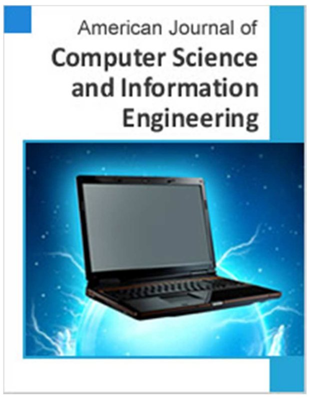
American Journal of Computer Science and Information Engineering
2017; 4(6): 58-63
http://www.aascit.org/journal/ajcsie
ISSN: 2381-1110 (Print); ISSN: 2381-1129 (Online)

American Association for Science and Technology



Up-to-Date High Utility Pattern Mining Algorithm Using Up-to-Date Utility-List for High Quality Pattern

Shaikh Nikhat Fatma

Department of Computer Engineering, Pillai HOC of Engineering and Technology, Rasayani, India

Email address
nikhats10@yahoo.com

Citation
Shaikh Nikhat Fatma. Up-to-Date High Utility Pattern Mining Algorithm Using Up-to-Date Utility-List for High Quality Pattern. *American Journal of Computer Science and Information Engineering*. Vol. 4, No. 6, 2017, pp. 58-63.

Keywords
Association Rule Mining,
Minimum Utility Threshold,
High Utility Pattern Mining,
High Utility Pattern,
Up-to-Date High Utility Pattern,
Utility List,
Up-to-Date Utility List,
Frequent Item Set Mining

Received: September 6, 2017
Accepted: November 16, 2017
Published: December 7, 2017

Abstract
Knowledge discovery in databases (KDD) is to identify efficient and helpful information from large databases and provide automated analysis and solutions. In particular, finding association rules from transaction databases is most commonly seen in data mining. There are several algorithms have been developed to solve the problem to analysis the basket of the customer. These are mainly based on apriori. In fact mining pattern does not meet all the requirement of the business. Utility Mining is one of the extensions of Frequent Item set mining, which discovers item sets that occur frequently. In many real-life applications where utility item sets provide useful information in different decision-making domains such as business transactions, medical, security, fraudulent transactions, retail communities. Many algorithms have been developed to find high-utility patterns (HUPs) from databases without considering timestamp of patterns, especially in recent intervals. A pattern may not be a HUP in an entire database but may be a HUP in recent intervals. In this paper, an improved up-to-date high-utility pattern (UDHUP) is designed. It considers not only utility measure but also timestamp factor to discover the recent HUPs. In this paper the UDHUP- list algorithm is discussed. A new data structure, called up-to-date utility-list (UDU-list), is used to efficiently speed up the performance for mining UDHUPs.

1. Introduction

Large number of studies has been proposed for mining frequent item sets from the databases and successfully adopted in various application domains like market basket analysis, click stream analysis etc. In market basket analysis, mining frequent item sets from a database (usually a transactional database) refers to the discovery of the item sets which frequently appear in the transactions, but the unit profits and purchased quantities of items are not considered in the framework of frequent item set mining. Hence, it cannot satisfy the requirement of the user who is interested in discovering the item sets with high sales profits. In view of this, utility mining emerges as an important topic in data mining for discovering the item sets with high utility like profits. Mining high utility item sets from the databases refers to finding the item sets with high utilities or high profits. The basic meaning of utility is the interestedness /importance /profitability of items to the users. The utility of items in a transaction database consists of several aspects like importance of distinct items, which is called external utility, importance of the items in the transaction, which is called internal utility. An item set is called a high utility item set if its utility is no less than a user specified threshold; otherwise, the item set is called a low utility item set.

Mining high utility item sets 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, mining high utility item sets over data streams has emerged as an interesting topic because many users want to obtain valuable information from stream data, which are continually generated at rapid rates. However, in these environments, most of the previous high utility item set mining methods cannot efficiently work in terms of both runtime and memory usage. In addition, since they conduct their mining processes without any consideration of transactions' arrival-time, it is hard for these methods to sufficiently fulfill the needs of users when they want to obtain only up to date, relevant information over data streams.

