ISSN:0975-9646

Efficient High Utility Itemset Mining using extended UP Growth on Educational Feedback Dataset

Yamini P. Jawale[1], Prof. Nilesh Vani[2]

[1]*Reasearch Scholar,*
Godawari College of Engineering,Jalgaon.

[2]*Research Guide, Department of Computer Engineering,*
Godawari College of Engineering,Jalgaon.

***Abstract* - High utility itemsets refer to the sets of items with high utility like profit in a database, and efficient mining of high utility itemsets plays an important role in many real life applications and is an important research issue in data mining area. In recent years, the problems of high utility pattern mining become one of the most important research areas in data mining. The existing high utility mining algorithm generates large number of candidate itemsets, which takes much time to find utility value of all candidate itemsets.**
In this paper we are implementing a data structure that stores the utility related to the item and using this data structure we are reducing time and space complexity of UP Growth and UP Growth+ Algorithms. Various Standard and synthetic datasets are used with Educational feedback data set. An algorithm is proposed to find set of high utility itemset which avoids the candidate itemsets generation.

***Keywords*- Utility, Utility Information Record, Effective High Utility Itemset Mining.**

I. INTRODUCTION

Rapid development in database techniques facilitates storage and usage of data from large database and also to mine the same. How to obtain valuable information from database is a more crucial task today which results in a rise of research topics [1].

Mining frequent itemset [2] from the database DB is to find out set of itemset that occurs frequently. The frequency of itemset is the support count related to that itemset i.e. number of transactions containing that itemset. If the support of the itemset exceeds the *minimum support threshold* value then itemset is frequent.

Mining frequent itemset on takes presence and absence of itemset into account, other relative information related to the item is not considered. This results in the research area of finding out high utility itemset from database. Utility is one of the important features of itemset in transaction that specifies a utility/profit of itemset with frequency [6].

Table 1: External Utility table

Item	A	B	C	D	E	F	G	H
Profit	5	2	1	2	3	5	1	1

Table 2 : Internal Utility Example

TID	Transaction	Quantity	TU
T_1	{A,C,D,}	{1,10,1}	17
T_2	{A,C,E,G}	{2,6,2,5}	27
T_3	{A,B,D,E,F}	{2,2,6,2,1}	37
T_4	{B,C,D,E}	{4,13,3,1}	30
T_5	{B,C,E,G}	{ 2,4,1,2}	13
T_6	{A,B,C,D,H}	{1,1,1,1,2}	12

Recently, a number of high utility itemset mining algorithms [3] have been proposed. Most of the algorithms adopt a similar framework: firstly, generate candidate high utility itemsets from a database secondly, compute the exact utilities of the candidates by scanning the database to identify high utility itemsets [7]. However, the algorithms often generate a very large number of candidate itemsets and thus are confronted with two problems:

(1) Excessive memory requirement for storing candidate itemsets.
(2) A large amount of running time for generating candidates and computing their exact utilities.

When the number of candidates is so large that they cannot be stored in memory, the algorithms will fail or their performance will be degraded due to thrashing. To solve these problems, we propose in this paper an algorithm for high utility itemset mining [6].

The contributions of the paper are as follows:

1. A novel structure, called *effective information list*, is proposed. effective information list stores not only the utility information about an itemset but also the heuristic information about whether the itemset should be pruned or not.
2. An efficient algorithm, called Efficient High Utility Itemset Mining (EHUIM) Algorithm, is developed. Different from previous algorithms, EHUIM Algorithm does not generate candidate high utility itemsets. After constructing the initial utility-record from a mined database, EHUIM Algorithm, can mine high utility itemsets from these utility-record. We are using various standard and real data sets [4].

II. BACKGROUND

A. *Problem Definition*

Let I={i1, i2, i3, . . . , in} be a set of items and DB be a database composed of a utility table and a transaction table. Each item in I has a utility value in the utility table. Each transaction T in the transaction table has a unique identifier(tid) and is a subset of I, in which each item is associated with a count value. An itemset is a subset of I and is called a k-itemset if it contains k items.

