International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 9, Issue 5 (December 2013), PP. 38-50

EUP-Growth$^+$ - Efficient Algorithm for Mining High Utility Itemset

V.JayasudhaV.Umarani MPhil, (Phd)
Research ScholarAssistant Professor
Department of Computer ScienceDepartment of Computer Science
Sri Ramakrishna College of Arts and Science Sri Ramakrishna College of Arts and Science
For Women Coimbatore, Indiafor Women Coimbatore, India

Abstract:- In recent years, Utility mining becomes an emerging topicin the field of data mining. From a transaction database the discovery of itemsets with high utility like profits are referred as a high utility itemsets mining. In this paper, a new algorithm is proposed,named Enhanced Utility Pattern Growth$^+$ (EUP-Growth+), for reducing a large number of candidate itemsets for high utility itemsetswith a set of effective strategies. These strategies are used for pruning candidate itemsets effectively. By reducing a hefty number of candidate itemsets the mining performance upgrades in terms of execution time and space requirement. The selective information ofpotential high utility itemsetsare stored in the appropriate memory using a hashing technique and maintained in a tree-based data structure named Improved Utility Pattern Tree (IMUP-Tree). The performance of EUP-Growth+ is compared with the State-of-the-art algorithms on many types of both real and synthetic data sets. Experimental and comparative results reveal that the proposed algorithms, EUP- Growth+, not only reduce the number of PHUIs effectivelybut also outperform other algorithms.

Index Terms:- Candidate pruning, utility mining, frequent itemset, potential high utility itemset,

I. INTRODUCTION

Data mining is the process of enlightening non-trivial, formerly unknown and potentially useful information from large databases. Association Rule Mining is one of the popular and well research technique in data mining for finding interesting pattern between variables in a large database. The most well-known example for association rule mining is Market basket analysis. In ARM, the most important pattern mining is frequent pattern mining which is a fundamental research topic that has been applied to different kinds of databases, such as transnationaldatabases [1], [14], [21], streaming databases [18], [27], and time series databases [9], [12], and various application domains, such as Bioinformatics [8], [11], [20], Web click-stream analysis [7], [35], and mobile environments [15], [36]. The frequent itemsets identified by ARM reflect only the frequency of the existence or nonexistence of an item. Hence, the major drawbacks of frequent pattern mining are; first the impact of any other factor are not consider in frequent pattern mining. Next the non-frequent itemsets may contribute a large portion of the profit. Finallythe relative importance of each item is not considered in frequent pattern mining. Recently, to address the limitation of ARM, many types of association rule mining were defined as weightedfrequent pattern mining and Utility Mining.

In weighted frequent pattern mining, weights of items such as price and profits are considered in the transaction database.With this perception, even if some items come out infrequently, they might still be found if they have higher weights. However, in this weighted pattern mining, the quantities of items are not considered yet. Therefore, it cannot satisfy the user requirements who are interested in discovering the itemsets with high sales profits.

To overcome this, Utility mining becomes an emerging topicin the field of data mining. In a transaction database the discovery of itemsets with high utility like profits are referred as a high utility itemsets mining. In a transaction database, the utility of an item consists of two aspects: 1) External utility and 2) Internal utility. External utility provides importance's to distinct items. Internal utility provides importance'sto item in transactions. The utility of an itemset is defined as the product of external utility and internal utility. The utility of an itemset which is greater than a user-specified minimum utility threshold is called a high utility itemset; otherwise, it is called a low-utility itemset.

However, mining high utility itemsets from databases is not an easy task since downward closure property [1] in frequent itemset mining does not hold. In other words, pruning the search space for high utility itemset mining is difficult because a superset of a low-utility itemset may be a high utility itemset. A simple method to address this problem is to compute all itemsets from databases by the principle of fatigue. Obviously, this method requires large search space, particularly when databases restrain lots of long transactions or a lower

minimum utility threshold is set. Hence, In order to reduce the search space and efficiently capture all high utility itemsets with no mess is a crucial challenge in utility mining.

In existing studies, the performance of utility mining are improved by applying the overestimated methods [3], [10], [16], [17], [19], [24], [29], [30]. In these methods, potential high utility itemsets (PHUIs) are identified first, and then one more additional database scan is performed for identifying their utilities. However, these methods generate a hefty number of potential high utility itemsets and their mining performance is degraded subsequently. This situation may become worse when databases contain many long transactions or low thresholds are set. The hefty number of PHUIs forms a challenging problem for the mining performance since the algorithm generates a large number of PHUIs and also it requires high processing time it consumes.

In this paper, the existing UP-Growth$^+$ algorithm is enhanced to generate high utility itemsets efficiently for large datasets and reduce execution time when compared with existing algorithms. In the experimental section, experiments are conducted on our enhanced algorithm and existing algorithm with a variety of synthetic and real-time datasets.

The rest of this paper is organized as follows: In Section 2, the background and related work for high utility itemset mining are discussed. In Section 3, the proposed data structure and algorithms are described in details. In section 4, the experimental results are shown and conclusions are given in Section 5.

II. BACKGROUNDS

In this section, we first define the preliminary work of utility mining, and then introduce related work in utility mining.

2.1 Preliminary Work

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 $pr\ (i_p)$. An itemset X is a set of k distinct items $\{i_1,i_2, \dots i_m\}$, where$i_p \in I, 1 \le p \le m$. K is the length of X. An itemset with length k is called a k- itemset. A transaction database $D = \{T_1,T_2, \dots T_n\}$Contains a set of transactions, and each transaction $T_d (1 \le d \le n)$ 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: Utility of an item i_p in a transaction T_d is denoted as $u\ (i_p,T_d)$ and defined as $pr\ (i_p) \times q\ (i_p, T_d)$.

Definition: Utility of an itemset X in T_d is denoted as $u\ (X, T_d)$ and defined as $\sum_{i_p \in X \wedge X \subseteq T_d} u\ (i_p,T_d)$.