Temporal data mining [6, 7] is an attractive way to find temporal patterns and regularities from temporal databases, which can be used to reveal the ordered correlation of item sets along with timestamp. For example, the sales of soft drinks in summer and the sales of mittens in winter should be higher than those in the other seasons. The seasonal or periodic behaviours can only be discovered when the window size is properly set. The fixed window size may, however, hide the important information of the purchase item sets. To solve the limitations of temporal data mining, Hong et al. designed the up-to-date pattern mining to represent not only the frequent item sets in the entire database but also the up-to-date information from its past timestamp to the current one [16]. Based on the up-to-date concept, an item set may not be frequent (large) for an entire database but may be large up to date information since the item set seldom occurs early and may often occur lately. The up-to-date patterns include the recent item sets, which are frequent for a flexible period of time from the current time to its longest past. More useful information of the current usage can thus be provided compared to traditional association rules. For example, a new iPhone may not be considered as a frequent item in the entire database for retailers but may be concerned as a popular sold item during the announcement month or season. Temporal mining is a way to analyze ordered sequential databases with a respect index. The sequential data can be text data, gene sequences, purchased customers' logs or stock data, among others. Traditional way to mine a temporal database is to find the correlations of itemsets along with a fixed window size. Seasonal behaviors or some specific items can only be revealed at the proper window size for the mining process. An itemset may not be frequent over the entire database but may be highly frequent in the recent intervals with temporal factors, which can be seen in Figure 1.

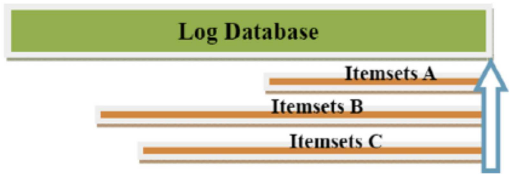


Figure 1. *Up-to –date pattern.*

For Association rules, only the frequency of an item or item set in a transaction is considered, which does not reflect any other factors such as price, quantity or profit. The meaning of ‘‘utility’’ can be defined as various factor based on the users' specification, such as profit, benefit, weight or risk. The problem of high-utility item set mining is to find the item sets (group of items) that generate a high profit in a database, when they are sold together. The user has to provide a value for a threshold called "minutil" (the minimum utility threshold). A high-utility item set mining algorithm outputs all the high-utility item sets, that is the item sets that generates at least "minutil" profit. For example, consider that "minutil" is set to 25 $ by the user.

{a,c}: 28$	{a,c,e}: 31$
{a,b,c,d,e}: 25$	{b,c}: 28$
{b,c,d}: 34$	{b,c,d,e}: 40$
{b,c,e}: 37$	{b,d}: 30$
{b,d,e}: 36$	{b,e}: 31$
{c,e}: 27$	

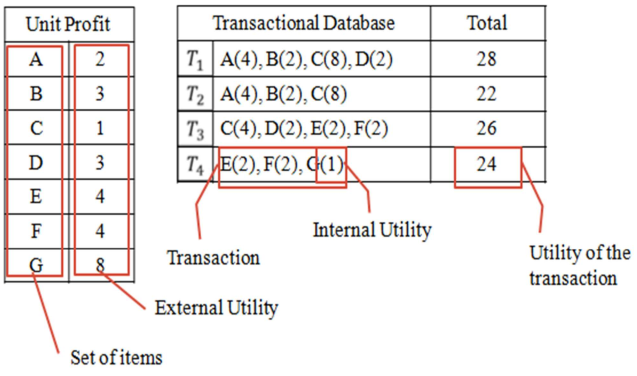
Figure 2. *Example of high utility items.*

Utility of items in transaction database involves following two aspects:

(1) The importance of distinct items, called external utility(e), and

(2) The importance of items in transactions, called internal utility (i).

Utility of Item set (U) = external utility (e) * internal utility (i).



