Scientific Journal of Impact Factor(SJIF): 3.134

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 6, June -2015

A NOVEL ALGORITHM FOR MINING HIGH UTILITY ITEMSETS

Mrs. Hetal M. Shah[1], Prof. Bhavesh A. Oza[2]

[1,2]Computer Science & Engineering ,L.D. College of Engineering ,Ahmedabad, GTU-Gujarat, India

Abstract— *Data mining is an activity fretful with examining of huge volumes of data to extract some useful information or knowledge and interesting patterns or relationships which in turn leads to better understanding of the essential processes. An emerging topic in the field of data mining is Utility Mining. The main objective of Utility Mining is to identify the itemsets with high utilities, by considering profit, quantity, cost or other user preferences. Mining High Utility itemsets from a transaction database is to find itemsets that have utility above a user-specified threshold. In this paper We also propose a new algorithm for high utility item set mining. The new algorithm outperforms previous algorithms in terms of execution time.*

Keywords— *Frequent Itemset Mining, High Utility Itemset Mining, Transaction Weighted Utility (TWU), Threshold value (min_util)*

1. INTRODUCTION

Data mining is concerned with analysis of large volumes of data to automatically discover interesting patterns or relationships which in turn leads to better understanding of the fundamental processes. The primary goal is to discover hidden patterns, unexpected trends in the data. Data mining activities uses combination of techniques from database technologies, statistics, artificial intelligence and machine learning. The term is frequently misused to mean any form of large-scale data or information processing. The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns. Over the last two decades data mining has emerged as a significant research area .This is primary due to the inter-disciplinary nature of the subject and the diverse range of application domains in which data mining based products and techniques are being employed. This includes bioinformatics, genetics, medicine, clinical research, education, retail and marketing research. Data mining has been considerably used in the analysis of customer transactions in retail research where it is termed as market basket analysis. Market basket analysis has also been used to identify the purchase patterns of the alpha consumer. Alpha consumers are people that play a key role in connecting with the concept behind the inception and design of a product. The researchers were concentrated on the items which are frequently purchased by the customers. Different algorithms were proposed to find the frequent item sets like Apriori, FP-TREE etc. Later their interest moved to the high utility item sets. Utility may be in any form like profit, cost, sales or any other user preferences. An item set is said to be a high utility item [3], if the utility of that item is greater than or equal to user specified utility. The utility is obtained by the product of internal utility and external utility. The internal utility is the quantity of each item and the external utility is the profit obtained on that item. The most usual form that is also used in this paper is defined as a sum of products of internal and external utilities of present items. The goal of high utility itemset mining is to find all itemsets that give utility greater or equal to the user specified threshold.

The retail organizations stores large amounts of data regarding their sales and customers in databases. Later they view those databases and extract the information which they are interested in. Some organizations may interest in the frequent itemsets purchased by the customers, some may interested in the items which gives more profit to the organization. The organizations extract the information required to them based on their interest from the databases. This information helps the organization later, to take decisions to improve their market.

1.2 Frequent Itemset Mining

Frequent Itemset Mining (FIM) is a popular data mining task that is essential to a wide range of applications. Given a transaction database, FIM consists of discovering frequent itemsets. i.e. groups of items (itemsets) appearing frequently in transactions. However, an important limitation of FIM is that it assumes that each item cannot appear more than once in each transaction and that all items have the same importance (weight, unit profit or value). These assumptions often do not hold in real applications. For example, consider a database of customer transactions containing information about the quantities of items in each transaction and the unit profit of each item. FIM mining algorithms would discard this

information and may thus discover many frequent itemsets generating a low profit and fail to discover less frequent itemsets that generate a high profit.

1.3 High Utility Itemset Mining

The problem of FIM has been redefined as High-Utility Itemset Mining (HUIM) to consider the case where items can appear more than once in each transaction and where each item has a weight (e.g. unit profit). The goal of HUIM is to discover itemsets having a high utility (e.g. generating a high profit). HUIM has a wide range of applications such as website click stream analysis, cross-marketing in retail stores and biomedical applications. HUIM has also inspired several important data mining tasks such as high-utility sequential pattern mining and high-utility stream mining [10].Many studies have been carried to develop efficient HUIM algorithms. A popular approach to HUIM is to discover high-utility itemsets in two phases using the Transaction-Weighted-Downward closure model. This approach is adopted by algorithms such as Two-Phase, IHUP and UP Growth. These algorithms first generate a set of candidate high-utility itemsets by overestimating their utility in Phase 1. Then, in Phase 2, the algorithms perform a database scan to calculate the exact utility of candidates and filter low-utility itemsets. Recently, a more efficient approach was proposed in the HUI-Miner algorithm to mine high-utility itemsets directly using a single phase. HUI-Miner was shown to outperform previous algorithms and is thus the current best algorithm for HUIM. However, the task of high-utility itemset mining remains very costly in terms of execution time. Therefore, it remains an important challenge to design more efficient algorithms for this task.Our proposal is based on the observation that although HUI-Miner performs a single phase and thus do not generate candidates as per the definition of the two-phase model, HUI-Miner explores the search space of itemsets by generating itemsets and a costly join operation has to be performed to evaluate the utility of each itemset. To reduce the number of joins that are performed, we propose a novel pruning strategy named (**Approximated Utility based Pruning**) that can prune itemsets without having to perform joins. This strategy is easy to implement and very effective. We compare the performance of proposed HUIM and existing on four real-life datasets. Results show that proposed HUIM performs up to 95 % less join operations than existing HUIMiner and is up to six times faster than existing.We introduce our novel algorithm, which improves upon existing HUIM by being able to eliminate low-utility itemsets without performing join operations.

