

UP-GROWTH++ ALGORITHM FOR MINING HIGH UTILITY ITEMSETS

V.T.Shenbagamuthu, D.Prabha,

Abstract— The efficient discovery of high utility itemset from large transaction database is a crucial task of data mining. In past, many relevant algorithms have been presented. These algorithms surface the problem of generating large number of candidate itemset and thus degrade the mining performance in terms of execution time and space. In this paper, three algorithms are presented, such as EUP-Growth (utility pattern) for mining high utility itemset for pruning candidate itemsets. In these algorithms, compact tree structure (EUP-Tree) is used for discovering the useful itemset so that candidate item is generated with only two scan of database. The performance of EUP-Growth and is compared with the state-of-the algorithm for both real and synthetic data sets. Experimental results shown that the proposed algorithm reduce the number of candidates effectively but also outperform other algorithms substantially in terms of runtime even when databases contain lots of long transactions

Index Terms— high utility itemset, frequent itemset, candidate pruning, utility mining,



1 INTRODUCTION

Data mining is one of the best analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorizes it, and summarizes it into useful information. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases. Data mining commonly encompasses a variety of algorithms namely clustering, classification, association rule mining, regression, summarization and prediction. Among these algorithms, Association rules mining (ARM) is one of the most widely used techniques in data mining and knowledge discovery and has tremendous applications in business, science and other domains. The main objective of ARM is to identify frequently occurring patterns of itemsets.

It first finds all the itemsets whose co-occurrence frequency are beyond a minimum support threshold, and then generates rules from the frequent itemsets based on a minimum confidence threshold. Traditional ARM model treat all the items in the database equally by only considering if an item is present in a transaction or not.

The frequent itemsets identified by ARM does not reflect the impact of any other factor except frequency of the presence or absence of an item. Frequent itemset mining is a fundamental research topic with wide data mining Applications. Extensive studies have been proposed for mining frequent itemsets from the databases and successfully adopted in various application domains. In market analysis, mining frequent itemsets from a transaction database refers to the discovery of the itemsets which frequently appear together in the transactions. However, the unit profits and purchased quantities of items are not considered in frequent itemset mining. To address this problem, weighted association rule mining was proposed. In this framework, weights of items, such as unit profits of items in transaction databases, are considered. In this concept, even if some items appear infrequently, they might still be found if they have high weights. However, in this framework, the quantities of items are not

- V.T.Shenbagamuthu is currently pursuing masters of engineering in computer science and engineering from Sri Krishna college of engineering and technology,Coimbatore , PH-9655753876. E-mail: shenbagamuthuvt@gmail.com
- D.Prabha is currently working as a assistant Professor in Sri Krishna college of engineering and technology, Coimbatore, PH-7373350567. E-mail: prabha@skcet.ac.in

considered yet. Therefore it **utility mining** [4, 5,6, 7,] emerges as an important topic in data mining for discovering the itemsets with high utility like profits. The basic meaning of utility is the interestedness/importance /profitability of items to the users. Mining high utility itemsets from databases is an important task which is essential to a wide range of applications such as website click streaming analysis, cross-marketing in retail stores, business promotion in chain hypermarkets and even biomedical applications.

In recent years ,there are many algorithms proposed for discovering the high utility itemset,but these algorithms produced large number of candidate itemset.so mining performance degraded such that in terms of execution time and memory consumption.

The utility of items in a transaction database consists of two aspects: (1) the importance of distinct items, which is called external utility, and (2) the importance of the items in the transaction, which is called internal utility. The utility of an itemset is defined as the external utility multiplied by the internal utility. 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*.

There is no efficient strategy to find all the high utility itemsets due to the non existence of “downward closure property” (anti-monotone property) in the *utility mining* model. In other words, pruning search space for high utility itemset mining is difficult because a superset of a low utility itemset may be a high utility itemset . This Limitations achieved by UP –Growth++ algorithms. In UP-Growth++, use compact tree structure called EUP -Tree .the information of high utility are storing into the EUP-Tree for discovering the utility itemset, such that candidate itemset generated with two scan of database. It reduce the execution time and memory consumption.

