ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

A Combinatorial Temporal Closed+ High Utility Itemset Mining Algorithm in Transactional Database

Santhosh.J[1], Sukanya[2]

Assistant Professor, Department of Computer Science, Sree Narayana Guru College, Coimbatore, Tamil Nadu[1]

M.Phil. Scholar, Department of Computer Science, Sree Narayana Guru College, Coimbatore, Tamil Nadu[2]

Abstract: High utility item set mining from a transactional database helps to discover the items with high utility based on profit, cost and quantity. Although several significant algorithms have been proposed in recent years, they experienced the problem of producing a large number of candidate itemsets for high utility itemsets. Such a huge number of candidate item sets degrades and reduces the mining performance in terms of storage space requirement and execution time. The situation may become worse when the database contains lots of datasets, long transactions or long high utility itemsets. The proposal introduces three algorithms which are temporal High utility pattern growth (THUP-Growth), temporal closed frequent pattern growth (TCFP-Growth) and temporal UP-Growth+, for mining closed high utility itemsets with a set of effective strategies for pruning candidate item sets rapidly. The information of high utility itemsets is maintained in a tree-based data structure named closed+ utility pattern tree (TCUP-Tree) such that candidate itemsets can be generated efficiently with only two scans of database, then that will be segmented into multiple clusters for fast computation. The proposed algorithms reduce the number of candidates and database scans effectively. This also outperforms best than the existing algorithms and significantly reduces the runtime and memory and storage overhead, especially when databases contain lots of high and long transactions.

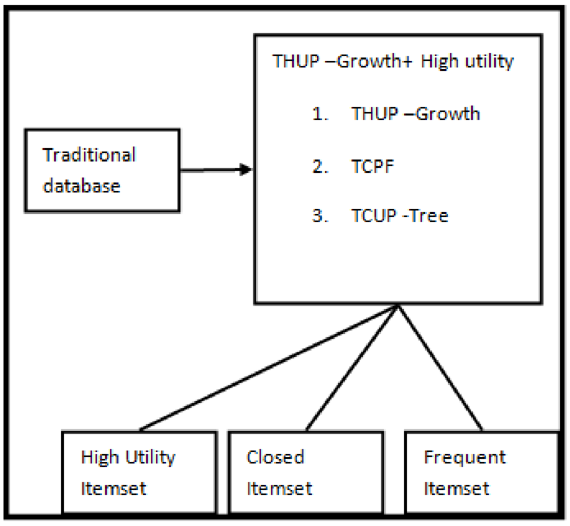
Keywords: Frequent itemset, high utility itemset, closed and frequent itemset, FP growth, utility mining, data mining.

I. INTRODUCTION

Association rule mining is one of the most important techniques of data mining which was introduced in [1]. Data mining aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. These rules are mostly used in various areas such as telecommunication networks, market and risk management, and inventory control and so on. Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is Lk, Lk = {I1, I2,…., Ik}, association rules with this itemsets are generated in the following way: the first rule is {I1, I2, … , Ik-1}⇒ {Ik}, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty. Since the second sub problem is quite straight forward, most of the researches focus on the first sub problem. The first sub-problem can be further divided into two sub-problems: candidate large itemsets generation process and frequent itemsets generation process. We call those itemsets whose support exceed the support threshold as large or frequent itemsets, those itemsets that are expected or have the hope to be large or frequent are called candidate itemsets. In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions. Further, the association rules are sometimes very large. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. The proposed strategies to reduce the number of association rules, such as generating only "interesting" rules, generating only "non redundant" rules, or generating only those rules satisfying certain other criteria such as coverage, leverage, lift or strength. This work reviews the most well known algorithm for producing association rules. The review ends with an outlook on tools which have the potential to deal with long itemsets and considerably reduce the amount of itemsets returned.

II. ARCHITECTURE DIAGRAM

The following architecture diagram explains the steps and mechanisms followed in the proposed system. The input resource is transactional databases which contains market basket analysis dataset. We applied the algorithms and mechanisms on the dataset and the derived outputs are high quality itemset, closed itemset and frequent itemset.