Definition: Utility of an itemset X in D is denoted as u (X) and defined $\sum_{X \subseteq T_d \wedge T_d \in D} u\ (X, T_d)$

Definition: 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. **Definition:** Transaction utility of a transaction T_d is denoted as $TU\ (T_d)$ and defined as $u\ (T_d,T_d)$.

Definition: 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 $\sum_{X \subseteq T_d \wedge T_d \in D} TU(T_d)$.

Definition: An itemset X is called a high transaction- weighted utility itemset (abbreviated as HTWUI) if TWU(X) is no less than min_util.

2.2 Related Work

Extensive studies have been proposed for mining frequent patterns [1], [2], [13], [14], [21], [22], [34], and [40]. Among this, Apriori [1] is the first association rule mining algorithm that pioneered the use of support based pruning to systematically control the exponential growth of candidate itemsets. Apriori algorithm faces two problems dealing with large datasets; first it requires multiple scans of transaction database, incurring a major time cost. Second it generates too many candidate sets which take a lot of memory space. All of the Apriori-based mining algorithms [1],[2],[3] have time and space cost problems when handling a huge number of candidate sets and a large database.

Numerous pattern growth-based association rule mining algorithms are available in the literature [14], [21]. FP-Growth [14] is widely recognized. It achieves a better performance than Apriori-based algorithms since it finds frequent itemsets without generating any candidate itemset and scans database just twice. In the framework of frequent itemset mining, the importance of items is not considered and also it does not satisfies user requirement.

To overcome this, the topic called weighted association rule mining was brought to the attention [4], [26], [28], [31], [37], [38], [39]. CAI et al. First proposed in the concept of weighted items and weighted association rules [4]. However, since the framework of weighted association rules does not have downward closure property, mining performance is degrades subsequently. To address this problem, the perception of weighted downward closure property [28], use transaction weight and weighted support that can not only reflect the importance of an itemset but also maintain the downward closure property during the mining process. The goal is to steer the mining focus to those significant relationships involving items with significant weights rather than being flooded in the combinatorial explosion of insignificant relationships. Although weighted association rule mining considers the importance of items but still items, quantities in transactions database are not taken

into considerations. Thus, the problem of high utility itemset mining is elevated and many studies[3], [5], [10], [16], [17], [19], [24], [25], [29], [30], [32], [33]have elucidated this issues. Recent research has focused on efficient high utility mining using intermediate anti-monotone measures for pruning the search space. Liu et al [19] proposed a two phase algorithm to mine high utility itemsets. In phase I, it employs an Apriori-based level-wise method to enumerate HTWUIs. Candidate itemsets with length k are generated from length k-1 HTWUIs and once in each pass scans the database to compute theirTWUs. After the above steps, the whole set of HTWUIs is collected in phase I. In phase II,high utility itemsets (HTWUI) are identified with an additional database scan. Although Two-Phase algorithm reduces search space by using TWDC property, nonetheless it generates too many candidates to obtain HTWUIs and requires multiple database scans. A framework for high utility itemset mining was proposed recently by Yao et al [16, It is a mining method for describing pruning strategies based on the mathematical properties of utility constraints. It developed an algorithm named Umining and other heuristic based algorithm Umining_H to discover high utility itemsets. However, this algorithm is based on the mathematical approach and it suffers from poor performance when mining dense datasets and long patterns much like the Apriori algorithm for frequent pattern mining.

An isolated item discarding strategy (abbreviated as IIDS) was proposed by Li et al. [17] to reduce the number of candidates. During level-wise search the isolated items are pruned and reduces the number of candidateitemsets. However, this algorithm still scans database for several times and uses a candidate generation-and-test scheme to find high utility itemset which increases time complexity.

The two novel algorithms, named utility pattern growth (UP- Growth) and UP-Growth +, and a compact tree structure, called a utility pattern tree (UP-Tree), for discovering high utility itemsets and maintaining important information related to utility patterns within databases were proposed by Tseng et al., [30], [31]. Several strategies are proposed for facilitating the mining processes of UP-Growth and UP-Growth+ by maintaining only essential information onthe UP - Tree. By these strategies, overestimated utilities of candidates can be well reduced by discarding utilities of the items that cannot be high utilized or are not involved in the search space. The proposed strategies decrease the overestimated utilities of PHUIs and also greatly reduces the number of candidates. Although the UP-Growth+ algorithm reduces the number of potential high utility itemsets for large datasets. But it takes more execution time and I/O Operation and also it contains overrated utility itemset due to random memory allocation in the Up - tree.

As stated above, the number of generating PHUIs is a critical issue for the performance of algorithms. The random memory allocation is a precarious issue for the mining speed of the up-tree. Therefore, this study aims at proposing several strategies for reducing memory, I/O operations, PHUIs, and overestimated utilities. By applying the proposed strategies, the number of generated PHUIs can be highly reduced and high utility itemsets can be identified more efficiently. Finally reduces memory in the UP - tree.

III. PROPOSED METHODS

The structure of the proposed methods consists of two steps: In first step it requires two database scan for constructing a global IMUP- Tree with the first two strategies (given in Section 3.1). In second step,PHUIs are generated recursively from global IMUP-Tree and local IMUP-Trees by EUP-Growth+ with the third and fourth strategies (given in Section 3.2).

3.1 The Proposed Data Structure: IMUP-Tree

To alleviate the mining speed, merge the UP-tree [30], [31] with one of the hashing technique for reducing the memory consumption. The improved UP-tree named as an IMUP - Tree is used to store the information about transactions and high utility itemsetsin the appropriate memory and maintained as tree structure. Two strategies with RH algorithm are applied for reducing the memory and to store the overestimated utility of each item during the construction of a global IMUP-Tree.The two strategies and the construction of global IMUP-tree with the two strategies are briefly discussed in the following sections.

3.1.1 DGUM: Discarding Global Unpromising Items and Memory Allocation during Constructing a Global IMUP-Tree

The construction of a global IMUP-Tree is accomplished with two scans of the original database. In the first scan, Transaction Utility of each transaction is computed. At the same time, Transaction Weighted Utility of each single item is also accrued. By TWDC property, an unpromising item is defined as an item and its supersets are unpromising to be high utility itemsets if its TWU is less than the minimum utility threshold.