Definition 1. The external utility of item i, denoted as ext_util(i), is the utility value of i in the utility table of DB.

Definition 2. The internal utility of item i in transaction T, denoted as int_util(i, T), is the count value associated with i in T in the transaction table of DB.

Definition 3. The utility of item i in transaction T, denoted as util(i, T), is the product of int_util(i, T) and ext_util(i), where util(i, T) = int_util(i, T) × ext_util(i).

For example, in Table 2 and 3, ext_util(e) = 3, int_util(e, T5) = 1, and util(e,T5)= int_util(e, T5) × ext_util(e) = 1 x 3 =3.

Definition 4. The utility of itemset X in transaction T, denoted as util(X, T), is the sum of the utilities of all the items in X in T in which X is contained, where u(X, T) =$\sum_{i \in X \wedge X \subset T} u(i, T)$

Definition 5. The utility of itemset X, denoted as u(X),is the sum of the utilities of X in all the transactions containing X in DB, where u(X) = $\sum_{T \in DB \wedge X \subset T} u(X; T)$

For example, in Table 2, u({ae}, T2) = u(a, T2) + u(e, T2)= 2 × 5 + 2 × 3 = 16, and u({ae}) = u({ae}, T2) + u({ae},T5) = 16 + 14 = 30.

Definition 6. The utility of transaction T, denoted as tu(T), is the sum of the utilities of all the items in T, where tu(T) =$\sum_{i \in T} u(i, T)$,

and the total utility of DB is the sum of the utilities of all the transactions in DB.

B. *Related Work*

Many algorithms have been proposed for high utility itemset mining but all they first produce candidate itemset which require more time and space. Here in this algorithm a search space from the UP Growth algorithm [5] is minimized. A effective information list structure is used instead of UP Tree.

III. PROPOSED METHOD

The framework of the proposed method consist of following steps: 1) Scan database to construct Effective Information List. 2) Apply EHUI mining algorithm. 3) Generate High Utility Itemsets.

A. *Effective Information List Structure*

In the section, we propose a utility information record structure to maintain the utility information about a database [7].

1) Initial Effective Information List

Initial Effective Information List storing the utility information about a mined database can be constructed by two scans of the database. Firstly, the transaction-weighted utilities of all items are accumulated by a database scan. If the transaction-weighted utility of an item is less than a given *minutil*, the item is no longer considered. For the items whose transaction-weighted utilities exceed the *minutil*, they are sorted in transaction-weighted-utility-ascending order.

2) Effective Information List of 2-Itemsets

No need for database scan, the utility information record of 2-itemset{xy} can be constructed by the intersection of the utility list of {x} and that of {y}. The algorithm identifies common transactions by comparing the tids in the two effective information list. Suppose the lengths of the effective information lists are p and q respectively, and then (p + q) comparisons at most are enough for identifying common transactions, because all tids in a effective information list are ordered. The identification process is actually a 2-way comparison.

3) *Effective Information List of k-Itemsets (k≥3)*

To construct the effective information list of k-itemset $\{i_1 \cdots i_{(k-1)} i_k\}(k \geq 3)$, we can directly intersect the utility-list of $\{i_1 \cdots i_{(k-2)} i_{(k-1)}\}$ and that of $\{i_1 \cdots i_{(k-2)} i_k\}$ as we do to construct the utility-list of a 2-itemset.

B. *EHUIM Algorithm*

After constructing a Utility information record a EHUIM Algorithm can mine all high utility itemset from database.