Unit Profit	
A	2
B	3
C	1
D	3
E	4
F	4
G	8

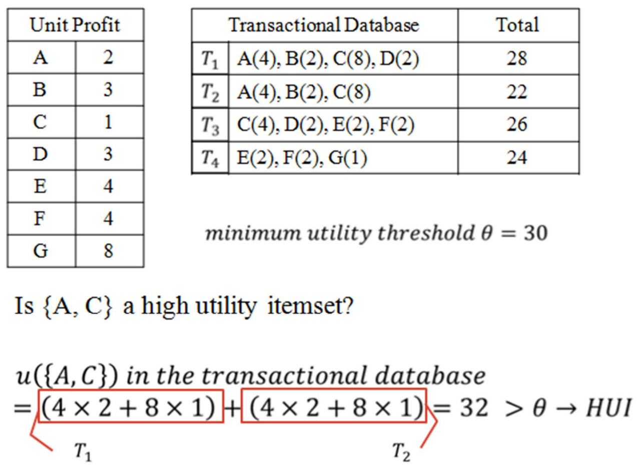
	Transactional Database	Total
T_1	A(4), B(2), C(8), D(2)	28
T_2	A(4), B(2), C(8)	22
T_3	C(4), D(2), E(2), F(2)	26
T_4	E(2), F(2), G(1)	24

Figure 3. *Internal and External Utility.*

High-utility pattern mining (HUPM) was thus proposed to concern both profits (external utility) and sold quantities (internal utility) of the purchase patterns [5]. A pattern is concerned as a high-utility pattern (HUP) if its utility is no less than the pre-defined minimum utility threshold. High-utility item set mining (HUIM) is a useful set of techniques for discovering patterns in transaction databases, which considers both quantity and profit of items. However, most algorithms for mining high-utility item sets (HUIs) assume that the information stored in databases is precise, i.e., that there is no uncertainty. But in many real-life applications, an item or item set is not only present or absent in transactions but is also associated with an existence probability. This is especially the case for data collected experimentally or using noisy sensors. In the past, many algorithms were respectively proposed to effectively mine frequent item sets in uncertain databases. As an example let us analyze sales in a large retail store. We can find that item set {bread, milk} is frequent, item set {caviar, champagne} is of high utility and item set {beer} is utility frequent. A smart manager should pay special attention to item set {beer} as it is frequent and of high utility. On the other side, item set {bread, milk} is frequent but not of high utility and item set {caviar, champagne} gives high utility but is not frequent.

If the utility of an item set X is no less than a user specified minimum utility threshold θ, we call the item set as high utility item set

$$U(X) \geq \theta \rightarrow X \text{ is high utility itemset}$$



Unit Profit	
A	2
B	3
C	1
D	3
E	4
F	4
G	8

	Transactional Database	Total
T_1	A(4), B(2), C(8), D(2)	28
T_2	A(4), B(2), C(8)	22
T_3	C(4), D(2), E(2), F(2)	26
T_4	E(2), F(2), G(1)	24

$minimum\ utility\ threshold\ \theta = 30$

Is {A, C} a high utility itemset?

$$u(\{A,C\})\ in\ the\ transactional\ database = \underbrace{\boxed{(4\times 2+8\times 1)}}_{T_1} + \underbrace{\boxed{(4\times 2+8\times 1)}}_{T_2} = 32 > \theta \rightarrow HUI$$

Figure 4. *High Utility Item set.*

The discovered information of HUPs can be generally used in various applications, such as decision support systems [5], as well as a framework of data mining based analysis [8], or knowledge discovery of material science and engineering [9], to aid mangers or retailers for making the efficient decisions or profitable strategies [10]. A pattern is not a HUP in the entire databases but is a HUP in the recent intervals by considering timestamp factor. For example, the combination of {jacket, stocking} may not be concerned as a HUP in an entire database but only in winter season. It is thus important to find the seasonal or periodic HUPs than the entire ones. In the past, several studies have been addressed the problem of temporal (or temporal maximal) HUPM in data stream [11].

Data mining techniques are used to derive useful and helpful information to aid managers for making efficient decisions in many domains, such as basket analysis, DNA sequence analysis and Web-log analysis. The most common way to discover knowledge focuses on FIM in a binary database, which only reflects the frequency of the presence or absence of an item in the database. In practice, the purchase products or items may contain several quantities in the transactions. High-utility pattern mining (HUPM) is thus an extension of the frequent itemsets mining (FIM) by taking both quantities and profits of items into consideration, thus revealing valuable patterns than frequent ones. For example, the number of purchase jewels or gourmet foods may be lower than the number of beverages but can bring higher profits for retailers.

2. High Utility Pattern Mining

Let $I = \{i1, i2, \ldots, im\}$ be a finite set of items in database D, with each item i_j having corresponding profit $p(i_j)$. An itemset $X \in I$ with k distinct items has length k and is referred to as a k-itemset. The transactional database is denoted as $D = \{T1, T2, \ldots, Tn\}$, where $Tq \chi D$. A quantity $q(i_j, Tq)$ is the sold quantity of item i_j in transaction Tq. An example database and its profit table are respectively shown in Table 1 and Table 2.