During the second scan of the database, transactions are inserted into a global IMUP-Tree. When a transaction is retrieved, the unpromising items are removed from the transaction and their utilities should also be eliminated from the transaction's TU. This concept forms our first strategy.

Strategy 1. DGUM: Discarding global unpromising items and their actual utilities from transactions and transaction utilities of the database.

According to DGUM,while utilities of itemsets are being estimated, utilities of unpromising items can be pragmatic as irrelevant and be discarded. From this,we can realize that unpromising items play no role in high utility itemsets.

The new TU after pruning unpromising items is called reorganized transaction utility (abbreviated as RTU). The remaining promising items in the transaction are sorted in the descending order of TWU. Allocate memory for each promising items in the TWU using RH algorithm are described in the section 3.1.2. Moreover, before constructing an IMUP-Tree, DGUM can be performed repeatedly till all the promising items are allocated to the appropriate memory space. Transactions are inserted into allocated memory of the IMUP - tree which are generated by RH algorithm.

3.1.2Random Hashing

A random hashing algorithm which is a hash-based technique mines the potential high utility itemsets without any collision into the memory. It is a very efficient method in searching for the exact data item set in a very short time. The following are the steps which have been performed using RH algorithm.

The process of RH algorithm is to place each and every item in the memory location for the purpose of ease of usability. The basic things required for hashing process is hash functions. The hash function provides a way of assigning numbers to the input item such that the item can then be stored in the memory corresponding to the assigned number. Random hashing efficiently allocates the memory for the itemsets into the IMUP-tree. The potential high utility itemsets are mined exactly by means of penetrating the IMUP-tree.

Table 3 Pseudo Code for Random Algorithm

1. Generate descending order of TWU for the promising items.
2. Allocate the memory space for tree based on number of items in the database.
n = (Number of items) + x
Where 'x' may be any integer and
The memory size (n) must be nearest
Prime number to the total number of itemsin the database.
3. Allocate the memory space for the 1st item based on the hash function
h (k) = [(((a . k) + b) mod s) mod n]
Where, s is the total number of transactions in the table.
(a ,b) is any random number between the number of Items in transaction.
k is the items in the TWU
4. The above step is repeated until the memoryis allocated for all the items in TWU.
5.If collision occurs, change the random number inthe hash function to allocate the memory forcollided item.

Advantage:

The followings are the advantages of random hashing,

- The time and space complexity of the mining process are gradually reduced.
- It increases the mining process speed.

3.1.3 Strategy DGNM: Decreasing Global Node Utilities and Memory Allocation during Constructing a Global UP-Tree

It is shown in [3] that the tree-based framework for high utility itemset mining applies the divide-and-conquer technique in mining methods. Thus, the search space can be alienated into smaller subspaces. From this viewpoint, our second proposed strategy for decreasing overestimated utilities is to remove the utilities of descendant nodes from their node utilities in a global IMUP - Tree. The process is performed during the construction of the global IMUP-Tree. By applying strategy DGNM, the utilities of the nodes are inserted into the appropriate memory which reduces memory and also reduces utilities of the nodes that are closer to the root of a global UP-Tree. DGNM is especially suitable for the databases containing lots of long transactions. Inthe following sections, the process of constructing a global UP-Tree with strategies DGUM and DGNM are described.

3.1.4 Constructing a Global UP-Tree by Applying DGUM and DGNM

Recall that the construction of a global IMUP-Tree is performed with two database scans. In the first scan, Transaction Utility is computed; at the same time, each 1- item's TWU is also accrued.

Thus, we can get promising items and unpromising items.DGUM is applied to promising items by pruning the unpromising items and sorting the remaining promising items in a fixed order. The order can be used such as the lexicographic, support, or TWU. Each transaction after the above reorganization is called a reorganized transaction. The following paragraphs, we use the TWU descending order to explain the whole process since it is mentioned that the performance of this order is the best in previous studies [3]. The remaining promising items in the transaction are sorted in the descending order of TWU. Allocate memory for each promising items in the TWU using RH algorithm shown in the table 3. A function Insert Reorganized Transaction is called to apply DGNM during constructing a global IMUP-Tree. When a reorganized transaction $t_j{}' = \{i_1, i_2, i_3 \ldots i_n\}$ $(i_k \in I, 1 \leq K \leq n)$ is inserted into a global IMUP-Tree, Insert_ Reorganized_ Transaction (N,i_x) Is called, where N is a node in an IMUP - Tree and i_x is an item in$t_j{}'(i_x \in t_j{}', 1 \leq x \leq n)$. First, ($N_R$,$i_1$) is taken as input, where N_R is the root node of IMUP-Tree.

Table 4 The subroutine of Insert Reorganized Transaction

Subroutine: Insert _ Reorganized _ Transaction (N,i_x)
Line 1: If N has a child N_{i_x} such that N_{i_x}.item =i_x, increment N_{i_x}.count by 1. Otherwise, create a new child node N_{i_x} with N_{i_x}.item = $i_x$$N_{i_x}$.count = 1, N_{i_x}.parent = N N_{i_x}.nu = 0.
Line 2: Increase N_{i_x}.nu by (RTU (t_j ') - $\sum_{p=x+1}^{n} u(i_p, t_j{}')$), where $i_p \in t_j{}'$.
Line 3: If x ≠ n, call Insert_ Reorganized _ Transaction (N_{i_x},i_{x+1})

The node fori_1 ,N_{i_1} is found or created under $N_R$and its support is updated in Line 1.Then DGNM is applied in Line 2 by discarding the utilities of descendant nodes underN_{i_1} , i.e., N_{i_2}toN_{i_n}. Finally, in Line 3, (N_{i_1},i_2) is taken as input recursively.

3.2 ENHANCED UP-GROWTH+ ALGORITHM (EUP-GROWTH+)

In Enhanced UP-Growth+, node utility count in each path are used to reduce the overestimated utilities that are closer to their actual utilities values of the unpromising items and descendant nodes.

