International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-4, April 2014

Mining of High Utility Itemsets from Transactional Databases

D. Usha Nandini, Ezil Sam Leni, M. MariaNimmy

Abstract—Efficient discovery of high utility itemsets from transactional databases crucial task in data mining. UP-Growth and UP-Growth+ algorithms are proposed for mining high utility itemsets. In this paper we also proposed a compact tree structure, called Utility pattern tree (UP-Tree) and it maintains the information of high utility itemsets. Previously we proposed FP-Growth algorithm for mining only large number of frequent itemsets, but not generate the high utility itemsets. They have the issue of producing large number of candidate itemsets and probably it degrades mining performance in terms of speed and space requirement. However, our previous study needs more space and execution time. Many algorithms are used to show the performance of UP-Growth and UP-Growth+. UP-Growth and UP-Growth+ becomes more efficient since database contain long transactions and generate fewer number of candidates than FP-Growth. The experimental results and comparison validate its effectiveness.

Index Terms—Candidate pruning, Data mining, Frequent itemset, High utility itemset.

I. INTRODUCTION

Data mining is the extraction of hidden previously unknown and potentially useful information from databases. Data mining is sometimes called as data or knowledge discovery. Data mining is mainly used for analyze the data in a different way. Based on user queries it analyzes the relationships in stored data. Data mining is widely used in financial data analysis, retail industry, intrusion detection and other scientific applications. Data mining models can be applied to specific scenarios such as forecasting, risk and probability.

To discover the useful patterns from database, frequent pattern mining has been applied to different databases. In recent years, finding of frequent patterns from large databases is very important use full in many applications. The goal of frequent itemset mining is to identify all frequent itemsets and it collects the set of items that occur frequently together. The information of frequent set of items is presented as collection of if-then rules. The generations of association rules are straight forward, for finding the association and correlation relationship among the items.

However the unit profits and the purchased quantities are not considered in the frequent itemsets mining. Therefore, frequent itemset mining cannot not satisfy the needs of customers, who all are wanted the itemsets with high profits.

Utility mining concepts is used in data mining for discovering itemsets with high utility like high profits. We can determine utility of the item based on customer behaviors and interestingness.

High utility itemsets mining finds the itemsets, which having high profit and usefulness to users. Two aspects of this itemset are local utility and external utility. Transaction utility is defined as product of local utility and external utility. Transaction weight utilizations (TWU) can be found by this transaction utility. If the utility of an itemset is no less than a user specified minimum utility threshold is called high utility itemset. Otherwise, the itemset is called a low utility itemset.

Manuscript published on 30 April 2014.

* Correspondence Author (s)

F. D. Usha Nandini, Asst. Prof & Research Scholar, Department of CSE, Sathyabama University, Chennai, Tamil Nadu, India.

S. Dr. Ezil Sam Leni, Asst. Prof & HOD, Department of CSE, SRR Engineering college, Chennai, Tamil Nadu, India.

T. M. Maria Nimmy, Pursuing M.E in CSE at Sathyabama University, Chennai, Tamil Nadu, India.

II. BACKGROUND

The set of items $I = \{i_1, i_2, \ldots, i_m\}$. X is an itemset. The set of k distinct items $\{i_1, i_2, \ldots, i_k\}$. An itemset X is with length k is called k-itemset. D is a transaction database it contains $T_1, T_2, \ldots, T_n$. where each transaction belongs to database D. Each transaction T_d $(1 \leq d \leq n)$ has a unique identifier d, called TID. Each item i_p associated with quantity q (i_p, T_d), that is the purchase number of items. Consider the database with 6 transactions and 8 items. To quantify the usefulness or preferences of items, different values are used. Some definitions of this paper is given in below.

Definition 1: The quantitative measure of utility for item i_p in transaction T_d, is defined as $u(i_p, T_d)$. For example, $u(\{C\}, T_1) = 9\text{x}1 = 9$, in Table 1.

