Article can be accessed online at http://www.publishingindia.com

Utility Mining Algorithms - A Comparative Study

Sivamathi Chokkalingam*, Vijayarani S**

Abstract

Utility mining is an emerging topic in data mining. The aim of utility mining is to discover the itemsets that have maximum utilities. Here utility refers number of items bought, cost of an item or it can be any other user choice in a transaction database. Frequent itemset mining is starting point of utility mining. In frequent itemset mining most often occurring itemsets in a transaction are retrieved. The discovery of such frequent itemsets can help in many business decision making process. Frequent itemset mining concentrates on the number of occurrence of items in a transaction, but not the value of items. But utility mining considers importance of itemsets like the profit it earns in a transaction, quantity in a transaction. In this paper various utility mining algorithms like MEU (Mining with expected utility), FUM (Fast Utility Mining), Two-Phase, CTU-Mine, UP-Growth (Utility Pattern Growth), and FHM (Faster High Utility itemset Mining) MHUI-BIT (Mining High-Utility Itemsets based on BIT vector), MHUT-TID (Mining High-Utility Itemsets based on TIDlist), and THUI (Temporal High Utility Itemsets) are discussed.

Keywords: Utility mining, High utility itemset mining, MEU, MHUI-BIT & MHUT-TID, THUI-Mine, FUM, Two-Phase CTU-Mine, UP-Growth, FHM

Introduction

Data mining refers to extracting or mining knowledge from large amounts of data (Pujari, 2001). Data mining mainly concerns with the analysis of large volumes of data andretries previously unknown relationships or patterns from data. Frequent Itemset mining discovers the items based on how frequent the item occurs in a database without considering its importance. Frequent Itemset mining reflects statistical correlation of an item, but not its semantic significance (Kanimuthu *et al*., 2011). But for many real world applications utility of an item is also equally important.

Consider an electronic super store. Assume that the profit of a mobile phone is 5000 INR and the profit of a memory card is 15 INR. In a transaction database, memory card occurs in 10 transactions and mobile phone occurs in 3 transactions. The total profit of memory card is 150 INR and the total profit of a mobile phone is 15000 INR. As per frequent itemset mining memory card has a higher frequency. But the total profit of a mobile phone is much greater than a memory card. Hence, traditional frequent itemset mining cannot discover the most important itemsets. This is because frequent itemset mining does not consider the profit(i.e. utility) of an item, which is also highly important in decision making. As per utility mining by considering profit mobile phone has high utility.

Utility mining is an important task in marketing decision making process. Mining high utility itemsets from databases is an important task has a wide range of applications such as website click stream analysis, business promotion in chain hypermarkets, cross-marketing in retail stores, online e-commerce management, mobile commerce environment planning and even finding important patterns in biomedical applications (Tseng, Wu, Shie & Yu, 2012).

Rest of the paper is organised as follows: second section

* Ph.D Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, Tamil Nadu, India. Email: c.sivamathi@gmail.com

** Assistant Professor, Department of Computer Science, School of Computer Science and Engineering, Bharathiar University, Coimbatore, Tamil Nadu, India. Email: vijimohan_2000@yahoo.com

discusses utility mining process. Third section presents literature survey on utility mining algorithms, while fourth section describes various utility mining algorithms and its pseudo code developed by researchers. Fifth section presents the conclusion.

Utility Mining Process

In all utility mining algorithms utility of each itemset is calculated based on its internal utility. Each algorithm uses its own data structure to store the item utility information. In general, utility mining algorithms work as follows:

Step 1: Take items from a transaction database.

Step 2: Calculate Utility value for a transaction.

Step 3: Get minimum utility threshold.

Step 4: Output the high utility itemset whose utility is greater than minimum utility threshold.

High Utility Itemset and Frequent Itemset

Consider the database of Table 1. This database contains five transactions T1, T2, T3, T4, and T5. The second column of the database is transaction items appear in a transaction. The third column is utility (say profit) of each item in a particular transaction. The last column is utility of a transaction. For example consider T1, which has items i1, i2 and i3 with occurrence 1, 3 and 2 respectively. Then the utility of this transaction is (1 * 2) + (3 * 3) + (2 * 4) = 19. Similarly utility of remaining transaction are calculated and is given in the last column. If the minimum utility threshold is chosen as 20, the transactions T2, T4 and T5 are having high utility itemset.