The utility of an item ij in Tq is defined as $u(ij, Tq) = q(ij, Tq) \times p(ij)$.

For example, the utility of item $\{b\}$ in $T1$ is calculated as: $u(b, T1) = q(b, T1) \times p(b)$ $(= 1\times2)$ $(= 2)$.

The utility of X in Tq is denoted as $u(X, Tq)$, *which can be defined as*:

$$u(X, Tq) = \sum_{i_j \in X \wedge X \subseteq T_q} u(i_j, T_q)$$

For example, $u(bc, T1) = u(b, T1) + u(c, T1) = q(b, T1) \times p(b) + q(c, T1) \times p(c)$ $(= 1\times2+1\times 2)$ $(= 4)$.

3. Up-to-Date High Utility Pattern (UDHUP)

Many algorithms were proposed to mine HUPs and seldom algorithms were designed to mine UDPs. Mining recent HUPs from a log database by considering timestamps of the itemsets has not been proposed yet. In this paper, a new representation of up-to-date high-utility pattern (UDHUP) is thus designed to reveal more recent knowledge in real-world applications. The definitions of the UDHUP are the same as the HUPM except each transaction is sequentially ordered. The example used in this algorithm is shown in Table 2, and the profit table was shown in Table 1

Table 1. *Profit table for sample database.*

Item	a	b	c	D	E	f	g
Profit	1	2	1	5	4	3	1

Table 2. *Log database with ordered transaction and quantity.*

TID	Transaction time	Item	Transaction utility (tu)
1	2017/4/21 10:00	b:1,c:2, d:1, g:1	10
2	2017/4/21 11:10	a:4,b:1,c:3,d:1,e:1	18
3	2017/4/22 08:00	a:4,c:2,d:1	11
4	2017/4/22 15:00	c:2, e:1, f:1	9
5	2017/4/28 08:30	a:5, b:2, d:1, e:2	22
6	2017/5/01 10:00	a:3,b:4,c:1,f:2	18
7	2017/5/02 13:00	d:1, g:5	10

A user-specified minimum up-to-date high-utility threshold is defined as minUtil, which can be used to find

UDHUP.

The total utility of the database with its past lifetime to the current one is to sum transaction utilities from b to n, which is denoted as:

$$\mathrm{TU}_{[\beta,n]} = \sum_{q=\beta}^{n} \mathrm{tu}\ (\mathrm{Tq})$$

where 1≤β≤n, and n is the number of transactions in D.

For example, TU [1, 7] in Table 2 is to sum all transaction utilities from transaction 1 to 7, which is calculated as (10 + 18 + 11 + 9 + 22 + 18 + 10) (=98). TU [6, 7] in Table 2 is to sum all transaction utilities from transaction 6 to 7, which is calculated as (18 + 10) (=28).

A pattern X is called an up-to-date high-utility pattern (UDHUP) if $u(X)_{[\beta,n]} \geq$ minUtil *TU $_{[\beta,n]}$ with its lifetime from β to n, which can be thus represented as {X: u(X), [β, n]}.

4. UDHUP-List Algorithm

In this paper, a new data structure, called up-to-date utility-list (UDU-list), is used to efficiently speed up the performance for mining UDHUPs. The search space of the UDHUP-list algorithm can be represented by an enumeration tree. The presented UDU-list structure and the search space of UDHUP-list are described below.

UDU-list structure

The up-to-date utility-list (UDU-list) structure which consider timestamp factor is similar to the utility-list structure. An UDU-list of an item set X in a database D keeps a set of entries in which each entry includes the TID number of X, the utility of X in T_q, and the remaining utility of X in T_q. Notice that if an item/item set X after another item/item set Y in a transaction is denoted as Y ≺ X.

An entry of X consists of three elements, including

(1) the TID for X in T_q ($X \subseteq T_q$) denoted as tid,

(2) the set of utility for X in T_q (denoted as X.iu)

(3) the set of remaining utility for X in T_q (denoted as X.ru), in which X.ru is defined as:

$$\mathrm{X.ru} = \sum_{i \subseteq Tq \wedge i \notin Tq \wedge X \prec i} u(i, Tq)$$

Assume that the processing order of patterns in the UDHUP-list algorithm is sorted as the UDTWU-ascending order. For the example in Table 2, the initial UDTWU values of each 1- patterns are calculated as shown in Table 3.

