UP-Growth: An Efficient Algorithm for Mining High Utility Itemsets from Transactional Databases

Goura A Koti[1], Prof. J.Nagesh Babu[2]
M.Tech Student(CSE)[1], Assistant Professor[2]
Department of Computer Science and Engineering
Rao Bahadur Y. Mahabaleshwar Engineering College, Bellary

Abstract—**Utility mining is an emerging topic in data mining field. The main objective of Utility Mining is to identify the itemsets whose utility is highest than the user-specified threshold. Mining high utility itemsets from a transactional database refers to finding the itemsets that have utility above a user-specified threshold. Itemset Utility Mining which is an extension of frequent itemset mining identifies itemsets that occur frequently. An efficient mining of high utility itemsets plays an important role in many real-time applications such as business transaction, retail markets, supermarket, and medical applications and it is one of the important research issue in data mining area. Existing algorithms like Apriori, FP Growth incur the problem of producing a large number of candidate itemsets. Such a large number of candidate itemsets degrades the mining performance in terms of execution time and space requirement. The more the transaction length that the database contains may lead to worse case in achieving performance. In this paper, we propose an efficient algorithm, called utility pattern growth (UP-Growth) for mining high utility itemsets with a set of effective strategies for pruning candidate itemsets. A tree-based data structure named utility pattern tree (UP-Tree) is used to maintain the information of high utility itemsets. The performance of UP-Growth is compared with the IHUP algorithm. Experimental results show that the proposed algorithm will reduce the number of original database scans and generates less number of candidates which outperforms in terms of time and space.**

Index Terms—**Frequent Mining, UP-Tree, UP-Growth, Utility Mining.**

I. INTRODUCTION

Today, the information has become a superabundant in all sectors such as, from business transactions and scientific data, to satellite pictures, text reports and military intelligence. Though it brings huge new benefits but, it also creates big headache. So, it is very important to handle the huge data sets in an effective manner. Hence, **Data mining** is the way that helps in discovering relevant patterns from large data sets. In simple words, it is the process of analyzing data from different perspectives and summarizing it into useful information. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Technically, the goal is the extraction of patterns and knowledge from large amount of data, not the extraction of data itself.

Data mining is primarily used today by companies with a strong consumer focus - retail, real estate, financial transactions, banking, telecommunications and marketing organizations. It enables these companies to determine relationships among "internal" factors such as price, product positioning, or staff skills, and "external" factors such as economic indicators, competition, and customer demographics. And, it enables them to determine the impact on sales, customer satisfaction, and corporate profits. Finally, it enables them to "drill down" into summary information to view detail transactional data.

Discovering useful patterns hidden in a database plays an essential role in several data mining tasks, such as high utility pattern mining. Mining high utility itemsets from a transactional database refers to the discovery of itemsets with high utility like profits.

The following description brings the importance of utility mining which overcomes the drawbacks of frequent itemset and weighted association rule mining.

Frequent patterns [12] are itemsets, which appear in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a frequent itemsets. Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data. Frequent pattern mining helps in data indexing, classification, clustering, and other data mining tasks such as mining association rules [3]. The applications of this kind of mining are in the field of telecommunications, text analysis and census analysis. The interestingness metric used to express the frequency of itemsets is in terms of support value of the itemsets. The Support value of an itemset is the percentage of transactions that contain the itemset.
Problem: In the framework of frequent itemset mining, the importance of items to users is not considered.

Weighted association [8] rule mining approach is an extension of the traditional association rule mining [3] to which the weights are assigned to the items based on their significance. A weight of an item is a non negative real

number that shows the importance of each item. A pair (x, w) is called a weighted item where $x \in I$ is an item and $w \in W$ is the weight associated with x. A transaction is a set of weighted items, each of which may appear in multiple transactions with different weights.
Problem: Assigning weight for each item, based on its significance and generating the association rules for large itemsets that have above the user specified minimum weighted confidence.

The limitations of frequent or rare itemset mining motivated to develop a utility based mining [9] approach, which allows a user to conveniently express his or her perspectives concerning the usefulness of itemsets as utility values and then find itemsets with high utility values higher than a user-specified threshold. In utility based mining, the term utility refers to the quantitative representation of user preference i.e., the utility value of an itemset is the measurement of the importance of that itemset in the user's perspective. For e.g. if a sales analyst involved in some retail research needs to find out which itemsets in the stores earn the maximum sales revenue for the stores he or she will define the utility of any itemset as the monetary profit that the store earns by selling each unit of that itemset.