High utility itemset differs from frequent itemset mining. To clearly understand the difference, the same database can be used to calculate association rule mining. Association rule can be evaluated using support and confidence metrics. Support calculates fraction of transactions contains itemset i1, i2. Confidence measures how often item i2 occurs in transaction that contains i1. The aim of association rule mining is to find all rules having, support≥minimum_suppport threshold, confidence≥ minimum_confidence threshold.

Table 1: An Example Transaction Database

Transaction id	*Transactions*	*Utility of an transaction*
T1	(i1,1) (i2,3) (i3,2)	19
T2	(i1,2) (i3,3) (i4,1)	21
T3	(i2,2) (i3,1)	10
T4	(i2,3) (i3,1) (i4,3) (i5,2)	32
T5	(i1,1) (i2,3) (i3,2) (i4,1) (i5,2)	28

Table 2: Utility Table

Item	*Profit (utility)*
i1	2
i2	3
i3	4
i4	5
i5	2

Support count (α) is frequency of occurrence of an itemset.

Support count of itemset i1 and i2 αis,

$$\alpha = (\{i1,i2\}) = 2.$$

Consider n = number of transactions.

Support of an itemset = α /n,

Support of itemset ({i1, i2}) = 2/5 = 0.4.

Confidence of an itemset({i1, i2})conf({i1, i2})is,

conf({i1, i2}) = α ({i1, i2}) / α (i1) = 2/2 = 1.

Similarly,

Support of itemset ({i1, i3}) =3/5 = 0.6.

Confidence of an itemset ({i1, i3}) = 3/2= 1.5

Support of itemset ({i2, i3}) =4/5 = 0.8.

Confidence of an itemset ({i2, i3}) = 4/4= 1

If minsup_threshold is chosen as = 0.5,

minconf_threshold = 1 then, ({i1, i3}), ({i2, i3}) are frequent itemset.

Hence traditional frequent itemset mining techniques cannot be applied to utility mining (Tseng *et al.*, 2012). Hence new algorithms are proposed for mining high utility itemset.

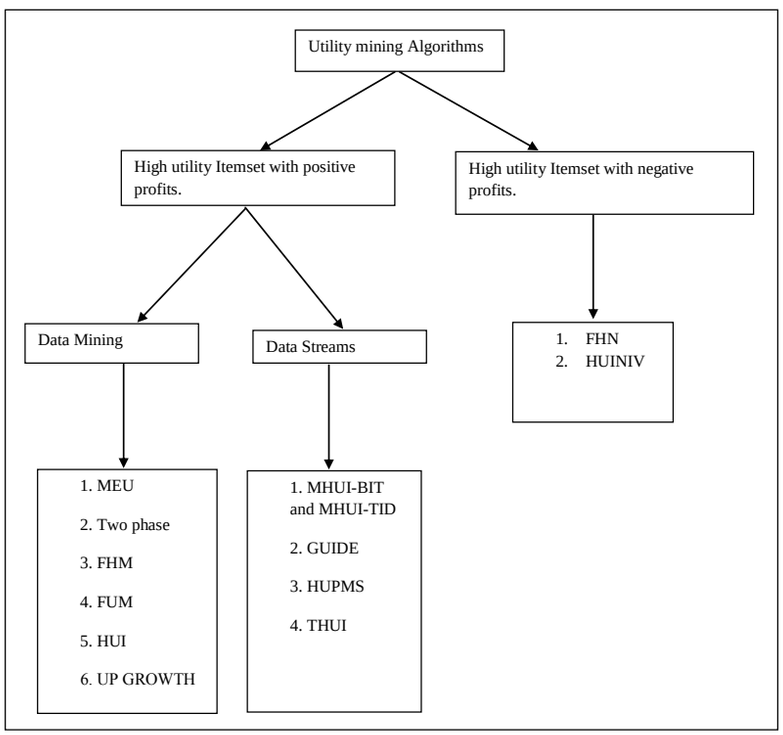
High Utility Itemset Mining

High utility itemset mining refers to the discovery of high utility itemsets. The main objective of high utility itemset mining is to identify the itemsets that have utility values above given utility threshold (Agrawal, Imielinski & Swami,1993). The term utility refers number of itemset, profit of an item , weight, popularity or it can be even measures of user's choice (Agrawal & Srikant, 1994). In this paper, a literature survey of various high utility itemset mining algorithms has been presented.