III. PROPOSED SYSTEM

i. Algorithms for Mining Closed + High Utility Itemsets

In this chapter, we introduce three proficient algorithms named as *THUP-Growth (*temporal High utility pattern growth) for finding high profitable itemsets, TCFP-Growth (temporal closed frequent pattern growth) for finding closed frequent itemsets and finally the information of high utility itemsets is maintained in a tree-based data structure named closed+ utility pattern tree (TCUP-Tree) such that candidate itemsets can be generated efficiently with only two scans of database, then that will be segmented into multiple clusters for fast computation.

The first step of implementation is to segment the transactional dataset into n partitions. This can be done by applying the following formula.

$$\mathrm{P(x)} = \sum_{t=0}^{n} \binom{n}{t} d^{n-t} \qquad (1)$$

From the method 1 transactional database is denoted as d. the number of segments is denoted as n. for each temporal threshold t, this performs partition function p(x). After segmenting the dataset, the high closed utility mining will be generated for each segment. This act as a ensemble approach.

The next step of the implementation is to mine closed + frequent+ utility combination from the transactional dataset. After successful preprocessing and partitioning the data will be allocated to the next phase, which is known as association rule mining. To make the utility pattern mining easier, we focused on the short rule which contains one item on each side of the rule in the rule set. Those rules are easier to understand by the data analyst and more likely to cause a business person to act and change a sales procedure. In order to select the most interesting association rules for finding high utility, closed and frequent item sets, this uses the iteration level data pattern analysis. This focuses on two different THUP-support measures which are as follows

The THUP- support-min measures rely on a minimum cost function. In addition the occurrence of an item set in a given transaction is weighted by the weight of its high utility and interesting item.

Ex: -	X-120
	Y-30
	Z-26

The THUP-support-min measure, which relies on a maximum cost function in addition the occurrence of an item set in a given transaction, is weighted by the number of unit in every transaction of the most interesting item. In the process of dealing optimization problems in market analysis, minimum and maximum are the largely used cost functions. So they are considered as suitable for driving the selection of a worthwhile subset of utility weighted data correlations. There are some problems have been addressed the problems are listed below.

A. THUP and Minimal THUP mining driven by a minimum THUP-support-min threshold, and

1	$\langle a,0\rangle\ \langle b,100\rangle\ \langle c,57\rangle\ \langle d,71\rangle$
2	$\langle a,0\rangle\ \langle b,43\rangle\ \langle c,29\rangle\ \langle d,71\rangle$
3	$\langle a,43\rangle\ \langle b,0\rangle\ \langle c,43\rangle\ \langle d,43\rangle$
4	$\langle a,100\rangle\ \langle b,0\rangle\ \langle c,43\rangle\ \langle d,100\rangle$
5	$\langle a,86\rangle\ \langle b,71\rangle\ \langle c,0\rangle\ \langle d,71\rangle$
6	$\langle a,57\rangle\ \langle b,71\rangle\ \langle c,0\rangle\ \langle d,71\rangle$

TABLE 1 sample transactional table

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

The first task which is the THUP support min threshold requires discovering THUPs and maximal THUPs (MTUIs) which include the item(s) with the most local interest and high utility within each transaction.

THUP	Utility Item –support-max
{c}	172 (maximum)
{a, b}	128 (maximum)
{a, c}	143(maximum)

Table: 2 high utility items, item sets with Minimum threshold >20

Table 2 reports the THUPs mined from Table 1 by enforcing a minimum THUP-support-min threshold equal or greater than 20 and their corresponding THUP-support minimum values. For instance {a,b} covers the transactions with tids 1, 2, 3, and 4 with a minimal weight 0 (associated with a in tids 1 and 2 and b in tids 3 and 4), while it covers the transactions with tids 5 and 6 with maximal weights 86 and 71, respectively.