A minimal node utility for each node can be acquired during the construction of a global IMUP-Tree. First, add an element, namely N.mnu, into each node of IMUP-Tree. N. menu is the minimal node utility of N. When N is traced, N.mnu keeps track of the minimal value of N.name's utility in different transactions. If N.menuis larger than you (N.name, Tcurrent), N.mnu is set to u (N.name, Tcurrent). The global IMUP-Tree with N.mnu in each node is shown in the Fig 2, N.mnu is the last number in each node.

Node utility count for each path can be acquired during the construction of a local IMUP-Tree. First, the node links in IMUP-Tree corresponding to the item i_m, in header table, are traced. Here item i_m*is* a bottom entry in the header table. Found nodes are traced to the root of the IMUP-Tree to get paths related to i_m. All retrieved t, their minimum node utility and support count are collected into i_m's conditional pattern base. Then, Path.node utility count is acquired into i_m's conditional pattern base as shown in the Eq (3.1),

***p.nuc = p.mnu – p.count*→(3.1)**

Where,

p.nuc is the node utility count of p in {i_m }-CPB.

p.mnu is the minimum node utility of p in {i_m}-CPB.

p.count is the support count of p in {i_m }-CPB.

The two strategies are introduced to enhance the UP-Growth⁺ named DPU and DPN. When a local IMUP-Tree is being constructed, minimal node utilities are retrievedfrom the global IMUP-Tree. In the mining process, when a path is retrieved, node utility count of the path is acquired.

3.2.1 Discarding local unpromising items and their estimated Node Utility Count from the path and path utilities of conditional pattern base

Discarding local unpromising items and their estimated Node Utility Count from the paths and path utilities of conditional pattern bases as shown in Eq (3.2)

$$\boldsymbol{pu(p\{i_m\} - CPB) = p.\{i_m\}.nu - \sum_{\forall i \supseteq UI(i_m) - CPB \ \wedge i \subseteq p} mnu(i) + p.nuc} \rightarrow \textbf{(3.2)}$$

3.2.2 Decreasing local Node path utilities for the nodes of local UP-Tree by estimating utilities of descendant Nodes

Decreasing local Node path utilities for the nodes of the local IMUP - Tree by estimating utilities of descendant Nodes as shown in the Eq (4.3)

$$\boldsymbol{N_{i_k}.nu_{new} = N_{i_k}.nu_{old} + pu(p,\{i_m\} - CPB) - \sum_{j=k+1}^{m'} mnu(i_j) + p.nuc} \rightarrow \textbf{(3.3)}$$

DPU is applied to each path by pruning the unpromising items. Remaining promising items of each path are sorted in a descending order which is called as reorganized paths. The DPN is applied during insert_reorganized_transaction$_{mnu}$ into a conditional IMUP-Tree. Assume a reorganized path P_j = <$N_{i_1} N_{i_2} N_{i_3} \ldots . N_{i_{m'}}$>, where N_{i_k} is the nodes in IMUP-Tree and 1 ≤ k ≤ m'.

Table 6 The Subroutine Insert_ Reorganized _Path mnu

Subroutine:Insert _ Reorganized _ Path $_{mnu}(N,i_x)$
Line 1: If N has a child N_{i_x} such that N_{i_x}.item =i_x,initially N_{i_x}.mnu = ∞ increment N_{i_x}.count by P_j . count. Otherwise, create a new child node N_{i_x} with N_{i_x}.item = i_x N_{i_x}.count = P_j . count,N_{i_x}.parent = N N_{i_x}.nu = 0.
Line 2: Increase N_{i_x}.nu by Eq(3.3)
Line 3: If N_{i_x} . mnu > mnu (i_x , P_j) set N_{i_x}.mnu to mnu (i_x , P_j) "
Line 4: If there exists a node N_{i_x} in P_j where x + 1<m',call Insert_ Reorganized _ Path$_{mnu}$ (N_{i_x},i_{x+1})

When N_{i_1}.item,i_1 is inserted into the conditional IMUP-Tree, the function Insert_ Reorganized _ Path $_{mnu}(N_{R'},i_1)$, as shown in Table 6, is called, where N_R' is root node of the conditional UP-Tree. The node fori_1,N_{i_1} is found or created under N_R' and its support is updated in Line 1 the element, minimal node utility, is added into N_{i_x} and set N_{i_x}.*mnu* = ∞ initially. Then DPN is applied in Line 2 by decreasing estimated utilities of descendant nodes underN_{i_1}, i.e., N_{i_2} to$N_{i_{m'}}$. Then, N_{i_x}.*mnu* is checked by inserting the procedure "If N_{i_x}.*mnu*>mnu (i_x, p_j), setN_{i_x}.*mnu* to mnu (i_x, p_j)" . Finally in Line 4, (N_{i_1},i_2) is taken as input recursively.

The complete set of PHUIs is generated by recursively calling the procedure named UP-Growth. Initially, EUP-Growth⁺ *(T_x, H_x, X)*is called, where T_x is the global IMUP- Tree and H_x is the global header table. The procedure of EUP-Growth⁺ is shown in Fig. 5

Table 7 The subroutine of EUP-Growth⁺

Subroutine: Enhanced UP-Growth⁺ (T_x, H_x, X)
Input: A IMUP-treeT_x, a header table H_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 */*
3)If $nu_{sum}(i_k) \geq$ min_util, do
4)Generate a PHUI Y = X ∪ i_k;
5)Set pu(i_k) as estimated utility Y;
6)Construct Y-CPB;
7)Put local promising item in Y-CPB into H_Y
8) Apply DPU to reduce path utilities of the paths;
9)Apply Insert_ Reorganized_ Path $_{mnu}$ to insert into T_Y with DPN;
10)If $T_Y \neq$ null then call Enhanced UP-Growth⁺ (T_x, H_x, X)
11)End if
12)End for

By comparing the existing system, it is clear that the number of Potential High Utility Itemsets, as well as the overestimated utilities of itemsets, are further reduced by EUP-Growth⁺.

