30}$ frequent itemsets, which are *(K1), . . . , (K100), (K1, K2), . . . , (K99, K100), . . . ,(K1, K2……. K100)*. And it will generate a very large number of association rules. While when we use the method of mining frequent closed itemsets, it will generate only two frequent closed itemsets:*{(K1, K2….. K50), (K1, K2……. K100)}*, and one association rule: *"(K1, K2….. K50) => (K51, K52……. K100)"*. Therefore, we can conclude that mining frequent closed itemsets is more efficient than mining frequent itemsets. How to mining frequent closed itemsets more efficiently is our research focus.

II. RELATED WORK

The reference [5] has proposed a method of mining frequent closed itemsets firstly, called A-close algorithm. A-close Algorithm mainly uses breadth-first search method to find frequent closed patterns. But the breadth-first search method is difficult to implement in very large database or datasets with long patterns, especially there are several candidates in the database, because which leads to check the database many times.

The CHARM algorithm [6] and MAFIA algorithm [7] are based on the database vertical representation. The MAFIA algorithm is at most considered for finding maximal patterns, however it has a preference to find closed itemsets. The CLOSET algorithm [8], based on the principle of FP-growth algorithm [9], builds the frequent pattern tree and then build the conditional of FP-trees recursively in a bottom-up tree-search manner.

The CLOSET algorithm is designed to find frequent closed itemsets, and produce association rules, which can reduce both the cost of computing and knowledge in the analysis of the association rule, by reducing results for just frequent closed itemsets. This algorithm mainly uses breadth-first search method to find frequent closed patterns with FP-tree structure. The CLOSET algorithm is a very effective technique but complicated. This algorithm scans the frequent items firstly, then divides frequent items to finding only closed frequent itemsets. The CLOSET algorithm recursively implements mining the subsets of the frequent closed itemsets. The algorithm then effectively creates conditional databases of the frequent closed items separately from the initial transaction database. The CLOSET algorithm can save the cost of time and memory. But when facing a large database, it should need more memory to store the extracted results. In the third section, we will analyze the CLOSET algorithm in detail. The CLOSET algorithm can save the cost of time and memory. But when facing a large database, it should need more memory to store the extracted results.

III. Optimized Algorithm of Mining Frequent Close Itemsets

With analyzing the CLOSET algorithm, when extracting the frequent itemsets, we found that this algorithm still generates a lot of useless frequent patterns which does not need to be replicated. When extracting a large amount of frequent patterns, it requires more space and more time, which leads to high cost as well as ineffectiveness. In this section, we will propose an optimized algorithm to reduce the number of frequent close patterns without loss any information.

A. Main Process of Proposed Algorithm

We use a simple example to explain how our proposed algorithm to implement mining frequent closed itemsets. A fruit basket items is shown in Fig.1. The transaction database of this figure is shown in Table 1, where *min_sup = 2*, and *min_sup* means minimum support.

Figure 1. Market Basket Items in Example

Table 1. Transaction Database

TID	Items in transaction
10	Apple , Coconut , Dates , Eggfruit ,Figs
20	Apple , Banana , Eggfruit
30	Coconut , Eggfruit ,Figs
40	Apple , Coconut ,Dates , Figs
50	Coconut , Eggfruit ,Figs

In the following, we will illustrate the main idea of our proposed algorithm with this example.

1) Count all the frequent items

This step begins by scanning all the database to find the frequent items and build frequent item list, named as *f_list*. The *f_list* include all the support items in the database and then store these items in descending order based on the number of the support, shown in Table 2. *f_list* = {Coconut: 4, Eggfruit:4, Figs: 4, Apple: 3, Dates:2}, and the number after ":" refers to the support of the item. We can omit any infrequent item such as Banana, because the support of this item=1 < *min_up*. For example, [Apple Banana Eggfruit] is listed as [Eggfruit Apple], where Banana is omitted.

Table 2. Sorted in Descending

TDB
Coconut ,Eggfruit, Figs, Apple ,Dates
Eggfruit, Apple
Coconut , Eggfruit ,Figs
Coconut , Figs, Apple, Dates
Coconut , Eggfruit ,Figs

2) Divide search space

Based on the *f_list*, we can divide all frequent closed itemsets into 5 non-overlap subsets. The first one involves Dates item, the second one involves Apple item without Dates, the third one involves Figs item without Apple and Dates, the fourth one contains Eggfruit without Figs, Apple and Dates, and the last one contains only Coconut. So far, we have found all the subsets of frequent closed itemsets.