Definition 2: The quantitative measure of utility for itemset X in transaction T_d, is defined as $u(X, T_d)$. For example, $u(\{AD\}, T_1) = u(\{A\}, T_1) + u(\{D\}, T_1) = 5 + 2 = 7$, in Table 1. For example, $u(\{AD\}) = u(\{AD\}, T_1) + u(\{AD\}, T_3) + u(\{AD\}, T_6) = 7 + 20 + 7 = 34$.

Definition 3: Each transaction Td, Internal utility value of item i_p is denoted as iu (i_p, T_d), is the value of i_p in T_d. For example, in Table1, iu$(C, T_2) = 5$.

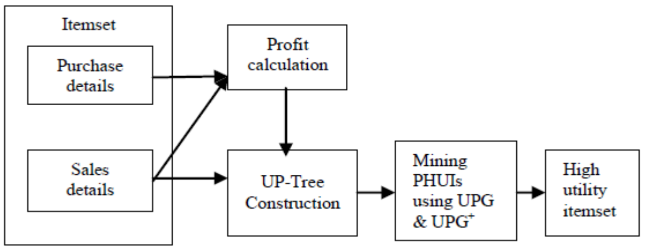


Fig. 1. Block Diagram

TABLE I: A SAMPLE DATABASE

TID	Transaction	TU
T_1	(A,1) (C,10) (D,1)	16
T_2	(A,2) (C,6) (E,2) (G,5)	23
T_3	(A,2) (B,2) (D,6) (E,2) (F,1)	35
T_4	(B,4) (C,13) (D,3) (E,1)	25
T_5	(B,2) (C,4) (E,1) (G,2)	11
T_6	(A,1) (B,1) (C,1) (D,1) (H,2)	14

TABLE II: PROFIT TABLE

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

Definition 4: Each transaction database, external utility of item i_p denoted as eu (i_p), is the value of i_p in the utility table of the database. For example, in Table 2, eu(A) = 5 & eu(D) = 2.
Definition 5: If the utility of an itemset is no less than a user specified minimum utility threshold is called high utility itemset. Otherwise, the itemset is called a low utility itemset.
Definition 6: Transaction utility of a transaction T_d is defined as u(T_d, T_d) and denoted as TU(T_d).For example, TU(T_2) = u({ACEG}, T_2) = 10.
Definition 7: The total sum of TU of all the transactions containing X is called as transaction weighted utility (TWU), which is denoted as TWU(X).If TWU(X) is no less than the minimum utility, X is called a high transaction-weighted utilization itemset.
Definition 8: If TWD(X) is no less than min_util then itemset X is called a High Transaction Weighted Utility Itemset.
Property 1: It is denoted as TWDC, If X is not a HTWUI any superset of X is a low utility itemset. The downward closure property can be maintained by using transaction weighted utilization. For example TU (T_2) = u ({ACEG}), T_2) = 23. TWU ({G}) = TU (T_2) + TU (T_5) = 23 + 11 = 34. If we set min_util is 50, {F}, {G} and {H} are not HTWUIs.

III. LITERATURE SURVEY

Previous studies have been used for mining only frequent itemsets. For this mining we use many mining algorithms like sequential pattern mining [3], frequent pattern mining and association rule mining [1]. To identify the frequent itemsets an association rule mining [1] has been proposed. This mining solves the issue of identifying the association rule between the items. We show the best results with real or synthetic data sets by applying two novel algorithms. The two proposed algorithms can be called as Apriori-Hybrid. We can get good scalability using this algorithm that is large number transaction it can handle.
To identify the significant binary relationship in a weighted setting, W. Wang [5] proposed mining association rules with weighted items. Downward closure property cannot be applied in this paper. The problem of downward closure property is improved by a new algorithm, named WARM (Weighted Association Rule Mining). Scalability and efficient identification of relationships in weighted setting can be improved.
J. Han et al in [5] proposed an FP-Growth algorithm to find the frequent itemset without generating any candidates. To maintain the information of itemsets, we build a compact data structure called FP-Tree. During the construction of FP-Tree it first scan data and find support count for each item. During this scanning it discard infrequent items and sort frequent items in decreasing order based on their support. FP-Growth is used for only mining frequent itemsets and does not have the ability to mine the high utility itemsets.
Liu et al in [4] proposed a Two-phase algorithm for pruning the number of candidates and obtains the complete set of high utility itemsets. It finds the utility of all itemsets by level wise search, and their TWU of all itemsets are computed. Then the complete set of High transaction weighted utility itemsets is identified. In additional database is required to filter the overestimated itemsets. Search space can be reduced and it requires only fewer database scans. However, this two-phase algorithm must scan the whole database again and again when added any new transaction from data streams.
Ahmed et al. [15] proposed an IHUP algorithm for generate the HTWUIs and avoid scanning database several times. IHUP achieves a better performance than Two-Phase algorithm. IHUP and Two-Phase produce same number of HTWUI. The critical issue for the performance of algorithms, it generates too many HTWUIs. Therefore, it aims at reducing itemsets overestimated utilities and proposes several strategies. The number of generated candidates can be reduced and high utility itemsets can be discovered by applying the proposed strategies.