IV. EXPERIMENTAL EVALUATIONS

The 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 algorithm is implemented in Java language.To evaluate the performance of the proposed

technique, both real and synthetic data sets are used in the experiments. The Synthetic Transactional data set is generated from the data generated based on fast algorithm for mining association rules [1]. Parameter descriptions and default values of synthetic data sets are shown in Table 9. Real world data sets Retail, a Weblog and Chess are obtained from the FIMI Repository [FIMI]. These data sets do not provide profit values or the quantity of each item for each transaction.

Table 9 Parameter setting of Synthetic Data Sets

Parameter Descriptions	Default
|D| Total number of transactions	100K
T: Average Transactional Length	10
|I|: Number of distinct items	1000
F: Average size of maximal potential frequent itemsets	6
Q: Maximum number of purchase items in transactions	10

As for the performance evaluation of the previous utility based pattern mining[19],[16], unit profits from 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.

Table 10 Characteristics of real data sets

Dataset	|D|	T	|I|	Type
Retail	88162	10.3	16470	Sparse
Chess	3196	37.0	75	Dense
Web Log	1.692.082	71.45	5.267.656	Sparse
Medical	16487	32	497	Sparse

Finally, the results are evaluated by using a real life dataset (medical) with real utility values is collected from medical shoppers that is located in Pollachi. The performance of proposed algorithm was compared with the existing algorithms UP-Growth [30] and UP-growth+ [31]. For convenience, PHUIs are called candidates in our experiments. The characteristics of the above data sets are shown in the Table 10.

4.1 COMPARATIVE ANALYSIS OF PROPOSED ALGORITHM ON DIFFERENT DATA SETS

In this part, the performance comparison on three real data sets are shown: dense data set Chess and sparse data sets Retail and Weblog. First, we show the results on real dense data set Chess in Fig. 4.

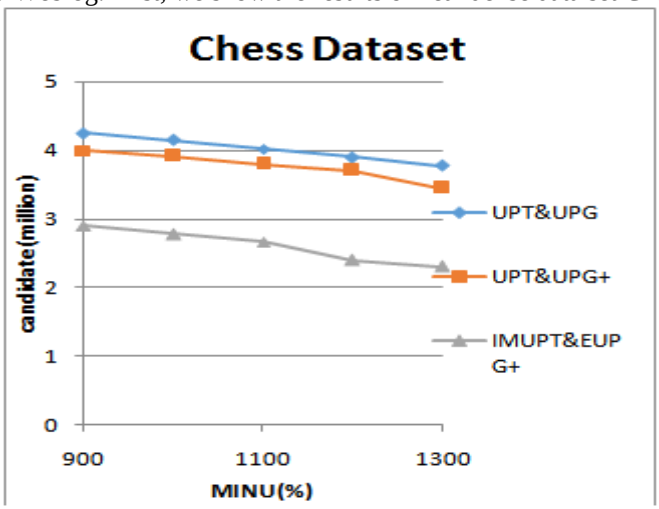

Figure 4 Candidate's comparison on the chess dataset

The chess dataset is an extremely dense data set. Dense data sets have many long frequent as well as high utilization patterns. Because the probability of an item's occurrence is very high in every transaction, for comparatively higher thresholds, dense datasets have too many candidate patterns. Here, first comparing the number of candidates.

From Fig. 4, the performance of the proposed methods substantially outperforms that of previous methods. The minimum utility threshold range of 900% to 1300% is used here. It is remarkable that since chess is a huge dense dataset, a large number of candidate patterns occur at a comparatively higher threshold (above 50%). The number of candidates increases rapidly below the utility threshold of 1000%. For utility thresholds of 900% and 1000%, the numbers of candidate patterns are too large for the existing algorithms. The runtime of UPT&UPG is the worst, followed by UPT&UPG, UPT&UPG$^+$ and IMUPT & EUPG$^+$ is the best. The main reason is the performance of UPT&UPG and UPT&UPG$^+$ is decided by the number of generating candidates.

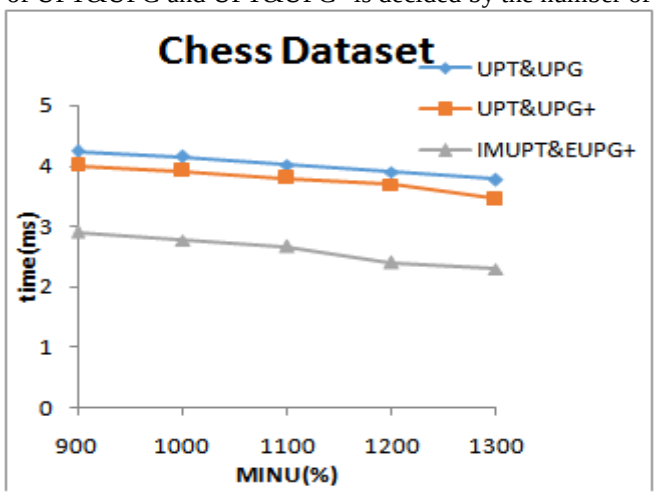

Figure 5 Time Comparision on Chess Data Set

Figure 5 it clearly shows the runtime comparison on the chess dataset. The runtime of the methods is just proportional to their number of candidates, that is, more the candidates produced by the method leads to the greater execution time. Due to several database scans with a large candidate set, the total time needed for the algorithm is also very large when compared to sparse data sets.