3) Find all the subsets of Frequent Closed Items

By constructing corresponding conditional databases recursively, we can find all the subsets of Frequent Closed Items (FCI).

a. To find the FCI containing Dates. We need only transactions with Dates. We denote Dates-conditional database as TDB|DATES, which is {[Coconut, Eggfruit, Figs, Apple]; [Coconut, Figs, Apple] }. We should omit Dates item in each transaction in the Dates-conditional database. The support of Dates item = 2. The Items of Coconut, Figs, and Apple appear two time respectively in TDB|DATES. That means, all the transactions with Dates also includes Coconut, Figs, and Apple. But, Eggfruit is infrequent because it appears only one time in TDB|DATES. So, {Coconut Figs Apple Dates: 2} is an FCI. And the itemset covers each frequent item in TDB|DATES.

b. To find the FCI containing APPLE without Dates. Similar with Dates-conditional database, the APPLE-conditional database, TDB|APPLE = {Coconut Eggfruit Figs, Eggfruit, Coconut Figs}. The item with Dates in all transactions are omitted, because all FCI with Dates have been found in TDB|DATES. The support of Apple = 3, since we don't have any item appearing in all transactions in the Apple-conditional and the summation support of Apple = 2 in the first FCI, so Apple: 3 is FCI.

To find all the FCI including Apple without DATES, we should be further project the Apple conditional database. The *f_list* APPLE = <Coconut: 2, Eggfruit: 2, Figs: 2> should ignore infrequent item if it is in *f_list*. Based on the *f_list* APPLE, the FCI of Apple wit out Dates can be divided into three parts:

The first one involves Apple and Figs without Dates, the second one contains Apple and Eggfruit without Dates or Figs, and the third one involves Apple and Coconut without Dates, Eggfruit or Figs. We can build a conditional databases recursively. The support of Figs and Apple equals to (Coconut Figs Apple Dates) which have already been found. It means that all transactions containing (Figs Apple) must also contain (Coconut Figs Apple Dates). Therefore, we don't have any frequent closed itemset involving Figs and Apple without Dates. Similarly, we cannot find FCI containing Coconut and Apple without Dates, Eggfruit or Figs, since {Coconut Apple} is a subset of {Coconut Figs Apple Dates} and sup{Coconut Apple} = sup{Coconut Figs Apple Dates}.The Eggfruit APPLE-conditional database, TDB|EGGFRUIT APPLE = { Coconut }, can't generate any frequent items. Therefore, {Eggfruit Apple: 2} should be a frequent closed itemset.

But the FCI in transaction contains {Apple: 3} have support equal three, and this frequent close item is reduplicate, because the items {Apple} is already found in another FCI{Coconut, Figs, Apple, Dates: 2} and second FCI {Eggfruit , Apple}:2 where the support is smaller. The support of Apple in first FCI transaction = 2 and the support of Apple in second FCI transaction = 2. Meaningful, the Sup{Apple} <= Sup {Coconut, Figs, Apple, Dates} + {Eggfruit , Apple}. Then the total support of this item = 4, so this frequent close item is reduplicate. Therefore, we can prune this frequent close item. This analysis process is shown in Fig. 2.

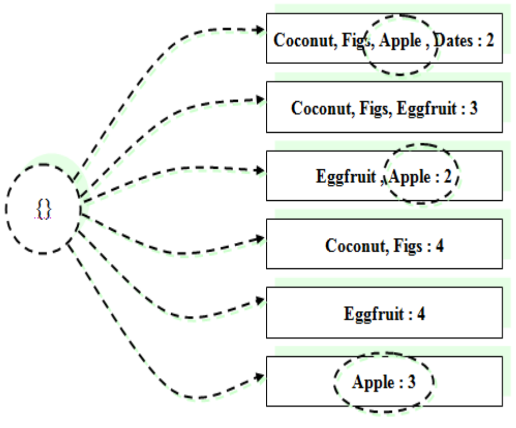

Figure 2. Duplicated Frequent Close Item "Apple"

c. To find FCI containing Figs without APPLE and Dates. The Figs -conditional database, TDB_{Figs} = {Coconut Eggfruit: 3, Coconut}, where Coconut Eggfruit: 3 means that Coconut and Eggfruit appear three times. Because Coconut Eggfruit Figs: 3 is not a subset of any found itemset, it is an FCI.