Table 3. *UDTWU of each 1-item.*

Item	$\mathrm{UDTWU}_{[1,7]}$
a	69
b	68
c	66
d	71
e	49
f	27
g	10

The processing order of patterns is thus {g ≺ f ≺ e ≺c≺b ≺a ≺ d}. The constructed UDU-list structures of 1-patterns are shown in Figure 5.

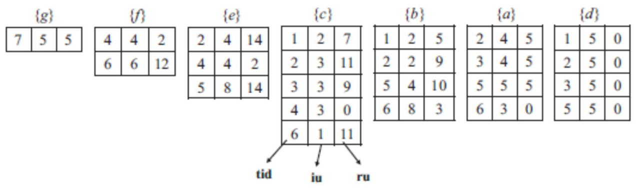


Figure 5. *Constructed UDU – list of 1- patterns.*

The construction procedure of the UDU-list is recursively processed for k-item sets if it is necessary to determine the patterns in the search space. The construction algorithm is shown in UDU- list construction Algorithm below.

```
INPUT: P: a pattern;
Px: the extension of P with an item x;
Py: the extension of P with an item y;
OUTPUT: Pxy.UDUL: the UDU-list of Pxy = Px ∪ Py, x is before y.
1: Pxy.UDUL← Ø. // the UDU-list of Pxy
2: for each entry Ex ∊ Px do
3: if ∃Ey ∊Py.UDUL and Ex.tid = Ey.tid then
4: if P.UDUL ≠ Ø then
5: Search entry E ∊ P.UDUL that E.tid = Ex.tid;
6: Exy ← < Ex.tid, Ey.iu + Ey.iu-E.iu, Ey.ru>.
7: else
8: Exy← < Ex.tid, Ex.iu + Ey.iu, Ey.ru>.
9: Pxy.UDUL ←Pxy.UDUL ∪{Exy}.
10: return Pxy.UDUL.
```

Based on the built UDU-list structure, the necessary information can be obtained by the following two operation as follows.

Give an item set X, the sum of its utilities within its lifetime [β, n] is denoted as X.IU[b,n], which is calculated as:

$$\mathrm{X.IU}_{[\beta,n]} = \sum_{i=\beta}^{n} \mathrm{X.iu}_{[i]}, \beta \in \text{Timelist (X) and } \beta \leq n$$

Give an itemset X, the sum of the remaining utilities except X within its lifetime [β, n] is denoted as X.RU[b,n], which is calculated as:

$$\mathrm{X.RU}_{[\beta,n]} = \sum_{i=\beta}^{n} \mathrm{X.ru}_{[i]},\ \beta \in \text{Timelist (X) and } \beta \leq n$$

The search space of the proposed UDHUP-list algorithm can be represented as the enumeration tree in UDTWU-ascending order of the discovered UDHTWUPs. Each node N is then determined by the summation of the *iu* and *ru* as the upper bound or Timelist (N) to decide whether the subsets of the processed node is required to be determined or not. If the summation of the *iu* and *ru* of the current processed node is larger than or equal to the minimum utility count within its lifetime, the subsets of the processed node will be generated and determined. The utility of N can be determined by its *iu* to find whether it is an UDHUP within its lifetime. The illustrated enumeration tree is shown in Figure 6.

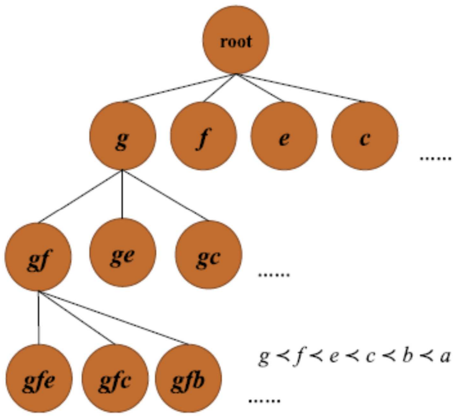


Figure 6. *An enumerated tree.*

The UDHUP-list pruning strategy

Based on the above definitions and the designed UDU-list structure, an efficient pruning strategy is presented to prune the search space, and thus greatly increase the efficiency of the UDHUP-list algorithm.