Many existing studies produce overestimated utilities to facilitate the mining performance of utility mining. To address this issue, propose a novel algorithm with a compact data structure for efficiently discovering high utility itemsets from transactional databases. The major steps of this work are summarized as follows:

1. A novel algorithm, called *UP-Growth++* (*Utility Pattern Growth*),is proposed for discovering high utility itemsets. Correspondingly, a compact tree structure, called *EUP-Tree* (*Enhanced Utility Pattern Tree*), is proposed to maintain the important information of the transaction database related to the utility patterns. High utility itemsets are then generated from the EUP-Tree efficiently with only two scans of the database.

2. Four strategies are proposed for efficient construction of EUP-Tree and the processing in UP-Growth++. By these strategies,the estimated utilities of candidates can be well reduced by discarding the utilities of the items which are impossible to be high utility or not involved in the search space.

3. Both of synthetic and real datasets are used in experimental evaluations to compare the performance of UP-Growth++ with the state-of-the-art utility mining algorithms. The experimental resultsshow that UP-Growth++ outperforms other algorithms substantially in terms of execution time, especially when the database contains lots of long ransactions.

TABLE 1

An Example Database

ID	TRANSACTION	TU
T1	(A,1)(C,1)(D,1)	17
T2	(A,2)(C,6)(E,2)(G,5)	27
T3	(A,1)(B,2)(C,1)(D,6)(E,1)(F,5)	37
T4	(B,4)(C,3)(D,3)(E,1)	30
T5	(B,2)(C,2)(E,1)(G,2)	13

TABLE 2

Profit table

ITEM	A	B	C	D	E	F	G
PROFIT	5	2	1	2	3	5	1

Min_util = 50

2 PROBLEM DEFINITIONS

Given a finite set of items $I = \{i_1, i_2, \dots, i_m\}$. Each item i_p ($1 \leq p \leq m$) has a unit profit $p(i_p)$. An itemset X is a set of k distinct items $\{i_1, i_2, \dots, i_k\}$, where $i_j \in I$, $1 \leq j \leq k$, and k is the length of X . An itemset with length k is called k -itemset. A transaction database $D = \{T_1, T_2, \dots, T_n\}$ contains a set of transactions, and each transaction T_d ($1 \leq d \leq n$) has a unique identifier d , called *TID*. Each item i_p in the transaction T_d is associated with a quantity $q(i_p, T_d)$, that is, the purchased number of i_p in T_d .

Definition 1. The utility of an item i_p in the transaction T_d is denoted as $u(i_p, T_d)$ and defined as $p(i_p) \times q(i_p, T_d)$. For example, in Table 1, $u(\{A\}, T1) = 5 \times 1 = 5$.

Definition 2. The utility of an itemset X in T_d is denoted as $u(X, T_d)$ and defined as $\sum_{i_p \in X} u(i_p, T_d)$. For example, $u(\{AC\}, T1) = u(\{A\}, T1) + u(\{C\}, T1) = 5 + 1 = 6$.

Definition 3. An itemset is called a *high utility itemset* if its utility is no less than a user-specified *minimum utility threshold* which is denoted as *min_util*. Otherwise, it is called a *low utility itemset*.

Given a transaction database D and a user-specified minimum utility threshold *min_util*, mining high utility itemsets from the transaction database is equivalent to discover from D all itemsets whose utilities are no less than *min_util*. After addressing the problem definition of utility mining, introduce the transaction-weighted downward closure.

Definition 5. The transaction utility of a transaction T_d is denoted as $TU(T_d)$ and defined as $u(T_d, T_d)$. For example, $TU(T1) = u(\{ACD\}, T1) = 8$.

Definition 6. The transaction-weighted utilization of an itemset X is the sum of the transaction utilities of all the transactions containing X , which is denoted as $TWU(X)$. For example, $TWU(\{AD\}) = TU(T1) + TU(T3) = 8 + 30 = 38$. If $TWU(X)$ is no less than the minimum utility threshold, X is called a *high transaction-weighted utilization itemset* (abbreviated as *HTWUI*).