Hence, its THUP-support-min value is 128 frequent, based on this closed and high utility itemset can be found from dataset THUPs. in Table 2 represent sets of items which contain at least one high utility purchased or highly purchased at each transaction.

As an extreme case, {a,c,d} has TUIsupport- maximum utility because at every sampled point of time at least one between a, b, or d (not necessarily the same at each instant) is null, possibly due to system temporal changes. From the table 2 {b} is high utility item with 385 and {a,c,d=243 }is high utility itemset.

Considering minimal THUPs allows the expert to focus the attention on the highly purchased item sets that contain highly transaction/null transaction and, thus, reduces the unfairness due to the possible insertion of highly weighted items in the extracted patterns. In Table 2 THUPs are partitioned between minimal and not in the transaction, as indicated next to each itemset THUP-support-min value (minimal/not minimal).

The next task requires discovering TCUPs and TCFP which include item(s) having maximal local interest within each transaction by exploiting the TCFP -support-min measure.

Table 3 reports the THUPs mined from Table 1 by enforcing a minimum THUP-support-min threshold equal to 120. They may represent sets of itemsets which contain only high utility items at each temporally splitted data instant is specified in column 1 and 2. This helps to identify the frequent high utility combination of items from the above analysis. The third column describes the maximum frequent and high utility closed item. Finally {d} has the maximum in all three.

Items	THUP-support-Min	TCFP (maximum qty)
{a}	286	{a,d} 200
{b}	285	{b,d} 171
{c}	172	{c,d} 143
{d}	427	{d} frequency=6 Utility=427

Table 3: Minimum THUP-support-min threshold =120

ii. THUP – Growth Tree Algorithm:

Input: filtered transaction from temporal scheme

Output: Frequent item set

Description: THUP -Growth: Allows frequent itemset discovery without candidate itemset generation. Like FP growth algorithm.

THUP allows identifying the utility item set discovery without candidate itemset generation

Steps:

(i) Build a compact data structure called the THUP tree built using 2 passes over the data-set.

(ii) Extracts frequent itemsets directly from the THUP -tree traversal through the Tree.

After the tree construction based on temporal basis, the system performs pruning process. This is done because of the following reasons.

- Pruning may be able to build a perfect decision tree which perfectly reflects the high utility and most closed item sets from the transactional data.
- In some cases, the tree may over fitting i.e. not be generally applicable and updatable.
- This helps to simplifying the tree based results and hence simplifying a transactional decision tree

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

The pruning process has done by applying prune rules over the tree data. The pruning process will be carried out by using the following algorithm.

THUP support mining algorithm

Input: Weighted transactional dataset (td)

Minimum utility support threshold (St)

Minimum Temporal threshold (T)

Step 1: read all transaction data (td) from the shopping cart

Step 2: get the temporal threshold (T) and segment the td. $T(td)=\sum(I \text{ to } n)\ \{Trans_i(split(T))\}$

Step3: scan transactional database Ttd and count THUP support of each item

Step 4: count item THUP support (td)

Step 5: create the initial THUP tree from the td.

Step 6: for each transaction ti in td, Insert ti in Tree

Step 7: store tree, St, null in a tree (which satisfies St)

Step 8: return the output from step 7.

a. Prune rules:

Pruning is the common frame for avoiding the problem of *over fitting* noisy data which is also known as unwanted repeated data. The basic idea is to incorporate a bias towards simpler theories in order to avoid complex rules with low coverage that contain irrelevant literals that have only been added to exclude noisy examples.

Algorithm: THUPMining

Description: The THUPMining algorithm takes three parameters one is the tree from the THUP support algorithm. Another one is minimum THUP support threshold. And finally perform prefix process.