Literature Review

Many utility based algorithms have been developed. Association rule mining (ARM) identifies frequent itemsets from transaction databases and generates association rules (Agrawal *et al.*, 1993). However, items are actually differs by its utilities. This utility helps to make a strong impact on the decision making in business applications. Therefore, traditional ARM cannot meet all the demands of business applications. The base of these traditional ARM algorithms is the "downward closure property" (also known as anti-monotone property) is defined as any subset of a frequent itemset must also be frequent (Agrawal *et al.*, 1993). This approach is well-organised since more number of item combinations can be ignored at each level. However, this property doesn't apply to the utility mining model. In utility mining, utility of a particular item may be low, but its superset may be a high utility itemset. An overview of various high utility itemset mining algorithms, its basic concepts and techniques used in various research papers have been given in this section. Fig. 1 displays of various high utility mining algorithms. Yao, Hamilton, and Butz (2004) proposed an algorithm called MEU (mining with expected utility). It finds all itemsets in a transaction database with utility values higher than the minimum utility threshold. Liu, Liao, and Choudhary (2005)proposed Two-Phase algorithm for finding high utility itemsets. Shankar, Purusothoman, Jayanthi and Babu (2009) presented an algorithm known as Fast Utility Mining (FUM). Tseng *et al*. (2012) proposed an efficient algorithm, namely UP-Growth i.e Utility Pattern Growth. All these algorithms calculate high utility values only for positive integers. There are algorithms like FHN (Faster High-Utility itemset miner with Negative unit profits)proposed by Fournier-Viger, Wu, Zida, and Tseng (2014) and HUINIV (High Utility Itemsets with Negative Item Values)-Mine, by Chu, Tseng, and Liang (n.d.) to handle itemsets with negative profits. In supermarkets they may offer free items to promote the sales. Such free items have negative utility. The algorithms HUNIV and FHN can handle transaction with items that have positive and negative utilities.

Fig. 1: Utility Mining Algorithms



Utility Mining Algorithms

Utility mining is useful in marketing, retail and wholesale business. By knowing high utility itemset, marketers can promote their sales. Recent researchers developed many algorithms to calculate high utility itemset. MEU, Two-Phase, FHM, FUM, UP-Growth, CTU-Mine are algorithms that calculate high utility itemset with positive profit only. MHUI-BIT and MHUI-TID, GUIDE, HUPMS and THUI algorithms can discover high utility itemset in data streams. FHN and HUINIV algorithms can handle itemset with negative profits.

MEU Algorithm

In the paper by Yao *et al* .(2004), a theoretical model called MEU (Mining with expected utility) is proposed, which finds all itemsets in a transaction database with utility values higher than the minimum utility threshold. MEU prunes the search space by predicting the high utility k-itemset, with the expected utility value. This expected utility value is predicted from the utility values of all items. If utility of an item is greater than this expected value, them it is added to the candidate set, otherwise, it is pruned. This laid the foundation for future utility mining algorithms. The drawback of this algorithm is that the prediction may overestimate. Such thing may lead to high memory space and more computation power. In addition, MEU may miss some high utility itemsets if the variance of the itemset supports is large.

An algorithm for frequent item set mining was presented (Hu & Mojsilovic, 2007). It uses a data structure called High yield partition tree. This algorithm finds segments of data, defined through certain rules. So this algorithm can be used to identify high utility items. It uses "rule-discovery" approach (Hu & Mojsilovic , 2007). This gives a group of patterns that contribute a predefined objective function.

MHUI-BIT and MHUI-TID Algorithm

In their paper,Li, Huang, Cheng Chen, Liu and Lee(2008) proposed two efficient sliding window based algorithms, MHUI-BIT (Mining High-Utility Itemsets based on BITvector) and MHUI-TID (Mining High-Utility Itemsets based on TID list), for mining high utility itemsets. Bitvector and TID list are used to represent item information. A tree-based summary data structure LexTree-2HTU was developed. This improves the efficiency of mining high utility itemsets.

Two-Phase Algorithm

Liu *et al*. (2005) proposed Two-Phase algorithm for finding high utility itemsets. In the first phase, a model called "transaction-weighted utilisation mining" was developed. At each level during the level-wise search the combinations of itemsets that have high transaction-weighted utilisation are added into the candidate set. Phase I may overestimate some low utility itemsets, but it never underestimates any itemsets (Liu*et al*., 2005). In phase II, database is scanned again to remove overestimated itemsets.