Definition 7. The *transaction-weighted downward closure*, which is abbreviated as *TWDC*, is stated as follows. For any itemset X , if X is not

a *HTWUI*, any superset of X is a low utility itemset. By this definition, the *downward closure property* can be maintained by using transaction-weighted utilization. For example, in Table 1, any superset of $\{AD\}$ is a low utility itemset since $TWU(\{AD\}) < \text{min_util}$

3 EXISTING SYSTEM

In the past ten years, a number of traditional ARM algorithms have been proposed. The general assumption of them is that each item in a database is treated equally. All of these algorithms exploit the "downward closure property" as proposed in Apriori [1]. The tree-based approaches such as FP-Growth [2] were proposed. It's widely recognized that FP-Growth achieves a better performance than Apriori-based approaches since it finds frequent itemsets without generating any candidate itemset and it scans database just twice. However, in the framework of frequent itemset mining the importance of items to users is not considered. The unit profits and purchased quantities of the items are not taken into considerations. An efficient association rules generation method, WFIM [3] approach is to push the weight constraints into the pattern growth algorithm while maintaining the downward closure property. In this paper, a weight range and a minimum weight constraint are defined and items are given different weights within the weight range. The weight and support of each item are considered separately for pruning the search space. But quantity not considered, so this paper can not discover the frequent itemset with high sale profit.

Thus, some methods were proposed for mining high utility itemsets from the databases, such as Two-Phase [4], IIDS [5] and IHUP [6], UP growth [7] used to prune search space. Although it is shown to have good performance, it cannot capture the complete set of high utility itemsets since some high utility patterns may be pruned during the process. Two-Phase algorithm [4] proposed by Liu et al. consists of two phases. In phase I, Two-Phase algorithm propose a breadth first search strategy to generate HTWUIs. It generates candidate itemsets of length k from HTWUIs of length $(k-1)$ and prunes candidate itemsets by TWDC property. In phase II, high utility itemsets and their utilities are identified

from the HTWUIs by scanning original database once.

Although Two-Phase algorithm effectively reduces the search space by TWDC property and captures the complete set of high utility itemsets, it still generates too many candidates for HTWUIs and requires multiple database scans in phase II. To overcome this problem, Li et al. [5] proposed an isolated items discarding strategy, abbreviated as IIDS, to reduce the number of candidates. By pruning isolated items during the level-wise search, the number of candidate itemsets for HTWUIs in phase I can be reduced effectively. However, this approach still scans database multiple times and uses a candidate generation-and-test scheme to find high utility itemsets.

Ahmed et al. [6] proposed a tree-based algorithm, called IHUP, for mining high utility itemsets. They use an IHUP-Tree to maintain the information of high utility itemsets and transactions. Every node in IHUP-Tree consists of an item name, a support count, and a TWU value. The HTWUIs are generated from the IHUP-Tree by applying the FP-Growth algorithm [2]. Thus, HTWUIs in phase I can be found more efficiently without generating candidates for HTWUIs. In step 3, high utility itemsets and their utilities are identified from the set of HTWUIs by scanning the original database once. Although IHUP finds HTWUIs without generating any candidates for HTWUIs and achieves a better performance than IIDS and Two-Phase, it still produces too many HTWUIs in phase I. Note that IHUP and Two-Phase produce the same number of HTWUIs in phase I since they use transaction-weighted utilization mining model to overestimate the utilities of the itemsets. However, this model may overestimate too many low utility itemsets as HTWUIs and produce too many candidate itemsets in phase I. Such a large number of HTWUIs degrades the mining performance in phase I in terms of execution time and memory consumption.

4 PROPOSED SYSTEM

The framework of the proposed methods consists of three steps: 1) Scan the database twice to construct a global EUP Tree with the first two strategies .2) recursively generate PHUIs from global EUP-Tree and local EUP-Trees by UP-Growth++ with the third and fourth Strategies. 3) identify actual high utility

itemsets from the set of Note that we use a new term “potential high utility itemsets” to distinguish the patterns found by our methods from HTWUIs since our methods are not based on traditional TWU model. By our effective strategies, the set of PHUIs will become much smaller than the set of HTWUIs.