Formally an item set S is useful to a user if it satisfies a utility constraint i.e. any constraint in the form $u(S) >= minutil$, is useful to a user where u(S) is the utility value of the itemset and minutil is a utility threshold defined by the user. An itemset is called a high utility itemset if its utility is no less than a user specified threshold; otherwise, the itemset is called a low utility itemset.

Existing methods [3, 5, and 7] often generate a huge set of high utility itemsets and the mining performance is degraded consequently. The case when the database contains many long transactions will be the worst. The huge number of potential high utility itemsets forms a challenging problem to the mining performance since the higher processing cost is incurred with more number of potential high utility itemsets. Thus, the main aim of our proposed algorithm is to reduce the number of candidate itemsets.

To achieve this, we propose an efficient algorithm, called UP-Growth (Utility Pattern Growth) with a compact data structure called UP-Tree for discovering high utility itemsets.

II. RELATED WORK

A. *Definitions*

Here we are discussing some basic definitions about utility of an item, utility of itemset in transaction, utility of itemset in database, related works and the problem involved in the existing methods.

Given a finite set of items $I = \{ i_1, i_2, \dots i_m \}$ each item i_p $(1 \le p \le m)$ has a unit profit $p_r(i_p)$ An itemset X is a set of k distinct items $\{ i_1, i_2, \dots i_k\}$, where k is the length of X. An itemset with length k is called a k-itemset. A transaction database $D = \{T1, T_2,\dots, T_n \}$ contains a set of transactions, and each transaction T_d has a unique identifier d, called TID. Each item i_p in transaction T_d is associated with a quantity q (i_p, T_d) that is, the purchased quantity of i_p in T_d.

Definition 1:
Utility of an item i_p in a transaction T_d is the product of unit profit and the purchased quantity. It is denoted as $u\ (i_p, T_d)$ and defined as $pr\ (i_p, T_d) \times q\ (i_p, T_d)$

Definition 2:
Utility of an itemset X in T_d is the sum of utilities of the transaction containing X. It is denoted as $U(X, T_d)$ and defined as $\sum_{ip \epsilon X \wedge X \subseteq Td} u\ (i_p, T_d)$

Definition 3:
Utility of an itemset X in D is the sum of utilities of all the transactions containing X in the database D. It is denoted as u(X) and $\sum_{X \subseteq Td \wedge Td \epsilon D} u(X, Td)$.

Definition 4:
An itemset is called a high utility itemset if its utility is no less than a user-specified minimum utility threshold or else it is low-utility itemset.

Definition 5:
Transaction utility of a transaction T_d is the sum of utilities of all the items in the transaction T_d. It is denoted as $TU\ (T_d)$ and defined as $u(T_d, T_d)$

Definition 6:
Transaction-weighted utility of an itemset X is the sum of the transaction utilities of all the transactions containing X, which is denoted as TWU(X) and defined as $\sum_{X \subseteq Td \wedge Td \epsilon D} TU\ (T_d)$

Definition 7:
An itemset X is called a high-transaction weighted utility itemset (HTWUI) if TWU(X) is no less than minimum utility threshold.

B. *Literature Survey*

The paper [3] proposed Apriori algorithm, is used to obtain frequent itemsets from the database. This algorithm simply counts item occurrences to further determine the large one itemsets which requires scanning of database each time for each item. Next, the database scan is performed to count the support of candidate's itemsets. Then the association rules are generated from frequent itemsets. After identifying the large itemsets, only those itemsets which have the support greater than the minimum support are allowed.
Drawback: It generates a lot of candidate item sets and scans database every time and when a new transaction is added to the database then it should rescan the entire database again.

The paper [5] proposed an efficient FP-tree based mining method that builds frequent pattern tree (FP-tree) structure, for storing crucial information about frequent patterns into compressed structure. Pattern fragment growth mines the complete set of frequent patterns using the FP-growth. It constructs a highly compact FP-tree, which is usually substantially smaller than the original database, by which costly database scans are saved in the subsequent mining processes. It applies a pattern growth method which avoids costly candidate generation.
Drawback: FP-Growth Consumes more memory and performs badly with long pattern data sets. Thus it is not able to find high utility itemsets.