IV. PROPOSED METHODS

In this section two algorithms have been proposed for mining high utility itemsets, namely UP-Growth and UP-Growth$^+$. In this proposed method, we first introduce a proposed UP-Tree structure for maintain the information of itemsets.

A. *The proposed UP-Tree*

The UP-Tree is constructed for improving the mining performance. UP-Tree maintains the information of all high utility itemsets from databases. The framework of UP-Tree follows three steps:
1) Construction of UP-Tree [11].
2) Generate Potential High Utility Itemsets (PHUIs) from UP-Tree.
3) Identify the high utility itemsets using PHUI.

Construction of global UP-Tree has two strategies (DGU & DGN) for decreasing the overestimated utility of each item during the construction of a global UP-Tree.
1) Discarding global unpromising items (i.e., DGU strategy)
Is to eliminate the low utility items and their utilities from the transaction utilities
2) Discarding global node utilities (i.e., DGN strategy) during global UP-Tree construction.

Definition 11: An item ip is called a promising item if its overestimated utility is no less than min_util. Otherwise the item is called as an unpromising item.
All supersets of unpromising item are not high utility items. The supersets of promising item are high

TABLE III: REORGANIZED TRANSACTION AND THEIR RTUS

TID	Transaction	TU
T_1	(A,1) (C,10) (D,1)	16
T_2	(A,2) (C,6) (E,2)	18
T_3	(A,2) (B,2) (D,6) (E,2)	30
T_4	(B,4) (C,13) (D,3) (E,1)	25
T_5	(B,2) (C,4) (E,1)	9
T_6	(A,1) (B,1) (C,1) (D,1)	12

TABLE IV: MINIMUM ITEM UTILITY TABLE

Item	A	B	C	D	E
Profit	5	2	1	2	3

Utility itemsets. After pruning the unpromising items we reorganize the transaction T_r, it is called as Reorganized Transaction Utility (RTU). The transaction of RTU is denoted as RTU (T_r). Now the strategy DGU uses this RTU only instead of TWU to overestimate the utilities. The strategy DGU can be performed until it contains no unpromising item in reorganized transactions.

By the property 1 we know the unpromising items from the table 1. The TWU ({F}, {G}, {H}) is below than the min_util by the property 1. So the items F, G, H in corresponding transactions are the unpromising items. We can remove the items from the database in table 1. Now the transaction utility (TU) is reorganized in the following table 3, after new TU is inserted. The RTU (T_2), RTU (T_3), RTU (T_5) and RTU (T_6) have been modified, because the TWU ({F}, {G}, {H},) is less than the min_util.

B. UP-Growth

The UP-Growth [9] is one of the most efficient algorithms to generate high utility itemsets from transactional databases depending on construction of a global UP-Tree.

By the strategy DGN, the node utilities which are closer to UP-Tree root node are effectively reduced. The PHUI is similar to Transaction Weighted Utility (TWU), which compute all itemsets utility with the help of estimated utility. Finally, identify high utility itemsets which is not less than min_util from PHUIs values. Here, a header table is used to facilitate the traversal of UP-Tree. The Header table has entry of each item with its name, an overestimated utility, and a link to other node.