Input: a FP tree (Htree)

Minimum THUP support threshold (St)

The set of items with patterns (prefix)

Output: F values which is a set of THUP extending prefix

Steps:

```
1. Initially assign 0 for F
        i.   F=0
2. For each item I in the header tree table HTree.
3. I=prefix U{i}-generate a new itemset I by joining prefix and I with THUP support set to the THUP support item i
4. If I is infrequent
        i.   Store I.
5. End if
6. If THUP-support(I) <= St then
        i.   F=FU{I}
7. End if
8. Conditional_pattern (P)=generate(Htree, ,I)
9. HTree_I=createFP-tree(Conditional_pattern)
10. Perform pruning
11. Prune=identify(Htree I, St)
12. Htee=remove(HtreeI, prune)
13. If HTreeI #0 then
        i.   F=F U THUPMining(HTree, St, I)
14. End
15. Return the output F.
```

The below fig 1: shows the execution time performance of the proposed system and it proves that it outperforms the earlier systems.

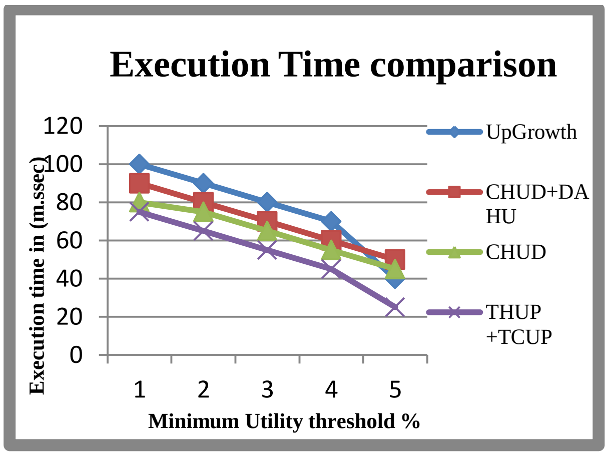


Fig 1 Execution comparison chart

IV. CONCLUSION

The system proposed temporal high utility data set and low utility data set in the high dimensional transactional dataset. Temporal High utility item set mining from a transactional database helps to discover the items with high utility based on different parameter have been proposed. The situation may become worse when the database contains lots of datasets, long transactions or long high utility item sets. The proposal introduced two algorithms which are temporal utility pattern growth (TUP-Growth) and temporal UP-Growth+, for mining high utility item sets with a set of effective strategies for pruning candidate item sets rapidly. The information of

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

high utility itemsets is maintained in a tree-based data structure named utility pattern tree (TUP-Tree) such that candidate itemsets can be generated efficiently with only two scans of database, then that will be segmented into multiple clusters for fast computation. The proposed algorithms reduce the number of candidates and database scans effectively. The experiments' and results shows the proposed system performs better than existing system.

REFERENCES

[1] Tseng, Vincent S., et al. "Efficient algorithms for mining high utility itemsets from transactional databases." *Knowledge and Data Engineering, IEEE Transactions on* 25.8 (2013): 1772-1786.
[2] Liu, Mengchi, and Junfeng Qu. "Mining high utility itemsets without candidate generation." *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012.
[3] Wu, Cheng Wei, et al. "Efficient mining of a concise and lossless representation of high utility itemsets." *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011.
[4] Gouda, Karam, and Mohammed Zaki. "Efficiently mining maximal frequent itemsets." *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001.
[5] Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994.
[6] Boulicaut, Jean-François, Artur Bykowski, and Christophe Rigotti. "Free-sets: a condensed representation of boolean data for the approximation of frequency queries." *Data Mining and Knowledge Discovery* 7.1 (2003): 5-22.
[7] Calders, Toon, and Bart Goethals. "Mining all non-derivable frequent itemsets."*Principles of Data Mining and Knowledge Discovery*. Springer Berlin Heidelberg, 2002. 74-86.
[8] Liu, Mengchi, and Junfeng Qu. "Mining high utility itemsets without candidate generation." *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012.
[9] Chuang, Kun-Ta, Jiun-Long Huang, and Ming-Syan Chen. "Mining top-k frequent patterns in the presence of the memory constraint." *The VLDB Journal* 17.5 (2008): 1321-1344.