The paper [6] proposed Two-Phase algorithm to efficiently prune down the number of candidates and can precisely obtain the complete set of high utility itemsets. In Phase I, High transaction-weighted utilization itemsets (HTWUIs) are identified. The size of candidate set is reduced by only considering the supersets of high transaction-weighted utilization itemsets. In Phase II, one database scan is performed to filter out the high transaction-weighted utilization itemsets that are indeed low utility itemsets.
Drawback: It generates numerous candidates to obtain HTWUIs and requires multiple database scans.

Traditional methods of association rule mining consider the appearance of an item in a transaction, whether or not it is purchased, as a binary variable. However, customers may purchase more than one of the same item, and the unit cost may vary among items. Hence, developing an efficient algorithm is crucial for utility mining.

To overcome this problem, the paper [7] proposed isolated items discarding strategy (IIDS) to reduce the number of candidates. By pruning isolated items during level wise search, the number of candidate itemsets for HTWUIs in phase one can be reduced.
Drawback: This algorithm still scans database for several times and uses a candidate generation-and-test scheme to find high utility itemsets and thus cannot improved performance.

The paper [2] proposed a tree-based algorithm, named Incremental High Utility Pattern (IHUP). A tree based structure called IHUP-Tree is used to maintain the information about itemsets and their utilities.
Drawback: Though it achieves a better performance than IIDS and Two-Phase, it still produces several HTWUIs. Since the overestimated utility calculated by TWU is too large.

C. Problem Statement

In the literature we have studied the different methods proposed for high utility mining from large datasets. But all this methods frequently generate a huge set of PHUIs and their mining performance is degraded consequently. Further in case of long transactions in dataset or low thresholds are set, then this condition may become worst. The huge number of PHUIs forms a challenging problem to the mining performance since the more PHUIs the algorithm generates, the higher processing time it consumes. Thus to overcome this challenges the efficient algorithm is presented in this paper.

The main aim of this project is to achieve the following aspects:

- Reducing the number of scans in the original database.
- Minimize memory utilization (Reducing the search space).
- Reducing the total execution and computation time.
- Reducing the resource utilization.
- Increase the performance in terms of time and space complexity.

III. PROPOSED WORK

The proposed method consists of three main parts: 1) Construction of global UP-tree with two strategies (DGU and DGN) by scanning the database twice. 2) Generating the potential high utility itemsets using UP Growth with two strategies (DLU and DLN) 3) from the set of PHUIs identify actual high utility item set.

A. Construction of UP-Tree

To avoid repeated scanning of original database, the UP-tree data structure is maintained to keep track of the information of the transaction and high utility itemsets. In a UP-Tree, each node has its own detailed information such as node's item name, support count, parent name, node link to which it points to a node and a set of child nodes. The UP-Tree maintains a table named Header Table to facilitate the traversal of tree. In header table, each entry records an item name, an overestimated utility, and a link.

The construction of a global UP-Tree is performed with two database scans. In the first scan, each transaction's TU is computed; at the same time, each 1- item's TWU is also accumulated. Thus we can get promising items and unpromising items. The transactions are inserted into a UP-Tree in the second scan. When a transaction is retrieved, the unpromising items are removed from the transaction and their utilities are also eliminated from the transaction's TU. Thus, new TU are calculated after pruning unpromising items which are called reorganized transaction utility (abbreviated as RTU). Then, Reorganized Transactions will be constructed with the RTU.

Thus, **Strategy DGU (Discarding Global Unpromising Items during Constructing a Global UP-Tree)** states discard global unpromising items and their actual utilities from transactions and transaction utilities of the database.

Definition 8: An item whose TWU is less than minimum utility threshold is said to be unpromising item.
Definition 9: An unpromising item and all its supersets are not high utility itemsets. In other words, only the supersets of promising items are possible to be high utility itemsets.

Since unpromising items play no role in high utility itemsets, these items are removed. Thus, when utilities of itemsets are being estimated, utilities of unpromising items can be regarded as irrelevant and be discarded.

This makes to store less information in UP-Tree and also smaller overestimated utilities for item sets are generated. Strategy DGU uses RTU to overestimate the utilities of item sets instead of TWU. Since the utilities of unpromising items are excluded, RTU will be no larger than TWU. Therefore, the number of PHUIs with DGU will be no more than that of HTWUIs generated with TWU. DGU is quite effective especially when transactions contain lots of unpromising items.