The global UP-Tree contains more sub paths. Each sub path in the global UP-Tree is considered from bottom node of the header table and this path is named as conditional pattern base (CPB). Candidates will be generated too many, after constructing a global UP-Tree. For avoiding the more generated candidates we propose an UP-Growth algorithm for efficiently generating PHUIs from the global UP-Tree with two strategies (DLU and DLN) in the framework of UP-Tree. The itemsets with overestimated utilities can be decreased and the number of PHUIs also can be reduced by the two strategies.

Subroutine: UP-Growth (T_s, H_s, S)
Input: a header table Hs for T_s, A UP-Tree T_s, an itemset S, min_util.
Output: All Potential High Utility Itemsets in T_s.

```
1. for each entry $i_n$ in Hs do.
2. Trace each node related to item $i_n$ via $i_n$.hlink and calculate
     $i_n$.nu to $nu_{sum}(i_n)$;
   /* $nu_{sum}(i_n)$: the sum of node utilities of $i_n$*/
3. If $nu_{sum}$ $(i_n) \geq$ min_util, then do
4. Generate a PHUI I = S$\cup i_n$ ;
5. Set pu ($i_n$) as estimated utility of Y set pu ($i_n$);
6. Construct I-CPB;
7. Put local promising items in I-CPB into Hm.
8. Apply DLU to reduce path utilities of the paths;
9. To insert paths into Tm with DLN apply
   Insert_Reorgnized_Path;
10. If $T_m \neq$ null then call UP-Growth ($T_m$, $H_m$, I);
11. End if
12. End for
```

Fig. 1. The subroutine of UP-Growth

C. An Improved Mining Method: UP-Growth$^+$

Although DGU and DGN strategies are efficiently reduce the number of candidates (i.e., global UP-Tree). But they cannot be applied during the construction of the local UP-Tree UP-Growth algorithm gives better performance than FP-Growth by using DLU and DLN to decrease overestimated utilities of itemsets and however, the overestimated utilities can be closer to their actual utilities by eliminating the estimated utilities that are closer to actual utilities of unpromising items and descendant nodes. To reduce overestimated utilities more effectively, we propose an improved mining method, named UP-Growth$^+$. Minimum item utility table is used to reduce the overestimated utilities in UP-Growth. But in UP-Growth$^+$, minimal node utilities in each path are used to make the estimated pruning values closer to real utility values of the pruned items in database.

During construction of a global UP-Tree the minimal node utility for each node can be acquired. After the modification of global UP-Tree, the process is done by two strategies (DNU and DNN) of UP-Growth$^+$. By this strategy we can simply replace minimum item utility with minimal node utility. Strategy DNU is used for discarding local Unpromising items and their estimated node utilities and strategy DNN is used to decreasing local Node utilities for the nodes of local UP-Tree by estimated utilities of descendant Nodes. After mining the full UP-Tree by this UP-Growth$^+$, we can identify all the PHUIs in the UP-Tree. The number of PHUIs of UP-Growth$^+$ is less than the Up-Growth. The overestimated utilities of itemsets are further reduced by UP-Growth$^+$.

Identify High Utility Itemsets

After finding the all potential high utility itemsets, we can identify all the high utility itemsets by scanning the database only once. This is called phase II. In phase I, the number of generated candidates is less but in phase II HTWUIs is too large. In this, high utility itemsets can be identified by scanning RT (Reorganized Transaction). In the reorganized transaction, there is no unpromising item.

V. IMPLEMENTATIONS

The implementation of the proposed system is evaluated and shows that the proposed system outperform the existing systems. In this section implementation on real dataset and synthetic data sets are summarized on UP-Growth & UP-Growth+ algorithm.