4.1.2 Performance analysis for Retail Data Set

The dataset retail is provided by Tom Brijs (FIMI), and contains the retail market basket data from an anonymous Belgian retail store.It is an extreme sparse dataset. Sparse data sets normally have too many distinct items. Although in the average case their transaction length is small, they normally have many transactions.From Fig 5.3, it evidently shows the comparison of the candidate's on the Retail dataset. The minimum utility threshold range of 400 to 800 is used

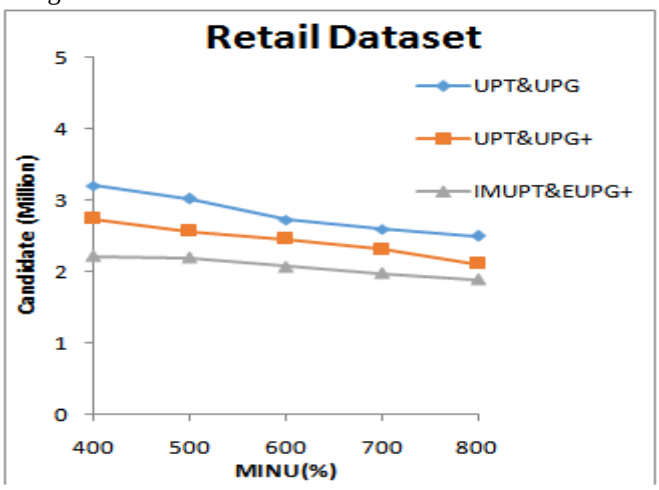

Figure 6 Candidate's comparison on the RetailDataset

here. It is clear that the performance of UPT&UPG is the worst since it generates the most candidates.

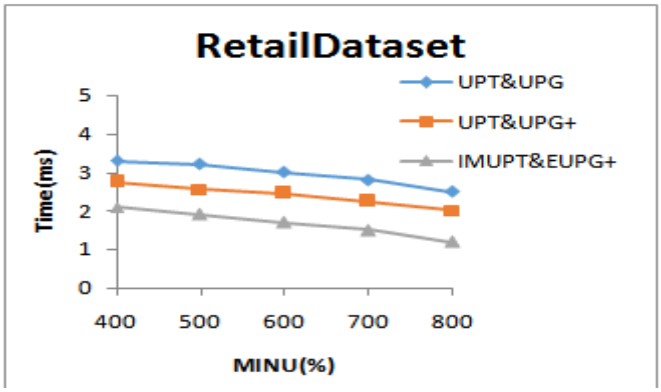

Table 7Time Comparison on the Retail Dataset

From Fig 7, it clearly shows the running time comparison on using the Retail Data Set. It's clear that the runtime of UPT&UPG is the worst, followed by UPT&UPG, UPT&UPG+, and IMUPT&EUPG+ is the best. Besides, although the number of candidates of UPT&FPG, UPT&UPG, and UPT&UPG+ are almost the same, the execution time of UPT&UPG is the worst among the three methods since UP-Growth+ and EUP-Growth + efficiently prune the search space of local IMUP-Trees.

4.2 PERFORMANCE COMPARISON UNDER DIFFERENT PARAMETERS

The performance under varied average transaction length (T) is shown in Fig 8.

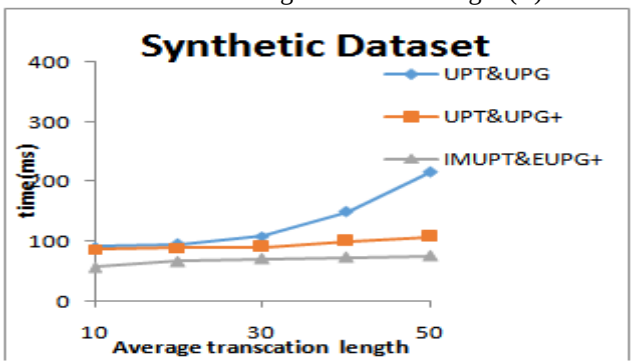

Figure 8 Time Comparison on the Synthetic Dataset

The experiment is performed on synthetic data sets Tx.F6.|I|1,000.|D|100k and min_util is set to 1 percent. From Fig 8, it's evident that the runtime of all algorithms increases with increasing T because when T is larger, transactions and databases become longer and larger. Also, the runtime of the methods is proportional to the number of candidates. The difference of the performance between the methods appears when T is larger than 25. The best method is UPT&UPG+ and the worst one is UPT&UPG.

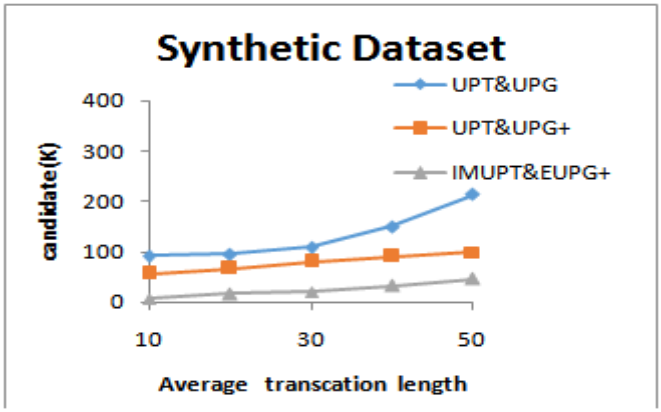

Figure 9 Candidate's comparison on the Synthetic Dataset

From Fig.9, it clearly shows the number of candidates generated by UPT&UPG+ is the smallest. This shows that EUP-Growth+ can effectively prune more candidates by decreasing overestimated utilities when transactions are longer. In other words, UP-Growth+ is more efficient on the data sets with longer transactions.

4.3 SCALABILITY OF THE PROPOSED METHODS

In this section, the scalability of the proposed method is performed on synthetic data sets T10.F6.|I|1,000.|D|xk. Results of run time, number of candidates and the number of high utility itemsets are shown in Fig.10 and Table 10 respectively.

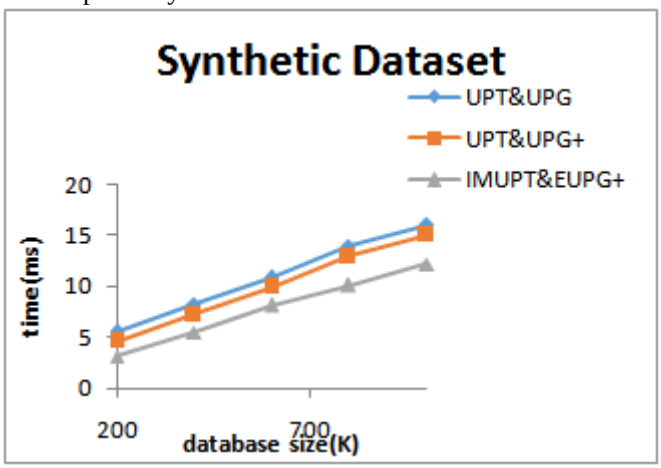

Fig. 10 Experimental results under varied database size.