Also, Coconut appear in all transactions in the Figs-conditional database, and Coconut and Figs have not a subset of any FCI with the equal support. Therefore, {Coconut, Figs: 4} is FCI. But this FCI contains {Coconut, Figs : 4} is reduplicate, since the items with Coconut and Figs is already found in the first FCI transaction and third transaction with the support more than its support.

The support of Coconut and Figs in first FCI transaction = 2. The support of Coconut and Figs in second transaction = 3. Meaningful, the Sup{Coconut, Figs } <= Sup {Coconut, Figs, Apple, Dates} + { Coconut, Eggfruit ,Figs }. Then the total support of these items = 5, so this frequent close itemset is reduplicate, which should be Pruned. This analysis process is shown in Fig. 3.

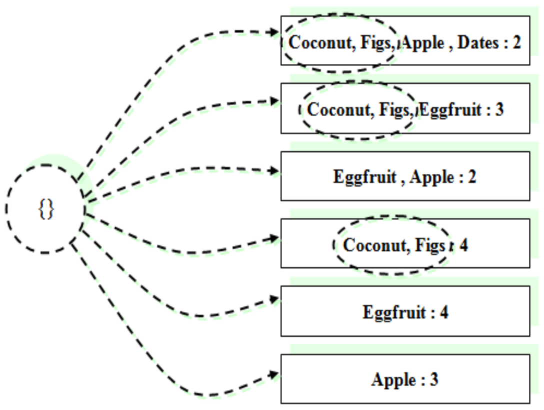

Figure 3. Duplicated Frequent Close Item " Coconut and Figs"

d. To find FCI containing Eggfruit without Figs, Apple and Dates. Similarly, the Eggfruit-conditional database, $TDB_{Eggfruit}$= {Coconut: 3}. But Coconut Eggfruit is not an FCI because it is an right subset of Coconut Eggfruit Figs and sup(Coconut, Eggfruit)=sup(Coconut, Eggfruit, Figs).

Therefore, {Eggfruit: 4} is a FCI. But this FCI in transaction contains {Eggfruit: 4}, which have support equal four. This frequent close itemset is reduplicate because the items Eggfruit is already found in another relation with support more than 4. The support of {Eggfruit} in second transaction = 3 and the support of Eggfruit in third transaction = 2. Meaningful, the Sup{ Eggfruit } <= Sup{ Coconut, Eggfruit, Figs } + { Eggfruit ,Apple }. Then the total support of this item = 5, so this reduplicate frequent close itemset should be pruned. This analysis process is shown in Fig. 4.

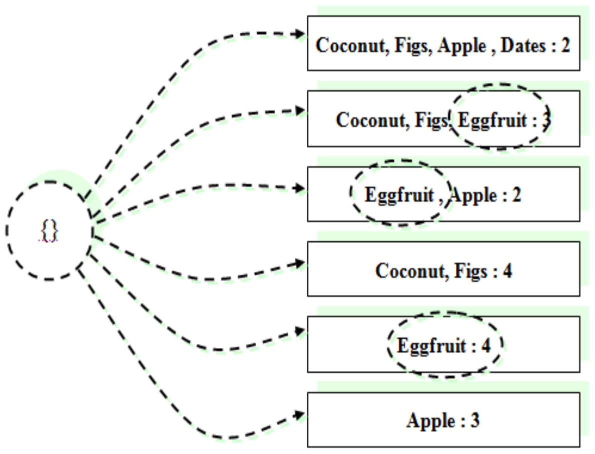

Figure 4. Duplicated Frequent Close Item "Eggfruit"

e. To find FCI involving Coconut. We previously know that there is no FCI involving Coconut without Figs. Therefore, there is no FCI with only Coconut.

4) *Final Frequent Closed Itemsets*

Finally, the extracted set of FCI is:

{[APPLE, COCONUT, DATES, FIGS: 2], [APPLE, EGGFRUIT: 2], [COCONUT, EGGFRUIT, FIGS]: 3}.