TABLE V: PARAMETER SETTINGS OF SYNTHETIC DATASETS

Parameter settings	Default
|D|: Total number of transactions	100k
T: Average length transaction	10
|I|: Number of distinct items	1000
F: Average size of maximal potential frequent itemsets	6
Q: Maximum number of purchased items in transaction	10

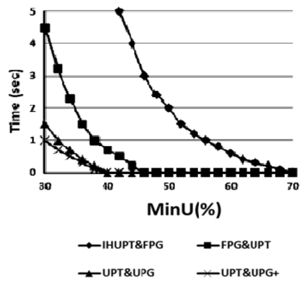
TABLE VI: REAL DATASET CHARACTERISTICS

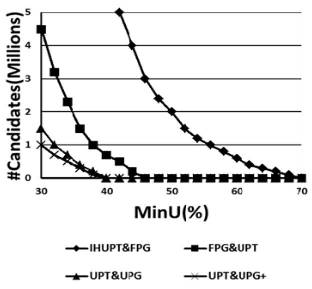
Dataset	|D|	T	|I|	Type
Chain-store	1,112,949	7.2	46,086	Sparse
Chess	3,196	37.0	75	Dense
Food mart	4,141	4.4	1,559	Sparse

UPG & UPG+ is the best algorithm or finding high utility itemsets. The algorithms of this paper are implemented in .NET language. The operating system is Microsoft Windows 7, 8 or xp. Parameter settings of the synthetic datasets are given in the above table 5.

A. *Performance Comparison*

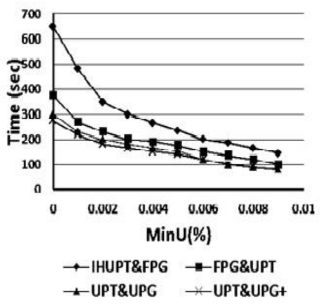
Here, we show the performance comparison on three real data sets. They are dense data set Chess and sparse data set Chain-Store and Foodmart. IHUP$_{TWU}$-Tree, IHUP$_{TWU}$ algorithm proposed in [6] and FP-Growth, is denoted as IHUPT & FPG. We generate two new methods based on UP-Growth (with DGU, DGN, DLU, DLN), UP-G$^+$ and UP-Tree respectively, namely UPT & UPG and UPT & UPG$^+$. First we compare the performance on real dense data set Chess in fig 3. We can identify that it outperforms that of previous methods. In fig. 3.a the runtime of the UPT&FPG, UPT&UPG and UPT&UPG$^+$ is best compared with IHUPT&FPG. Experimental results on real sparse data sets are shown in Fig. 4. In Fig. 4. (a) and (b) performance on Chain store dataset is shown. In Fig. 4 (a), the runtime of IHUPT&FPG is the worst, followed by UPT&FPG, UPT&UPG and UPT&UPG+ is the best. The execution time of UPT&FPG is the worst, since UP-Growth+ and UP-Growth efficiently prune the search space of local UP-Trees. The performance is decided by the number of generated candidates. In fig.1 runtime is proportional to their number of candidates. Therefore more candidates the method produces, its execution time is greater.
In fig 5 experimental results of phase II are shown. We show the results on foodmart only, because the run time for phase II is very long for large databases like Chain-Store. The runtime for phase II is not only proportional to number of candidates in phase II but also increases fiercely. The performance is highly dependent on the runtime in phase II since the overhead of scanning database is large.