Property: Given the UDU-list of an item set X with its lifetime in the search space of UDHUP-list w.r.t. an enumeration tree, the sum of all the iu and ru in the UDU-list with its lifetime is always larger than or equal to the up-to-date utility of any one of its children.

According to this Property, a pruning strategy in the enumeration tree can be used to reduce the search space of UDHUPs by exploiting the iu, ru and Timelist of an item set. The sum of all the iu and ru in its lifetime from Timelist can be used to determine whether the item set can be pre-pruned.

Pruning strategy. Let X be an itemset (node) encountered during the depth-first search of the enumeration tree. With any lifetime [β, n], if the sum of $X.IU_{[\beta,n]}$ and $X.RU_{[\beta,n]}$ of the item set /node X according its constructed UDU-list structure is less than the minimum utility count (minUtil X $TU_{[\beta,n]}$), none of the child node X is an UDHUP in lifetime [β',n] (β≤β'), and this part of the search space can be pruned.

Rationale. This pruning condition directly follows from the above Property. According to Property, if the sum of $X.IU_{[\beta,n]}$ and $X.RU_{[\beta,n]}$ is less than the minimum utility count (minUtil X $TU_{[\beta,n]}$) for a given item set (node) X, then any of its child node (supersets) is not a UDHUP in the recent lifetime [β',n] (β≤β'), then can be ignored directly.

According to the above Property, the correctness and completeness of the proposed UDHUP-list algorithm are obtained. This can ensure that the proposed algorithm can extract the correctly UDHUPs based on the sum of all utility values. Moreover, the complete set of UDHUPs can guarantee there are no missing UDHUPs based on the proposed algorithm. The pseudo code of the proposed UDHUP-list algorithm is described below.

```
INPUT: D (n = |D|), a transactional database;
ptable, a pre-defined profit table;
minUtil, the minimum up-to-date utility threshold;
minLen, the pre-defined minimum satisfied length.
OUTPUT: The set of UDHUPs.
1: Timelist ← Ø, acmtu-list ← Ø, and UDULs ← Ø;
2: scan D to calculate the Timelist and acmtu-list;
3: scan D to find UDHTWUP$_1$ ←{UDTWU$(i_j)_{[\beta,n]}$ ≥
   ($TU_{[1,n]}$ -acmtu ($T_\beta$))× minUtil, β ∊ Timelist ($i_j$)};
4: sort 1-items I* ∊ UDHTWUP$^1$ in ascending order of
   their UDTWU;
5: scan D to construct UDULs for each I* in UDHTWUP$^1$;
6: call UDHUP-Mine (Ø, I*, minUtil, minLen);
7: return UDHUPs
```

The UDHUP-list algorithm shown above takes the inputs as:

(1) a transactional database D,

(2) a pre-defined profit table ptable,

(3) the minimum up-to-date utility threshold minUtil, and

(4) the pre-defined minimum satisfied length minLen.

It first scans the database to find the Timelist of each item and the accumulated utility list (acmtu-list) as the necessary information for the later mining process (lines 1–2). The database is thus scanned again to discover the set of UDHTWUP^1 (lines 3), and the set of the UDHTWUP^1 is sorted in ascending order of their UDTWU (lines 4). The UDHUP-list algorithm scans the database to construct UDULs for each I* in UDHTWUP^1 (lines 5). After that, the set of all the 1-extensions of item set I* in UDHTWUP^1 is recursively processed using a UDHUP-Mine procedure (line 6). The depth-first UDHUP-Mine procedure is described below.

```
INPUT: P, a pattern; extUDULOfP, a set of UDU-list of all
  1-extensions of pattern P; minUtil, the minimum up-to-date
  utility threshold; minLen, the pre-defined minimum
  satisfied length.
OUTPUT: The set of UDHUPs.
 1: for each pattern Px ∈ extUDULOfP do
 2:    if Timelist(Px) = Ø or Timelist(Px).first > n – minLen then
 3:       continue.
 4:    for β ← Timelist(Px).first, n do
 5:       if Timelist(Px) = Ø or Timelist(Px).first > n – minLen
  then
 6:          break
 7:       else
 8:          if $Px.IU_{[\beta,n]} + Px.RU_{[\beta,n]} \geqslant (TU_{[1,n]} - acmtu(T_\beta))\times$
  minUtil then
 9:             FlagOfDepthSearch ← true;
10:          if $Px.IU_{[\beta,n]} \geqslant (TU_{[1,n]} - acmtu(T_\beta))\times$ minUtil then
11:             UDHUPs ← UDHUPs ∪ Px;
12:          else
13:             $Px.IU_{[\beta,n]} \leftarrow Px.IU_{[\beta,n]} - PxX.iu_{[\beta]}$;
14:             $Px.RU_{[\beta,n]} \leftarrow Px.RU_{[\beta,n]} - Px.ru_{[\beta]}$;
15:             remove β from Timelist(X);
16:   if FlagOfDepthSearch == true then
17:      extUDULOfPx ← Ø.
18:      for each y after x such that Py ∈ extUDULOfP do
19:         Pxy ← Px ∪ Py;
20:         Pxy.UDUL ← Construct(P, Px, Py);
21:         extUDULOfPx ← extUDULOfPx ∪ Pxy.UDUL;
22:      call UDHUP-Mine (Px, extUDULOfPx, minUtil,
  minLen);
23: return UDHUPs
```