Fig.10 shows that there are significant differences in runtime on different database size. Total runtime of IMUPT&EUPG+ is the best, followed by UPT&UPG+ and UPT&UPG being the worst. This is because when the size of the database increases, runtime for identifying high utility itemsets also increases. Here, the importance of runtime is emphasized again. From Table 10, it is clear that the number of PHUIs generated by IMUPT&EUPG+ outperforms other methods in the databases with varied database sizes. Overall, the performance of IMUPT&EUPG+ outperforms the other compared algorithms with increasing size of databases since it generates the least PHUIs.

TABLE 10: Number of Candidates and High Utility Item sets under Varied Database Sizes

Database	UPT&UPG	UPT&UPG+	IMUPT&EUPG+
200k	51,534	32,261	18,175
400k	56,844	39,450	18,976
600k	52,845	34,164	17,324
800k	52,491	35,645	18,188
1000k	50,073	32,789	17,144

4.4 MEMORY USAGE OF THE PROPOSED METHODS

In this section, memory consumption of the proposed methods (in GB) is shown in the Tables 11 and 12 under varied min_util on Retail data set and varied database sizes on synthetic data sets T10.F6.|I|1,000.|D|xk, respectively.

In Table 11, it is clear that the memory usage of all methods increases with decreasing min_util since less min_util makes UP-Trees and IMUP- Trees larger. Generally, IMUP&EUPG+ uses the least memory in IMUP-tree to store the PHUIs. This is because the strategies effectively decrease the number of PHUIs in local IMUP-Trees.

Table 11 Memory in (GB) Usage under Varied min_util on Retail Dataset

	Memory (GB)		
Min_util	UPT&UPG	UPT&UPG+	IMUPT&EUPG+
0.1%	1.017	0.478	0.299
0.08%	1.033	0.923	0.916
0.06%	1.132	1.069	1.021

0.04%	1.294	1.188	1.125
0.02%	1.364	1.288	1.174

In the Table 12, it is clear that the memory usage increases with increasing database size. Generally, IMUP&EUPG+ uses the least memory among the three methods.This is because the strategies effectively decrease the number of PHUIs in local IMUP-Trees. On the other hand, the fewer PHUIs are generated by IMUPT&EUPG+, it consumes less memory.

Table 12 Memory in (GB) Usage under Varied Database SizesT10.F6.|I|1000.|D|xK (min_util = 0.1 Percent)

	Memory (GB)		
Database	UPT&UPG	UPT&UPG+	IMUPT&EUPG+
200K	1.221	1.028	0.979
400K	1.350	1.290	1.185
600K	1.490	1.354	1.252
800K	1.555	1.422	1.326
1000K	1.699	1.548	1.437

5.7 Summary of the Experimental Results

Experimental results in this section show that the proposed methods outperform the state-of-the-art algorithms almost in all cases on both real and synthetic data sets. The reasons are described as follows,

First, memory used by global IMUP-Tree is much less the memory used by UP-Tree since DGUM and DGNM effectively decrease memory consumption of utilities by RH-algorithm during the construction of a global IMUP-Tree.

Second, EUP-Growth+ generate much fewer candidates than UP-growth and UP-Growth+ since DPU, and DPN are applied during the construction of local IMUP- Trees. By the proposed algorithm with the strategies, generation of candidates can be more efficient since lots of useless candidates are pruned.

Third, generally, EUP-Growth+ utilizes minimal node utilities and a path node utility count for further decreasing overestimated utilities of itemsets. They are more effective especially when there are many longer transactions in databases. By the reasons mentioned above, the proposed algorithm EUP-Growth+ achieve better performance than UP- Growth and UP-Growth+ algorithms.

V. CONCLUSIONS

In this thesis, an efficient algorithm for mining high utility itemset called EUP-growth+ is proposed for mining high utility itemsets with a set of effective strategies for pruning potential high utility itemsets. A data structure namedIMUP-Tree is proposed for maintaining the information of high utility itemsets into the appropriate memory using hashing techniques for reducing memory and time. The EUP-Growth + algorithm efficiently generatesPHUIs from IMUP-Tree with only two database scans. By developing four strategies, the mining performance is enhanced significantly since both the search space and the number of candidates are effectively reduced. The proposed algorithm builds the IMUP - tree to reduce the memory consumption while storing the utility itemsets. An IMUP-tree is built only for pruned database that fit into main memory easily. According to recent observations, the performances of the algorithms strongly depends on the minimum utility levels and the features of the data sets (the nature and the size of the data sets). Therefore it is employed in the proposed algorithm to guarantee the time and the memory are further reduced in the case of sparse and dense data sets. In the experiments, both real and synthetic data sets were used for performance evaluation. Experimental results reveal that the strategies considerably improved performance by reducing the search space, time and the number of candidates. It is evident from experiments that EUPG+ algorithm outperforms substantially than UPG+ and UPG algorithms.

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]. R. Agrawal and R. Srikant, "Mining Sequential Patterns," Proc. 11th Int'l Conf. Data Eng., pp. 3-14, Mar. 1995.

[3]. 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.

[4]. 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), pp. 68-77, 1998.

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

[6]. J.H. Chang, "Mining Weighted Sequential Patterns in a Sequence Database with a Time-Interval Weight," Knowledge-Based Systems, vol. 24, no. 1, pp. 1-9, 2011.

[7]. M.S. Chen, J.S. Park, and P.S. Yu, "Efficient Data Mining for Path Traversal Patterns," IEEE Trans. Knowledge and Data Eng., vol. 10, no. 2, pp. 209-221, Mar. 1998.

[8]. C. Creighton and S. Hanash, "Mining Gene Expression Databases for Association Rules," Bioinformatics, vol. 19, no. 1, pp. 79-86, 2003.