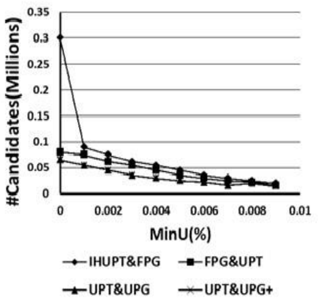




(a). Runtime for phase I on chess (b). Number of candidates on chess

Fig. 3. Performance comparison on dense data set





(a). Runtime for phase I on chess (b). Number of candidates on chess

Fig. 4. Performance comparison on sparse data sets

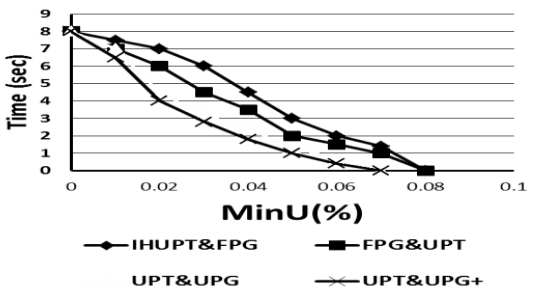


Fig. 5. Performance comparison of runtime for phase II

B. *Scalability and Memory usage of proposed methods*

In this section we can observe that all compare algorithms have good scalability on runtime. The runtime of UPT&UPG$^+$ is best compared with UPT&FPG and UPT&UPG. The performance of UPG&UPG$^+$ outperforms the other algorithms with increasing size of databases since it generates the least PHUIs in phase I. Generally UP-Growth+ outperforms UP-Growth but they have tradeoffs on memory usage. Because that UP-Growth+ utilizes minimal node utilities for decreasing overestimated utilities of itemsets. UP-Growth performs better only when min_util is small. High utility itemsets are efficiently identified from the set of PHUIs which is much smaller than HTWUIs generated by IHUP. The two proposed algorithms UP-Growth and UP-Growth+ achieve better performance than IHUP algorithm.

REFERENCES

[1] R. Srikant and R. Agrawal, “Fast algorithms for mining association rules,” in Proc. of The 20th VLDB Conf,” pp. 487-499, 1994.

[2] R. Agrawal, Imielinski. T and A. Swami, “Mining association rules between sets of items in large databases”, in proceedings of the ACM SIGMOD International Conference on Management of data, pp. 207-216, 1993.

[3] R. Agrawal and R. Srikant, “Mining Sequential Patterns,” in Proc. of the 11th Int’l Conference on Data Engineering, pp. 3-14, Mar 1995.

[4] Liu. Y, Liao. W, A. Choudhary, ”A Fast High Utility Itemsets Mining Algorithm,” In: 1st Workshop on Utility-Based Data Mining. Chicago Illinois, 2005.

[5] A.W.C. Fu, C.H. Cai, C.H. Cheng, and W.W. Kwong, “Mining Association Rules with Weighted Items,” Proc. Int’l DB Engineering and Apps Symp. (IDEAS ’98), pp. 68-77, 1998.

[6] Tanbeer. S.K., C.F. Ahmed, and Y.-K. Lee, “Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases,” IEEE Trans. Knowledge and Data Engineering, vol. 21, no. 12, pp. 1708-1721, Dec. 2009.

[7] J. Yang, W. Wang, and Yu. P, “Efficient Mining of Weighted Association Rules (WAR),” Proc. ACM SIGKDD Conference. Knowledge Discovery and Data Mining (KDD ‘00), pp. 270-274, 2000.
[8] Tao. F, Farid. M, and F. Murtagh, “Weighted Association Rule Mining Using Weighted Support and Significance Framework,” Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD ’03), pp. 661-666, 2003.
[9] Erwin. A, Gopalan. R.P, Achuthan. N.R., “A Bottom-Up Projection Based Set of rules for Mining High Utility Itemsets,” In: International Workshop on Integrating AI and Data Mining. Gold Coast, Australia, 2007.
[10] Tseng V.S, C.W. Wu, B.E. Shie, and P.S. Yu, “UP-Growth: An Efficient Algorithm for High Utility Itemsets Mining,” Proc. 16th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD ’10), pp. 253-262, 2010.
[11] Han. J, J. Pei, Yin. Y, “Mining frequent patterns without candidate generation,” In: ACM SIGMOD International Conference on Management of Data, 2000.
[12] S.J. Yen and Y.S. Lee, “Mining High Utility Quantitative Association Rules.” Proc. Ninth Int’l Conf. Data Warehousing and Knowledge Discovery (DWK), pp. 283-292, Sept. 2007.
[13] U. Yun, “An Efficient Mining of Weighted Frequent Patterns with Length Decreasing Support Constraints,” Knowledge-Based Systems, vol. 21, no. 8, pp. 741-752, Dec 2008.
[14] Y.-C. Li, C.C. Chang and J.S. Yeh, “Isolated Items Discarding Strategy for Discovering High Utility Itemsets,” Data and Knowledge Engineering, vol. 64, no. 1, pp. 198-217, (Jan 2008).
[15] http:// fimi.cs.helsinki.fi/, 2012.Frequent Itemset Mining Implementations Repository,