STRDEGY 1: Removing Global Unpromising Items during Constructing a Global EUP-Tree

The construction of a global EUP-Tree can be performed with two scans of the original database. The global EUP-Tree is depicted in Figure 1 .In the first scan, TU of each transaction is computed. At the same time, TWU of each single item is also accumulated. Allocate the memory for TWU using Random Hashing memory. Allocate the memory space for the 1st item based on the hash function

$$h(k) = [((a \cdot k) + b) \bmod s] \bmod n$$

By TWDC property, an item and its supersets are unpromising to be high utility item sets if its TWU is less than the minimum utility threshold. Such an item is called an unpromising item. An item i_p is called a promising item if $TWU(i_p) \geq \min_util$. Otherwise it is called an unpromising item. The unpromising items removed from original database. New TU after pruning unpromising items is called reorganized transaction utility (abbreviated as RTU). RTU of a reorganized transaction T_r is denoted as $RTU(T_r)$.

STRADEGY 2: Reducing Global Node Utilities during Constructing a Global EUP-Tree

It is shown that the tree-based framework for high utility item set mining applies the divide-and-conquer technique in mining processes. Thus, the search space can be divided into smaller subspaces. By applying strategy RGN, the utilities of the nodes that are closer to the root of a global EUP-Tree are further reduced. RGN is especially suitable for, the more items a transaction contains, the more utilities can be discarded by RGN. In following subsections, we describe the process of constructing a global EUP-Tree(Figure) with strategies RGU and RGN. By the strategies, overestimated utilities of item sets can be decreased and thus the number of PHUIs can be further reduced.

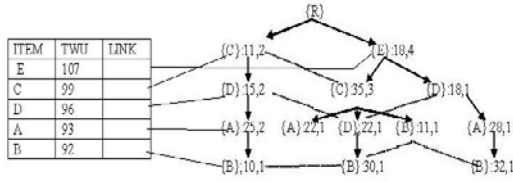


Figure 1

STRATEGY 3: Removing Local Unpromising Items during Constructing a Local UP-Tree

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 by the information in conditional pattern bases and (3) mine patterns from the conditional trees. However, strategies RGU and RGN cannot be applied into conditional EUP-Trees since actual utilities of items in different transactions are not maintained in a global EUP-Tree. We cannot know actual utilities of unpromising items that need to be discarded in conditional pattern bases unless an additional database scan is performed. To overcome this problem, a naïve solution is to maintain items' actual utilities in each transaction into each node of global EUP-Tree. However, this is impractical since it needs lots of memory space. In view of this, we propose two strategies, named RLU and RLN, that are applied in the first two mining steps and introduced in this and next subsections, respectively.

STRATEGY 4: Reducing Local Node Utilities (RLN) during Constructing a Local EUP-Tree

Since $\{i_m\}$ -Tree must not contain the information about the items below i_m in the original EUP-Tree, we can discard the utilities of descendant nodes related to i_m in the original EUP-Tree while building $\{i_m\}$ -Tree. Because we cannot know actual utilities of the descendant nodes, we use minimum item utilities to estimate the discarded utilities. Path utility of item i_k in $\{i_m\}$ -CPB is denoted as $pu(i_k, \{i_m\}$ -CPB) and defined as the following Equation 1:

$$pu(p, \{i_m\} - CPB) = \sum_{VIEW(\{i_m\} - CPB)^{iEP}} min(i) * p.count \rightarrow 1$$

A conditional EUP-Tree can be constructed by two scans of a conditional pattern base. It is depicted in figure2. 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, RLU is applied to reduce overestimated utilities during the second scan of the conditional pattern base. When a path is retrieved, unpromising items and their estimated utilities are eliminated from the path and its path utility. Then the path is reorganized by the descending order of path utility of the items in the conditional pattern base.