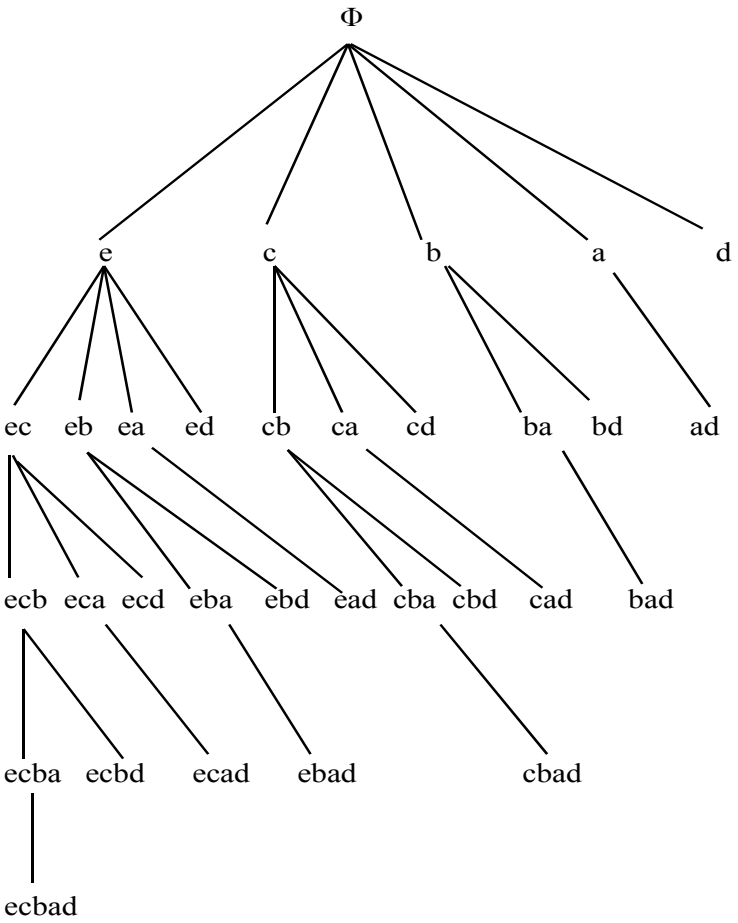
1) Domain Space:

The domain space of the high utility itemset mining problem can be represented as a combination tree. Given a set of items $I = \{i_1, i_2, i_3, \ldots i_n\}$ and a total order on all items (suppose $i_1 < i_2 < \cdots < i_n$), a combination tree representing all itemsets can be constructed as follows.

Firstly, the root of the tree is created; secondly, then child nodes of the root representing n 1-itemsets are created, respectively; thirdly, for a node representing itemset$\{i_s \cdots i_e\}$ $(1 \leq s \leq e < n)$, the (n−e) child nodes of the node representing itemsets $\{i_s \cdots i_e i_{(e+1)}\}$,$\{i_s \cdots i_e i_{(e+2)}\}$, ...,$\{i_s \cdots i_e i_n\}$ are created. The third step is done repeatedly until all leaf nodes are created. For example, given I = {e,c, b, a, d} and e < c < b < a < d, a combination tree representing all itemsets of I is depicted in Figure 1.

2) *pruning Strategy:*

For a database with n items, exhaustive search has to check 2n itemsets. To reduce the search space, we can exploit the iutils and rutils in the utility-list of an itemset. The sum of all the iutils in the utility-list of an itemset is the utility of the itemset according to Definition 5, and thus the itemset is high utility if the sum exceeds a given minutil. The sum of all the iutils and rutils in the utility-list provides EHUIM Algorithm with the key information about whether the itemset should be pruned or not.



Lemma 1. Given the utility information record of itemset X, if the sum of all the iutils and rutils in the utility information record is less than a given "minutil", any extension X' of X is not high utility.

Figure 1: Combination Tree

Algorithm 1: *Build - Combination Algorithm*

Input: *P.EIL*, the effective information list of itemset *P*;
Px.EIL, the effective information list of itemset *Px*;
Py.EIL, the effective information list of itemset *Py*.
Output: *Pxy.EIL*, the effective information list of itemset *Pxy*.