It is possible to apply the divide-and-conquer technique in tree-based framework for high utility item set mining processes. Thus, the search space can be divided into smaller subspaces. The items that are descendant nodes of the item i_m will not appear in $\{i_m\}$-Tree; only the items that are ancestor nodes of i_m will appear in $\{i_m\}$-Tree. From this viewpoint, our second proposed strategy for decreasing overestimated utilities is to remove the utilities of descendant nodes from their node utilities in global UP-Tree.

Thus, **Strategy DGN (Decreasing Global Node Utilities during Constructing a Global UP-Tree)** states decrease the global node utilities for the nodes of global UP-Tree by actual utilities of descendant nodes. DGN is especially suitable for the databases containing lots of long transactions. In other words, the more items a transaction contains, the more utilities can be discarded by DGN.

By applying these two strategies, the overestimated utilities stored in the nodes of global UP-Tree are minimized.

B. UP-Growth

Once after the construction of a global UP-Tree, the next step is to mine UP-Tree for generating PHUIs. Thus, we propose an algorithm UP-Growth (Utility Pattern Growth) which has two strategies namely, Discarding Local Unpromising Items (DLU) and Decreasing Local Node Utilities (DLN).

The common method for generating patterns in tree based algorithms contains three steps: (1) Generate conditional pattern bases by tracing the paths in the original tree, (2) construct conditional trees (also called local trees in this paper) by the information in conditional pattern bases and (3) mine patterns from the conditional trees. For the two strategies, we maintain a minimum item utility table to keep minimum item utilities for all global promising items in the database.

First, the node links in UP-Tree corresponding to the item i_m, which is the bottom entry in header table, are traced. Found nodes are traced to root of the UP-Tree to get paths related to i_m. All retrieved paths, their path utilities and support counts are collected into i_m's conditional pattern base. A conditional UP-Tree can be constructed by two scans of a conditional pattern base. For the first scan, local promising and unpromising items are learned by summing the path utility for each item in the conditional pattern base. Then, DLU is applied to reduce overestimated utilities during the second scan of the conditional pattern base.

The **Strategy DLU (Discarding Local Unpromising Items during Constructing a Local UP-Tree)** states discard the local unpromising items and their estimated utilities from the paths and path utilities of conditional pattern bases

The paths are reorganized by pruning unpromising items by DLU and resorted by a fixed order. The paths are called reorganized paths. Then, the node utility is recalculated for the node N_{ik} in $\{i_m\}$-Tree. DLN is local version of DGN and is applied during inserting reorganized paths into a conditional UP-Tree.

The **Strategy DLN (Decreasing Local Node Utilities during Constructing a Local UP-Tree)** states decreasing Local Node utilities for the nodes of local UP-Tree by estimated utilities of descendant nodes.

By these strategies, overestimated utilities of item sets can be decreased and thus the number of PHUIs can be further reduced.

C. Efficiently identify High Utility Itemsets

The next step is to identify high utility itemsets and their utilities from the set of candidate itemsets by scanning original database once. But scanning original database is still time consuming since the original database is large and it contains lots of unpromising items. Thus, in the proposed work, high utility itemsets are identified by scanning reorganized transactions. Since there is no unpromising item in the reorganized transactions, the I/O cost and execution time for this process can be further reduced. This technique works well especially when the original database contains lots of unpromising items.

IV. Experimental Evaluation and Performance Study

This section describes the experimental environment and the performance of the proposed algorithm with different parameters compared to the IHUP algorithm. This algorithm is implemented in java language. The software tool used is NetBeans IDE 8.0.

A. Experimental Environment

In order to show the performance of proposed UP-Growth algorithm, we compare with the IHUP algorithm. Here we assume a simple transaction database with few items and transactions.

B. Results

For varying user-specified utility threshold, the graphs below shows the number of candidates generated, time taken for execution and the memory usage for existing and proposed algorithm. The varying minimum utility threshold set with the values 5, 10, 15,20,25,30 and 35.

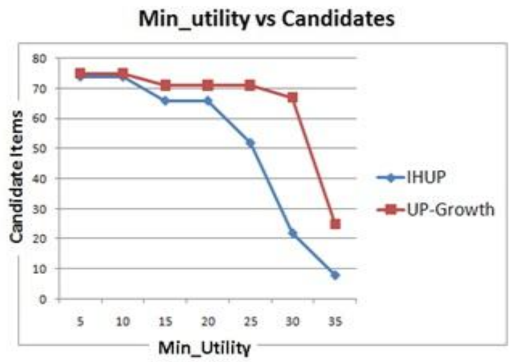

Figure 1: Minimum utility versus number of candidates

The figure 1 shows that the proposed UP-Growth algorithm generates less number of candidate items compared to IHUP algorithm due to pruning of itemsets with the efficient strategies.