Fig. 2: Pseudo Code of the 2P Algorithm

```
Input:     database DB. constraintsminUtil and minSup
Output: all utility-frequent itemsets
/* Phase 1: find all quasi utility-frequent itemsets */
[1] CandidateSet = QUF-APriori(DB, minUtil, minSup)
/* Phase 2: prune utility-infrequent itemsets */
[2] foreach c in CandidateSet:
[3] foreach T in DB:
[4] if c in T and u(c,T) >= minUtil:
[5] c.count += 1
[6] return {c in CandidateSet | c.count>= minSup}
```

THUI

A novel method, namely THUI (Temporal High Utility Itemsets) –Mine was proposed by Tseng (2009), for mining temporal high utility itemsets from data streams. It can effectively discover the temporal high utility itemsets by generating a few temporal high transaction-weighted. This algorithm not only considers profits and quantities but also common branches of items in a multi-database environment. Utilisation 2-itemsets such that the execution time can be reduced substantially in mining all high utility itemsets in data streams (Tseng, 2009).Thus all temporal high utility itemsets in all time windows are discovered. The advantages of this algorithm are less candidate itemsets, less execution time, low memory space etc.

Rare Utility Itemsets Algorithm

G.C.Lan et al proposed a new kind of patterns, called Rare Utility Itemsets (Lan, Hong &Tseng, n.d.). The algorithm

TP-RUI-MD (Two-Phase Algorithm for Mining Rare Utility Itemsets in Multiple Databases) was proposed to retrieve rare utility itemsets in a multi-database environment. Here rare utility itemsets refer to itemsets that occur rarely in a database and have combination of high utility itemsets. The TP-RUI-MD algorithm uses the level-wise technique for discovering the rare-utility itemsets in a multi-database environment (Lan *et al.*, n.d.).

FUM Algorithm

Shankar *et al*. (2009) present an algorithm known as Fast Utility Mining (FUM). This algorithm retrieves all high utility itemsets. It will not consider the entire set of itemsets in a transaction. FUM considers the distinct itemsets involved in a transaction. Thus the candidate set of combinations of distinct itemsets in a transaction is calculated. This reduces the execution time and improves performance efficiency. It also demands less main memory storage requirements, hence reduced hardware cost.By combining FUM and Fast Utility Frequent mining (FUFM) algorithms authors also propose different types of itemsets such as High Utility and High Frequency itemsets (HUHF), High Utility and Low Frequency itemsets (HULF), Low Utility and High Frequency itemsets (LUHF), and Low Utility and Low Frequency itemsets (LULF).

Fig. 3: Pseudocode of the FUM Algorithm

```
Input:    Database DB, Transaction T,minUtil
Output:  High Utility ItemsetsH
[1] Compute the utility value ∀single itemset
[2] for each T ϵ DB
[3] begin
[4]       if T ⊄S {where S ⊂ DB k S= [0 ..T – 1]}
[5]       begin
[6]       CandidateSet – CombinationGenerator(T);
[7]       for each C ϵ CandidateSet
[8]                 begin
[9]                           if(C ⊄ H) ^ (U(C,T)>minUtil)
[10]                          H.add(C);
[11]                end
[12]      end
[13]end
[14]return(H);
CombinationGenerator(T) – Generate all possible
combination of itemsetϵ T.
```

Fig. 4: Pseudo Code of the FUFM Algorithm

```
Input: database DB.constraintsminUtil and minSup
Output: all utility-frequent itemsets
[1] L = 1
[2] find the set of candidates of length L with support >= minSup
[3] computeexteded support for all candidates and
output utility frequent itemsets
[4] Increment the value of L by 1
[5] use the frequent itemset mining algorithm to obtain new
set of frequent candidates of length L from the old set of
frequent candidates
[6] stop if the new set is empty otherwise go to [3]
```

Fig. 5: Pseudo Code of the HUHF Algorithm

```
Input     :Database DB; Constraints minUtil and minSup
Output   : Ranked list of customers who buy HUHF items
[1] Compute High utility and high frequent (HUHF) itemsets
using FUFM algorithm
[2] For each I ∈ HUHF itemset, scan the database DB to find the customers who
buy that itemset
[3] Increment the count value associated with the customer who is a buyer of I.
[4] Stop if the HUHF is empty else Go to [2]
[5] List the HUHF customers in descending order of the count value associated
with each customer
[6] return (list of HUHF customers)
```

Fig. 6: Pseudo Code of the HULF Itemsets Mining

```
Input: Database DB; Constraints minUtil and minSup
Output : High Utility and Low Frequency Itemsets (HULF)
[1] Compute High utility itemsets HU using FUM algorithm.
[2] Compute High utility and high frequent itemsets HUHF using FUFM algorithm.
[3] HULF = HU -HUHF /*set difference operation*/
[4] return (HULF)
```