B. Optimized Algorithm

In the above section, we use an example to explain the main process our proposed algorithm. In the following, the pseudo code is presented, and this algorithm is named as CLOSET*:

CLOSET* Algorithm:

Input: TDB and min_Sup;

Output: FCI (Optimized) ;

```
1)  For each Transaction in database
2)      Count the frequent items
3)      If(frequent_item >= min_sup)
4)          {Add to f_list;}
5)      Else
6)          Delete (frequent_item)
7)  For each frequent set
8)      Identify the subset and add to list; // based on f_list
9)          Sort subset in descending order:
10) For each transaction in list
11)     If (transaction(x) contains subset(frequent set)
12)         { Prune selected item;}
13)     Else
14)         {Add to tree sub transaction as frequent closed Pattern }
15) For each transaction in frequent closed items
        If (FCI[X] Subset FCI[Y,Z] ) & (Sup[X]<=
        $\Sigma$Sup[Y,Z])
16)     {Prune selected transaction ;}
17)     Else
        {Add to new sub transaction as final}
```

IV. EXPERIMENTS AND ANALYSIS

In this section, we present a performance comparison between our proposed algorithm and CLOSET algorithm, and illustrate the experimental results. The experiments are performed in environment of Core(TM)i-2130 CPU @ 3.40 GHz and 4.00 GB memory size. And Java language on windows 7 platform is used to implement the proposed algorithm. The experiments dataset is shown in Table 3.

Table 3. Experiments Datasets

No.	Databases	Transaction Number	Items Number
I.	Fruits Data	6.0	6.0
II.	Market Basket	11040.0	12.0
III.	List of Medications	13454.0	20.0
IV.	List of Retail goods	100000.0	400.0

In our experiments, we have a comparison of CLOSET algorithm with our algorithm, in order to highlight the benefits of our proposed CLOSET* algorithm, the experiments focus on evaluating the size of generated FCI compared between the CLOSET* algorithm and CLOSET algorithm. We have used four datasets in our experiments. The first dataset about fruits contains 5 transactions over 6 items, the average length of transactions is 4 and most of the transactions contain about 3 to 4 items. The second dataset about retail market basket data from an anonymous Chinese retail store. This market basket datasets contains 11040.0 transactions over 12 items. The average length of transactions is 13, and most of the transactions contain about 7 to 11 items. The third dataset about medication contains the retail List of medications from a pharmacy store. It contains 13454.0 transactions over 20 items. The average length of transactions is 15, and most of the transactions contain between 9 to 15 items. The fourth dataset about retail goods contains 100000.0 transactions over 400 items. The average length of transactions is 200, and most of the transactions contain between 220 to 300 items.