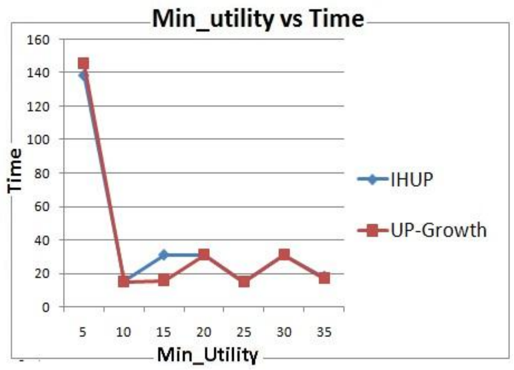

Figure 2: Minimum utility versus Execution Time

The figure 2 shows that the proposed UP-Growth algorithm consumes less time to find the high utility itemsets compared to IHUP algorithm due to less number of candidate items and scanning the reorganized transactions which has no unpromising items, instead of scanning the large database.

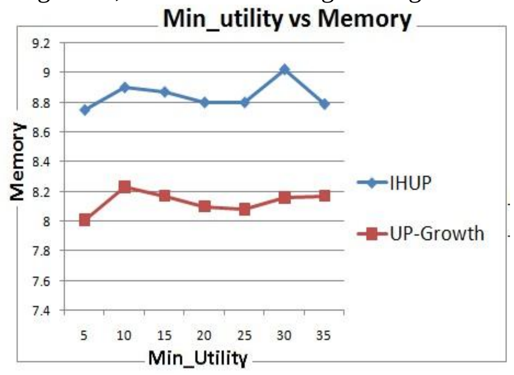

Figure 3: Minimum utility versus Memory Usage

The figure 3 shows that the proposed UP-Growth algorithm consumes less memory compared to IHUP algorithm due to less number of candidates generated and require no scanning of original database regularly.

V. CONCLUSION

The main problems with the existing methods are the generation a huge set of candidate items and scanning of the original database several times. Hence, the proposed algorithm ensures that it generates a few candidate items with only two scans. From the above experimental results, we can conclude that the proposed algorithm can efficiently find the high utility itemsets with the four strategies (DGU, DGN, DLU, and DLN) developed in this paper. Since the algorithm generates few candidate items, it takes less time to find the high utility itemsets. And also the memory consumed is less compared to the existing algorithms. Thus, pruning the itemsets very well at early stages saves the time as well as space.

VI. REFERENCE

[1] 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 Transaction on knowledge and data engineering, vol. 25, no. 8, Aug 2013.

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

[3] R. Agrawal , T. Imielinski, A. Swami, 1993, 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

[4] R. Agrawal, R Srikant, Fast algorithms for mining association rules,in : Proceedings of 20th international Conference on Very Large Databases ,Santiago, Chile, 1994, pp.487-499

[5] J Han, J.Pei, Y.Yin ,R. Mao Mining frequent Patterns without candidate generation:a frequent -pattern tree approach , Data Mining and Knowledge Discovery 8(1)(2004) 53-87

[6] Liu. Y, Liao. W,A. Choudhary, A fast high utility itemsets mining algorithm, in: Proceedings of the Utility-Based Data Mining Workshp, August 2005

[7] Y.-C. Li, J.-S. Yeh, and C.-C. Chang, "Isolated Items Discarding Strategy for Discovering High Utility Itemsets," Data and Knowledge Eng., vol. 64, no. 1, Jan. 2008.

[8] C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W. Kwong, "Mining Association Rules with Weighted Items," Proc. Int'l Database Eng. and Applications Symp. (IDEAS '98), 1998.

[9] R. Chan, Q. Yang, and Y. Shen, "Mining High Utility Itemsets," Proc. IEEE Third Int'l Conf. Data Mining, pp. 19-26, Nov. 2003.

[10] V.S. Tseng, 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), 2010.

[11] H. Yao, H.J. Hamilton, and L. Geng, "A Unified Framework for Utility-Based Measures for Mining Itemsets," Proc. ACM SIGKDD Second Workshop Utility-Based Data Mining, Aug. 2006.

[12] Jiawei Han, Hong Cheng, Dong Xin and Xifeng Yan, "Frequent pattern mining: current status and future directions," Data Mining Knowledge Discovery, January 2007