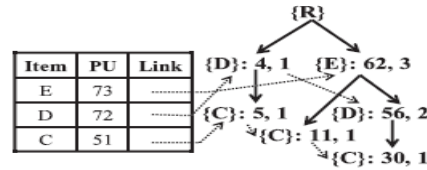


Figure 2(B-Tree with different strategies)

The complete set of PHUIs is generated by recursively calling the procedure named EUP-Growth. Initially, UP-Growth++ (TR, HR, null) is called, where TR is the global EUP-Tree and HR is the global header table. The procedure of UP-Growth++ is shown in algorithm 1 and 2:

Subroutine: *UP-Growth++*(T_x, H_x, X)
Input: A EUP-Tree T_x , a header table H_x for T_x , an itemset X , and a minimum utility threshold min_util .
Output: All PHUIs in T_x .
 (1) For each entry i_k in H_x do
 (2) Trace each node related to i_k via $i_k.hlink$ and accumulate $i_k.nu$ to $nu_sum(i_k)$;
 /* $nu_sum(i_k)$: the sum of node utilities of i_k */
 Allocate the memory for $nu_sum(i_k)$ using RH .
 (3) If $nu_sum(i_k) \geq min_util$, do
 (4) Generate a PHUI $v = X \cup i_k$;
 (5) Set $pu(i_k)$ as estimated utility of v ;
 (6) Construct v -CPB;
 (7) Put local promising items in v -CPB into H_v
 (8) Apply DLU to reduce path utilities of the paths;
 (9) Apply *Insert_Reorganized_path* to insert paths into T_v with DLN;
 (10) If $T_v \neq null$ then call *UP-Growth++*(T_v, H_v, v);
 (11) End if

(12) End for

Algorithm: 1

Subroutine:Insert_Reorganized_Path(N_{ix})

Line 1:If N has a child N_{iy},such that N_{ix.item}=i_x,increment N_{ix.count} by P_{j.count}.otherwise,create a new child node N_{ix} with N_{ix.item}=i_x, N_{ix.count}=P_{j.count},N_{ix.parent}=N and N_{ix.nu}=0.

Line 2:Increase N_{ix.nu} by Eq(3).

Line 3:If there exists a node N_{ix} in p_j where x+1<m', call Insert_Reorganized_path(N_{ix,i_x+1})

Algorithm 2

5 EXPERIMENTAL EVALUATION

Performance of the proposed algorithms is evaluated in this section. The experiments were performed on a 2.80 GHz Intel Pentium D Processor with 3.5 GB memory. The operating system is Microsoft Windows 7. The algorithms are implemented in Java language. Both real and synthetic data sets are used in the experiments. Synthetic data sets were generated from the data generator Parameter descriptions and default values of synthetic data sets are shown in Table 3. Real world data sets Accidents and chess are obtained from FIMI Repository Foodmart is acquired from Microsoft foodmart 2000 database. In the above data sets, except Chain-store and Foodmart, unit profits for items in utility tables are generated between 1 and 1,000 by using a log-normal distribution and quantities of items are generated randomly between 1 and 10. The two real data sets Chain-store and Foodmart already contain unit profits and purchased quantities.

TABLE 3
 PARAMETER SETTING DIFFERENT DATASETS

PARAMETER DESCRIPTIONS	DEFAULT
D Total number of transaction	100K
T Average transaction length	10
I Number of distinct items	1000
Q Maximum number of purchased items in transactions	10

6 PERFORMANCE COMPARISONS UNDER DIFFERENT DATASETS

In this part, we show the performance comparison on three real data sets: dense data set Web log and sparse data sets Chain-store. First, we show the results on sparse data set Chain store in Fig. 3 and 3a. In this graph we can observe that the performance of proposed methods substantially outperforms that of previous methods. The runtime of EUPT&UPG++ is the best.