Fig. 7: Pseudo Code of the LULF Itemsets Mining Algorithm

```
Input: Database DB; Constraints min_Util and min_Sup, LUHF
Output: Low Utility and Low Frequency Itemsets (LULF)
[1] Compute the utility value ∀single itemset
[2] For eachT ∈ DB
[3] begin
[4] if T ∉S {where I⊆ DB | I = [0 .. T-1]}
[5]     begin
[6]         Candidateset = CombinationGenerator (T)
[7]         For each C ∈CandidateSet
[8]         begin
[9]         if (C ∉ H) ) ^ U(C,T) <min_Util )
[10]        LU.add (C);
[11]        end
[12]    end
[13] end
[14] LULF = LU - LUHF/*set minus operation*/
[15] return (LULF)
CombinationGenerator(T) - Generate all possible combinations of itemset∈T
```

Fig.8: Pseudo Code of the LUHFI

```
Input: Database DB; Constraints minUtil and minSup
Output: Ranked list of customers who buy Low utility and high frequent items(LUHF)
[1] Compute LUHFitemsets using LUHFM algorithm
[2] For each I ∈ LUHF itemset, scan the database DB to
find the customers who buy I.
[3] Increment the count value associated with the
customer who is a buyer of I.
[4] Stop if the LUHF is empty else go to [2]
[5] List the LUHF customers in descending order of the
count value associated with each customer
[6] return (list of LUHF customers)
```

Fig.9: Pseudo Code of the HULFI Customer List

```
Input: Database DB; minUtil and minSup
Output: List of customers who buy High Utility Low Frequent items
[1] Compute High utility and low frequent (HULF) itemsets using HULFM algorithm
[2] For each I ∈HULF itemset, scan the database DB to find the customers who buy
that itemset
[3] Increment the count value associated with the customer whenever the customer
buys I.
[4] Stop if the HULF is empty else go to [2]
[5] List the HULF customers in descending order of the count value associated with
each customer
[6] return (list of High Utility Low Frequent items customers)
```

UP-Growth Algorithm

Vincent S. Tseng proposed an efficient algorithm, namely UP-Growth i.e Utility Pattern Growth (Tseng *et al*., 2012), for mining high utility itemsets with a set of techniques for pruning candidate itemsets. UP-Tree (Utility Pattern Tree) is a data structure used in this algorithm to store information of high utility itemsets. Here candidate itemsets are generated within two scans of the database. It has good performance when compared to other algorithms.

Fig. 10: Pseudocode of UP-Span algorithm

```
Input:  [1]CES:complex event sequence;
        [2]min_utility: minimum utility threshold;
        [3]MTD: maximum time duration;
Output: HUE_Set: The complete set of high utility episodes;
[1] Scan CES once to find high utility 1-epsiodes and calculate their EWUs and catch
     the associated minimal occurrences;
[2]  for each global event α do
[3]      if ( EWU ( α )≥min_utility)then
[4]          MiningHUE( α , moSet ( α ), MTD, min_utility);
[5]      end if;
[6]  ProcedureMiningHUE( episode α , moSet( α ),MTD, min_utility )
[7]  MiningSimultHUE( α , moSet ( α ), MTD, min_utility );
[8]  MiningSerialHUE( α , moSet ( α ), MTD , min_utility );
```

FHM Algorithm

Fournier-Viger *et al.* (2014) proposed an algorithm called FHM (Faster High Utility itemset Mining). It uses Estimated Utility Co-occurrence Pruning. It pre-calculates the transaction weighted utility measures of 2 itemsets. If an itemset contains a 2 itemsetand its transaction weighted utility is less than minutil, then it is considered as low utility itemset. Hence supersets, join of these itemsets are not calculated. Hence this algorithm avoids 95% of join operation and thus has very high performance.