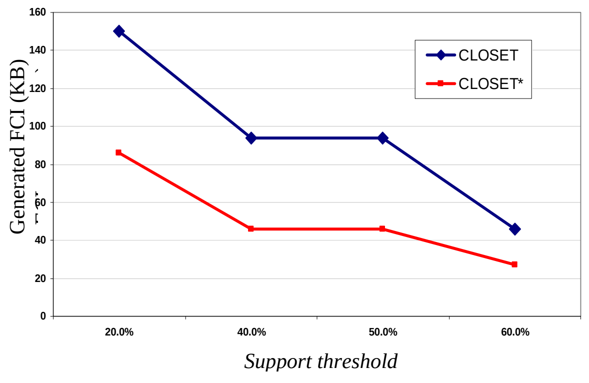

Figure 5. Compared Generated FCI in dataset I

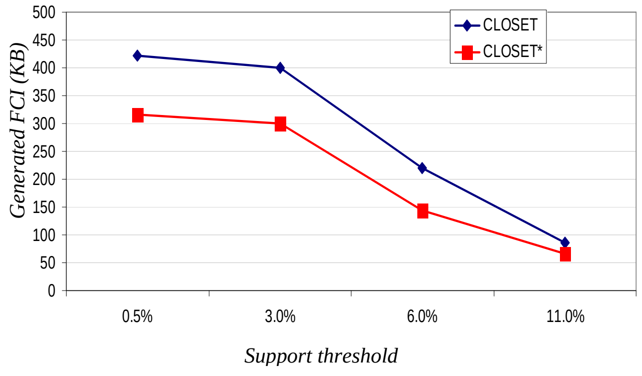

Figure 6. Compared Generated FCI in dataset II

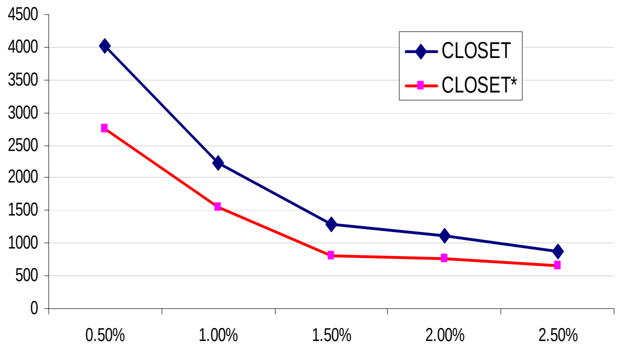

Figure 7. Compared Generated FCI in dataset III

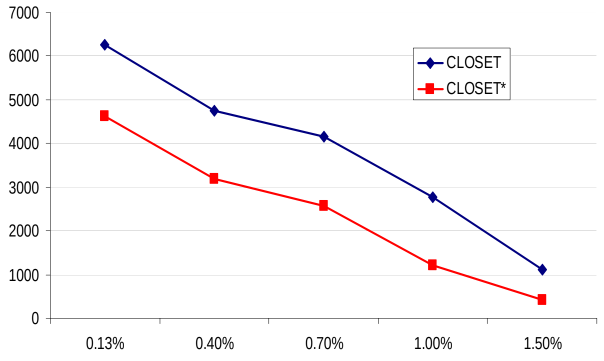

Figure 8. Compared Generated FCI in dataset IV

The experiments result is shown in Fig.5, Fig.6, Fig.7, and Fig. 8. We can figure that the size of generated FCI with our proposed method is less than the size of generated FCI with CLOSET algorithm. For example, considering the database3 with *min_Sup* = 0.5 %, our method generates size of 2750 KB of FCI, but the CLOSET algorithm generates size of 4010 KB of FCI. When *min_Sup*= 1.5 %, our method generates size of 804 KB of FCI, but the CLOSET algorithm generates size of 1290 KB of FCI.

V. CONCLUSION

In this paper, an optimized algorithm is proposed to extract frequent closed itemsets with less duplication without loss any information. This approach can reduce the number of generated frequent closed itemsets and the search space, which makes the data analysis much easier and flexible. With experiments based on several datasets, we demonstrate that the proposed algorithm can generate less FCI (Frequent Close Itemsets) with less duplication.

ACKNOWLEDGMENT

This work was supported by National Natural Science Fund of China under grants 61502185.

REFERENCES

[1] Dungarwa Jayesh and Neeru Yadav , “A Review paper for mining Frequent Closed Itemsets”, ISSN: 2321-7782 , 2014.
[2] Mohammad Arsyad and Sofianita Mutalib , “Review of Frequent Itemsets Mining in High Dimensional Dataset”, IEEE 978-1-4799-7910, 2014.
[3] J.J. Cameron, A. Cuzzocrea, F. Jiang, C.K.-S. Leung, “Mining frequent itemsets from sparse data streams in limited memory environments”, in: Proceedings of the WAIM 2013, Springer, pp. 51–57.
[4] C.K.-S. Leung, C.L. Carmichael, “Exploring social networks: a frequent pattern visualization approach”, in:Proceedings of the SocialCom 2010, IEEE Computer Society, pp. 419–424.
[5] Maryam Shekofteh, “A survey of algorithms in FCIM” , DOI 10.1109/DSDE.2010.
[6] M. Zaki and C. Hsiao, “ CHARM: An efficient algorithm for closed itemset mining”. In SDM'02, April 2002.
[7] D. Burdick, M. Calimlim, and J. Gehrke, “MAFIA A maximal frequent itemset algorithm for transactional databases”, In ICDE'01, April 2001.
[8] J. Pei, J. Han, and R. Mao, “ CLOSET: An efficient algorithm for mining frequent closed itemsets”, In DMKD'00, May 2000.
[9] J. Han, J. Pei, Y. Yin, “Mining frequent patterns without candidate generation”, In SIGMOD'00, May 2000.