The UDHUP-Mine procedure discovers the designed UDHUPs by continuously shrinking the lifetime of each pattern *Px*. Each pattern *Px* is determined to check whether the early termination is satisfied (lines 2–3). The pattern *Px* is recursively processed until the condition of UDHUP is achieved (lines 8–22). The depth search (lines 16–22) is performed if the summation of iu and ru for Px in the lifetime from β to n ($Px.IU_{[\beta,n]} + Px.RU_{[\beta,n]}$ is no less than minimum utility count (lines 8–9). The UDU-list construction procedure is also called to construct extensions of the UDU-list extensions for a pattern Px (lines 18–21). After that, the UDHUPs can be directly discovered (line 22).

5. Conclusion

In this paper, a new knowledge representation called up-to-date high-utility pattern (UDHUP) is shown to solve the limitations of traditional HUPM for revealing more meaningful and useful HUPs of recent trends. Based on the designed UDHUP, more recent up-to-date information of HUPs can thus be discovered within its effective lifetime. Utility-lists provide not only utility information about item sets but also important pruning information for HUI-Miner. It can mine high utility item sets without candidate generation, which avoids the costly generation and utility computation of candidates.

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: International conference on Very Large Data Bases, 1994, pp. 487-499.

[2] R. Agrawal, R. Srikant, Mining sequential patterns, in: International Conference on Data Engineering, 1995, pp. 3-14.

[3] S. B. Kotsiantis, Supervised machine learning: a review of classification techniques, in: The Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, 2007, pp. 3-24.

[4] P. Berkhin, A survey of clustering data mining techniques, Group. Multidimens. Data (2006) 25-71.

[5] R. Chan, Q. Yang, Y. D. Shen, Mining high utility itemsets, in: IEEE International Conference on Data Mining, 2003, pp. 19-26.

[6] J. M. Ale, G. H. Rossi, An approach to discovering temporal association rules, in: ACM symposium on Applied computing, vol. 1, 2000, pp. 294-300.

[7] C. Y. Chang, M. S. Chen, C. H. Lee, Mining general temporal association rules for items with different exhibition periods, in: IEEE International Conference on Data Mining, 2002, pp. 59-66.

[8] S. Chi, S. J. Suk, Y. Kang, S. P. Mulva, Development of a data mining-based analysis framework for multi-attribute construction project information, Adv. Eng. Inform. 26 (3) (2012) 574-581.

[9] O. AbuOmar, S. Nouranian, R. King, J. L. Bouvard, H. Toghiani, T. E. Lacy, et al., Data mining and knowledge discovery in materials science and engineering: a polymer nanocomposites case study, Adv. Eng. Inform. 27 (4) (2013) 615-624.

[10] C. W. Lin, T. P. Hong, G. C. Lan, J. W. Wong, W. Y. Lin, Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases, Adv. Eng. Inform. 29 (1) (2015) 16-27.

[11] H. F. Li, H. Y. Huang, Y. C. Chen, Y. J. Liu, S. Y. Lee, Fast and memory efficient mining of high utility itemsets in data streams, in: IEEE International Conference on Data Mining, 2008, pp. 881-886.