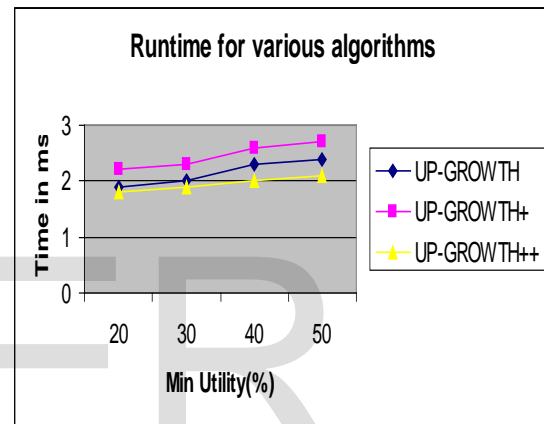


Figure 3

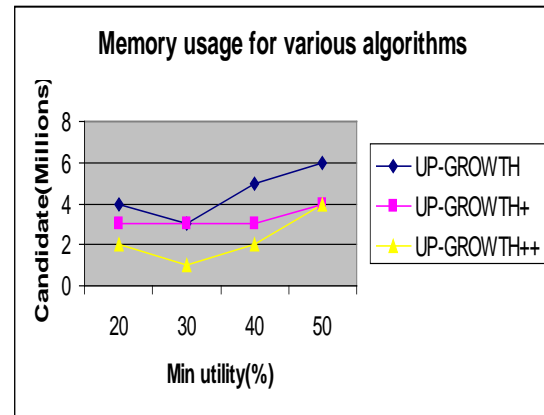


Figure 3a

Experimental results on real sparse data sets are shown in Fig. 10. The performance on dense dataset Web log shown in Fig4 and 4a

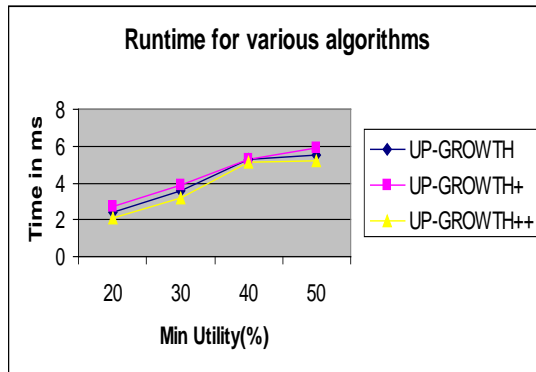
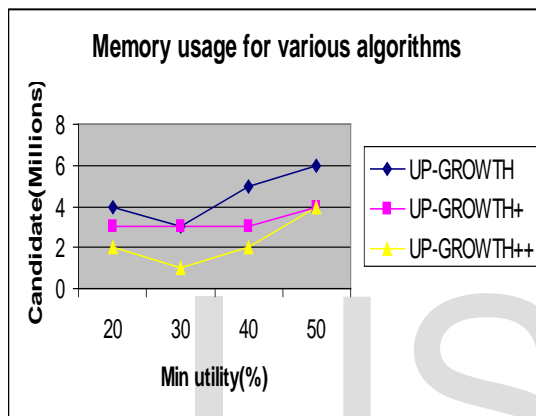


Figure 4a



By the above results and discussions, we can realize that Enhanced UP-Tree, Enhanced UP-Growth, efficiently decrease the number of candidates and make the performance much better than that of the state-of-the-art previous utility mining algorithms .

7 CONCLUSIONS

In this paper, proposed efficient algorithm named UP-Growth++ for mining high utility itemsets from transaction databases. A data structure named EUP-Tree was proposed for maintaining the information of high utility itemsets. PHUIs can be efficiently generated from EUP-Tree with only two database scans. Moreover, developed several strategies to decrease overestimated utility and enhance the performance of utility mining. In the experiments, both real and synthetic data sets were used to perform a thorough performance evaluation. Results show that the strategies considerably improved performance by reducing both the search space and the number of candidates.

8 REFERENCES

1. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases (VLDB), pp. 487-499, 1994.
2. Han,J.,Pei., &Yin,Y.(2000) " Mining frequent patterns without candidate generation" in proceeding of the 2007 international conference on management of data,(pp.1-12).
3. U. Yun and J.J. Leggett, "WFIM: Weighted Frequent Itemset Mining with a Weight Range and a Minimum Weight," Proc. SIAM Int'l Conf. Data Mining (SDM '05), pp. 636-640, 2005.
4. Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining algorithm," Proc. Utility-Based Data Mining Workshop,2005.
5. Y.-C. Li, J.-S. Yeh, and C.-C. Chang, "Isolated Items discardingStrategy for Hig Utility Itemsets," Data and Knowledge Eng., vol. 64, no. 1, pp. 198-217, Jan. 2008
6. 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
7. Vincent S.Tseng,Bai-En Shil,Cheng-Wei Wu " Efficient Algorithms for mining High Utility Itemsets from Transactional Databases.
8. Frequent Itemset Mining Implementations Repository, <http://fimi.cs.helsinki.fi/>, 2012