1. *Pxy.EIL = NULL*;
2. **for each** element Ex ∈Px.EIL **do**
3. **if** ∃*Ey*∈*Py.EIL* and *Ex.tid==Ey.tid* **then**
4. **if** *P.EIL* is not empty **then**
5. search such element *E*∈*P.EIL* that *E.tid==Ex.tid*;
6. *Exy=<Ex.tid, Ex.iutil+Ey.iutil -E.iutil, Ey.rutil>*;
7. **Else**
8. *Exy=<Ex.tid, Ex.iutil+Ey.iutil, Ey.rutil>*;
9. **end if**
10. append *Exy to Pxy.EIL*;
11. **end if**
12. **end for**
13. **return** *Pxy.EIL*;

3) EHUI Mining Algorithm:

Algorithm 2 shows the pseudo-code of EHUIM Algorithm. For each effective information list X in ULs (the second parameter), if the sum of all the iutils in X exceeds minutil, and then the extension associated with X is high utility and outputted. According to Lemma 1, only when the sum of all the iutils and rutilsin X exceeds minutil should it be processed further. When the initial utility-lists are constructed from a database, they are sorted and processed in transaction-weighted utility-ascending order. Therefore, all the utility information records in EILs are ordered as the initial utility information record are. To explore the search space, the algorithm intersects X and each effective information list Y after X in EILs. Suppose X is the effective information list of itemset Px and Y that of itemset Py, and then Build(P.EIL, X, Y) in line 8 is to construct the effective information list of itemset Pxy as stated in Algorithm 1. Finally, the set of effective information lists of all the 1-extensions of itemset Px is recursively processed. Given a database and a minutil, after the initial utility information record EILs are constructed, EHUIM(∅, EILs, minutil) can mine all high utility itemsets.

Algorithm 2: *EHUI Mining Algorithm*

Input:*P.EIL*, the effective information list of itemset *P*, initially empty; *EILs*, the set of utility-lists of all *P*'s 1-extensions;
minutil, the minimum utility threshold.
Output: all the high utility itemsets with *P* as prefix.

1. **for each** effective information list *X* in *EILs* **do**
2. **if** SUM(*X.iutils*)≥*minutil* **then**
3. output the extension associated with *X*;
4. **end if**
5. **if** SUM(*X.iutils*)+SUM(*X.rutils)*≥*minutil* **then**
6. *exEILs = NULL*;
7. **for each** effective information list *Y* after *X* in *EILs* **do**
8. *exEILs = exEILs*+Build(*P.EIL, X, Y*);
9. **end for**
10. EHUI(*X*, *exEILs, minutil*);
11. **end if**
12. **end for**

IV. EXPERIMENTAL EVALUATION

Performance of proposed algorithm is evaluated in this section. The experiments were performed on 2.20 GHz Core2 Duo Processor with 3GB memory. The operating system is Windows 7. The algorithms are implemented in Java language. Both real and standard datasets are used in this experiment. Standard data sets are obtained from FIMI Repository. Real datasets were generated from the actual values. Parameter descriptions and default values of datasets are shown in Table no 3. Educational dataset for evaluation of feedback report of faculty member is used as a real dataset.

Table 3: Statistics about Databases

Dataset	Educational Dataset
Size	26kb
Transactions	500
Items	10
Avg Length	10

A. Performance comparison on Educational Feedback data sets

Running Time

For almost all databases and *minutils*, EHUI performs the best. In Figure 3, EHUI is even an order of magnitude faster than UPGrowth and UPGrowth+. For the Educational Feedback Dataset, when *minutils* are 50%, 60%, and 70%, the running time required for EHUI are 40mSec. So running time for EHUI is always less compare to UP Growth and UP Growth+ Algorithms.