Table 3: Comparison of Utility Mining Algorithms in Data Mining

Algorithm	*Authors*	*Features*	*Limitation*
MEU	Yao *et al.*	Based on heuristic value.	Slow
Two-Phase	Liu *et al.*	High utility itemset in traditional database	Multiple scans of a database
FHM	Fournier-Viger*et al.*	Uses Estimated Utility Co occurrence Pruning.	Static database
FUM	Shankar *et al.*	Handles distinct set	It generates the combinations for the already generated subset of the itemsets.
UP-Growth	Tseng *et al.*	Pruning candidate itemset with two scans.	Uses tree structure
CTU-Mine	Erwin *et al.*	High utility itemset for pattern growth and dense data	Uses tree structure

Table 4: Comparison of Utility Mining Algorithms in Data Streams

Algorithm	*Authors*	*Features*	*Limitation*
MHUI-BIT & MHUT-TID	Li *et al.*	Item information, lexTree and HTU for data	Use join operation
THUI-Mine	Tseng *et al.*	Generates few candidate and high performance	Use join operation
GUIDE	Tseng *et al.*	Individual models for sliding , landmark window and time fading are given. High Performance and low memory usage.	Use MUI tree structure
HUPMS	Ahmed, Tanbeer, Jeong, and Choi	Interactive and incremental mining is possible.	Use HUS tree structure

Conclusion

In data mining, association rule mining is one of the most important tasks, that considers mining of frequent itemsets only. It does not consider profit or cost of an itemset. In real life applications there may be some items, which may not be frequent but may have high profit. Utility mining mines high utility itemsets in a transaction databse. It is very beneficial in several real-life applications. In this paper, we have presented a brief overview of various algorithms for high utility itemset mining. Among them FHM algorithm has higher performance. However most of these high utility itemset mining algorithms focus on static databases. Today in many applications, the data is processed in the continuous dynamic data streams. So more research issues now focus on high utility itemset mining in data streams. Also more research focus on high utility itemsets with negative profit.

References

Agrawal, R., Imielinski, T., &Swami, A. (1993). *Mining association rules between sets of items in large databases*. In Proceedings of the ACM SIGMOD International Conference on Management of Data, (pp. 207-216).

Agrawal, R., & Srikant, R.(1994). *Fast Algorithms for Mining Association Rules*. In Proceedings of the 20th International Conference Very Large Databases, (pp. 487-499).

Fournier-Viger, P., Wu, C., Zida, S., Tseng, V. S. (2014). *FHM: Faster High-Utility Item-set Mining using Estimated Utility Co-occurrence Pruning.* In Proceeding of 21st International Symposium on methodologies for intelligent systems (ISMIS 2014), (pp.83-92).

Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2008). Efficient mining of high utility item-sets from large datasets. *Advances in Knowledge Discovery and*

Data Mining, 5012, (pp. 554-561). *Utility Mining Algorithms - A Comparative Study.*

Hu, J., &Mojsilovic, A. (2007). High-utility pattern mining: A method for discovery of high-utility item sets. *Pattern Recognition*, 40(11), 3317-3324.

Kanimuthu, S., Premalatha, K., & Shankar, S. (2011). iFUM - Improved Fast Utility Mining. *International Journal of Computer Applications,* August, 27(11), 32-36.

Lan, G. C., Hong, T. P., &Tseng, V. S. (n.d.). *An Algorithm for Mining Rare-Utility Itemsets in a Multi-Database Environment*.

Li, H. F., Huang, H. Y., Cheng Chen, Y., Liu, Y., &Lee, S. (2008). *Fast and Memory Efficient Mining of High Utility Item-sets in Data Streams*. 8th IEEE International Conference on Data Mining.

Liu, Y., Liao, W., & Choudhary, A. (2005). *A Fast High Utility Item-sets Mining Algorithm*. In Proceedings of the Utility-based Data Mining Workshop.

Pujari, A. K. (2001). *Data mining Techniques.* A text book published by University Press (India) Private limited, (pp. 42).

Shankar, S., Purusothoman, T. P., Jayanthi, S., & Babu, N. (2009). *A Fast Algorithm for Mining High Utility Itemsets.* Proceedings of IEEE International Advance Computing Conference (IACC 2009), Patiala, India, (pp.1459-1464).

Tseng, V. S., Chu, C. J., & Liang, T. (2006). *Efficient mining of temporal high utility item-sets from data streams.* In Proceedings of ACM KDD Workshop Utility-based Data Mining (UBDM'06), Philadelphia, Pennsylvania, USA.

Tseng, V. S. (2009). *Efficient Mining of Temporal High Utility Itemsets.*

Tseng, V. S., Wu, C., Shie, B., &Yu, P. S. (2012). *Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases.* IEEE Transactions on Knowledge and Data Engineering, March, 25(8), 1772 - 1786.

Yao, H., Hamilton, H. J., &Butz, C.,J. (2004). *A Foundational Approach to Mining Itemset Utilities from Databases*. Proceedings of the 3rd SIAM International Conference on Data Mining, Orlando, Florida, (pp. 482-486).