[9]. M.Y. Eltabakh, M. Ouzzani, M.A. Khalil, W.G. Aref, and A.K. Elmagarmid, "Incremental Mining for Frequent Patterns in Evolving Time Series Databases," Technical Report CSD TR#08- 02, Purdue Univ., 2008.

[10]. A. Erwin, R.P. Gopalan, and N.R. Achuthan, "Efficient Mining of High Utility Itemsets from Large Data Sets," Proc. 12th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD), pp. 554-561, 2008.

[11]. E. Georgii, L. Richter, U. Ruckert, and S. Kramer, "Analyzing Microarray Data Using Quantitative Association Rules," Bioinformatics, vol. 21, pp. 123-129, 2005.

[12]. J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," Proc. Int'l Conf. on Data Eng., pp. 106-115, 1999.

[13]. J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," Proc. 21th Int'l Conf. Very Large Data Bases, pp. 420-431, Sept. 1995.

[14]. J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. ACM-SIGMOD Int'l Conf. Management of Data, pp. 1-12, 2000.

[15]. S.C. Lee, J. Paik, J. Ok, I. Song, and U.M. Kim, "Efficient Mining of User Behaviors by Temporal Mobile Access Patterns," Int'l J. Computer Science Security, vol. 7, no. 2, pp. 285-291, 2007.

[16]. H.F. Li, H.Y. Huang, Y.C. Chen, Y.J. Liu, and S.Y. Lee, "Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams," Proc. IEEE Eighth Int'l Conf. on Data Mining, pp. 881- 886, 2008.

[17]. 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, pp. 198-217, Jan. 2008.

[18]. C.H. Lin, D.Y. Chiu, Y.H. Wu, and A.L.P. Chen, "Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window," Proc. SIAM Int'l Conf. Data Mining (SDM '05), 2005.

[19]. Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm," Proc. Utility-Based Data Mining Workshop, 2005.

[20]. R. Martinez, N. Pasquier, and C. Pasquier, "GenMiner: Mining nonredundant Association Rules from Integrated Gene Expression Data and Annotations," Bioinformatics, vol. 24, pp. 2643-2644, 2008.

[21]. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, "H-Mine: Fast and Space-Preserving Frequent Pattern Mining in Large Databases," IIE Trans. Inst. of Industrial Engineers, vol. 39, no. 6, pp. 593-605, June 2007.

[22]. J. Pei, J. Han, B. MortazaviAsl, H. Pinto, Q. Chen, U. Moal, and M.C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The Prefixspan Approach," IEEE Trans. Knowledge and Data Eng., vol.16, no.10, pp. 1424-1440, Oct. 2004.

[23]. J. Pisharath, Y. Liu, B. Ozisikyilmaz, R. Narayanan, W.K. Liao, A. Choudhary, and G. Memik NU-MineBench Version 2.0 Data Set and Technical Report, http://cucis.ece.northwestern.edu/ projects/DMS/MineBench.html, 2012.

[24]. B.E. Shie, H.-F. Hsiao, V., S. Tseng, and P.S. Yu, "Mining High Utility Mobile Sequential Patterns in Mobile Commerce Environments," Proc. 16th Int'l Conf. Database Systems for Advanced Applications (DASFAA '11), vol. 6587/2011, pp. 224-238, 2011.

[25]. B.E. Shie, V.S. Tseng, and P.S. Yu, "Online Mining of Temporal Maximal Utility Itemsets from Data Streams," Proc. 25th Ann. ACM Symp. Applied Computing, Mar. 2010.

[26]. K. Sun and F. Bai, "Mining Weighted Association Rules without Preassigned Weights," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 4, pp. 489-495, Apr. 2008.

[27]. S.K. Tanbeer, C.F. Ahmed, B.S. Jeong, and Y.K. Lee, "Efficient Frequent Pattern Mining over Data Streams," Proc. ACM 17th Conf. Information and Knowledge Management, 2008.

[28]. F. Tao, F. Murtagh, and M. Farid, "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.

[29]. V.S. Tseng, C.J. Chu, and T. Liang, "Efficient Mining of Temporal High Utility Itemsets from Data Streams," Proc. ACM KDD Workshop Utility-Based Data Mining Workshop (UBDM '06), Aug. 2006.

[30]. 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), pp. 253-262, 2010.

[31]. V.S. Tseng, C.W. Wu, B.E. Shie, and P.S. Yu, " An Efficient Algorithm for Mining High Utility Itemsets from Transactional Database," IEEE Transactions On Knowledge And Data Engineering, Vol. 25, No. 8, August 2013

[32]. W. Wang, J. Yang, and P. Yu, "Efficient Mining of Weighted Association Rules (WAR)," Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '00), pp. 270-274, 2000.

[33]. 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, pp. 28-37, Aug. 2006.

[34]. S.J. Yen and Y.S. Lee, "Mining High Utility Quantitative Association Rules." Proc. Ninth Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK), pp. 283-292, Sept. 2007.

[35]. C.H. Yun and M.S. Chen, "Using PatternJoin and Purchase Combination for Mining Web Transaction Patterns in an Electronic Commerce Environment," Proc. IEEE 24th Ann. Int'l Computer Software and Application Conf., pp. 99-104, Oct. 2000.

[36]. C.H. Yun and M.S. Chen, "Mining Mobile Sequential Patterns in a Mobile Commerce Environment," IEEE Trans. Systems, Man, and CyberneticsPart C: Applications and Rev., vol. 37, no. 2, pp. 278-295, Mar. 2007.

[37]. U. Yun, "An Efficient Mining of Weighted Frequent Patterns with Length Decreasing Support Constraints," KnowledgeBased Systems, vol. 21, no. 8, pp. 741-752, Dec. 2008.

[38]. 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.

[39]. U. Yun and J.J. Leggett, "WIP: Mining Weighted Interesting Patterns with a Strong Weight and/or Support Affinity," Proc. SIAM Int'l Conf. Data Mining (SDM '06), pp. 623-627, Apr. 2006.