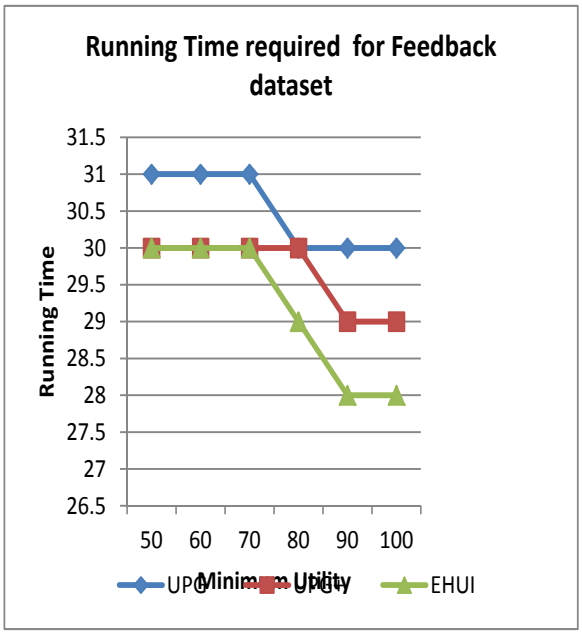


Figure 3: Time for Educational Feedback Dataset

Memory Consumption

Generally, the memory consumption of the algorithms is proportional to the number of candidate itemsets they generate. For example, for database *Chess*, UP Growth generates 623, UP Growth+ generates 551 and that of IHUP generates 30 candidate itemsets and consumes 17.60MB, 21.81MB, and 16.02MB of memory respectively[6]. Similar case is there for Educational Feedback Dataset. EHUI require near about similar space compare to UPGrowth and in some cases of UPGrowth+ algorithm.

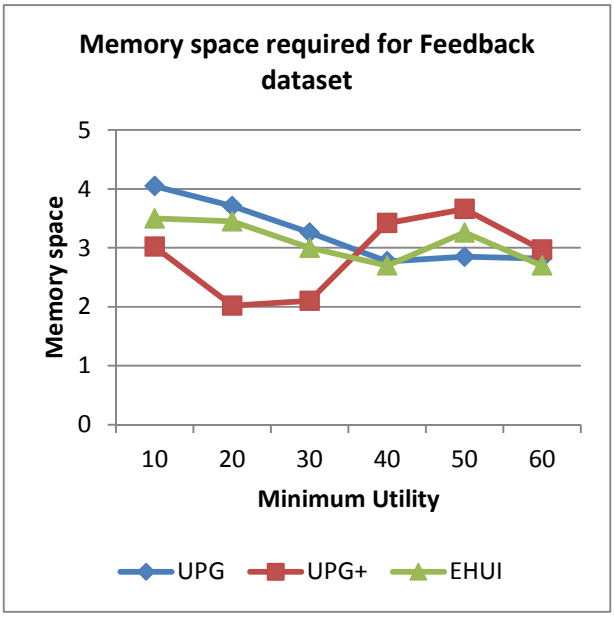


Figure 6: Memory Space for Educational Feedback Dataset

Itemsets Found

Higher the number of candidate itemsets in a algorithm, lower is the performance. EHUI mining generate similar number of itemsets compare to UPGrowth and UPGrowth+.

V. CONCLUSION

In this paper, we have proposed a novel data structure, effective information list, and developed an efficient algorithm, EHUI, for high utility itemset mining. Effective information list provide not only utility information about itemsets but also important pruning information for EHUI. We have used Educational real time and standard datasets. Previous algorithms have to process a very large number of candidate itemsets during their mining processes. However, most candidate itemsets are not high utility and are discarded finally. EHUI Algorithm can mine high utility itemsets without candidate generation, so that complexity of UPGrowth and UPGrowth+ is reduced as it require less time and space, which avoids the costly generation and utility computation of candidates.

REFERENCES

[1] Jyothi Pillai, O.P.Vyas "Overview of Itemset Utility Mining and its Applications" IJCA(0975 – 8887) Volume 5– No.11, August 2010.

[2] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future directions. Data Mining and Knowledge Discovery, 15(1):55–86, 2007.

[3] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K.Lee. Efficient tree structures for high utility patternmining in incremental databases. IEEE Transactions onKnowledge and Data Engineering, 21(12):1708–1721,2009.

[4] Frequent Itemset Mining Implementations Repository, http://fimi.cs.helsinki.fi/, 2013.

[5] Vincent S. Tseng, Bai-En Shie, Cheng Wei Wu, and Philip S. Yu, Fellow, "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 8, AUGUST 2013.

[6] Prashant V. Barhate, S. R. Chaudhari, P. C. Gill, "Efficient High Utility Itemset Mining using Utility Information Record" International Journal of Computer Applications (0975 – 8887) Volume 120 – No.4, June 2015.

[7] Mengchi Liu, Junfeng Qu, "Mining High Utility Itemsets without Candidate Generation" CIKM'12, October 29–November 2, 2012