2. RELATED WORK

Table 1: Example Database

TID	TRANSACTION	TU
T1	(A,1),(C,1),(D,1)	8
T2	(A,2),(C,6),(E,2),(G,5)	27
T3	(A,1),(B,2),(C,1),(D,6),(E,1),(F,5)	30
T4	(B,4),(C,3),(D,3),(E,1)	20
T5	(B,2),(C,2),(E,1),(G,2)	11

Table 2: Profit Table

Item	A	B	C	D	E	F	G
Profit	5	2	1	2	3	1	1

Definition 1: A frequent itemset is a set of items that appears at least in a pre-specified number of transactions. Formally, let I = {i1, i2, . . . , im} be a set of items and DB = {T1, T2, ..., Tn} a set of transactions where every transaction is also a set of items (i.e. itemset).
Definition 2 : The utility of an item ip is a numerical value yp defined by the user. It is transaction independent and reflects importance (usually profit) of the item. External utilities are stored in an utility table.
Definition 3: The utility of an item set X in a transaction Ti is denoted by U(X, Ti) & it is calculated as follows. For example ,U({*AC*}, *T1*) = U({*A*}, *T1*) + U({*C*}, *T1*) = 5 + 1 = 6.
Definition 4: The utility of an item set X in D is denoted by U(X) & it is calculated as follows
For example, U({*AD*}) = U({*AD*}, *T1*) + U({*AD*}, *T3*) = 7 + 17 = 24.
Definition 5: An itemset is called a *high utility itemset* if its utility is no less than a user-specified *minimum utility threshold* which is denoted as *min_util*. Otherwise, it is called a *low utility itemset*

3. HIGH UTILITY ITEMSETS MINING WITHOUT CANDIDATE GENERATION

Table 3: Transaction Utility Values

Item	TWU
A	65
B	61

C	96
D	58
E	88
F	30
G	38

Table 4: Approximated Utility Based Structure (AUBS)

Item	A	B	C	D	E	F
B	30					
C	65	61				
D	38	50	58			
E	57	61	77	50		
F	30	30	30	30	30	
G	27	38	38	0	38	0

In this section, we present our proposal, the new HUIM algorithm. The main procedure (Algorithm 1) takes as input a transaction database with utility values and the min_util threshold. The algorithm first scans the database to calculate the TWU of each item. Then, the algorithm identifies the set of all items having a TWU no less than min_util (other items are ignored since they cannot be part of a high-utility itemsets by Property 3).The TWU values of items are then used to establish a total order γ on items, which is the order of ascending TWU values.

A second database scan is then performed. During this database scan, items in transactions are reordered according to the total order γ, the utility-list of each item $i \epsilon I^*$ is built and our novel structure named AUBS (Approximated Utility Based Structure) is built. This latter structure is defined as a set of triples of the form $(a,b,c) \epsilon I^* \times I^* \times R$. A triples (a,b,c) indicates that TWU ({a,b})=c.The AUBS can be implemented as a triangular matrix as shown in Fig. 2 (right) or as a hashmap of hashmaps where only tuples of the form (a,b,c) such that $c \neq 0$ are kept. In our implementation, we have used this latter representation to be more memory efficient because we have observed that few items co-occurs with other items. Building the AUBS is very fast (it is performed with a single database scan) and occupies a small amount of memory, bounded by $|I^*| \times |I^*|$. After the construction of the AUBS, the depth-first search exploration of itemsets starts by calling the recursive procedure Search with the empty itemset Ø, the set of single items I^* ,min_util and the AUBS structure.

Search procedure starts from single items, it recursively explore the search space of itemsets by appending single items and it only prunes the ,search space based on Property 5. It can be easily seen based on Property 4 and 5 that this procedure is correct and complete to discover all high-utility itemsets.

Below is the proposed algorithm:

Input: D: a transaction database, minutil: a user − specified threshold

Output: the set of high − utility itemsets

Step 1 : Scan D to calculate the TWU of single items;
Step 2 : $I^* \leftarrow$ each item i such that TWU(i) $<$ minutil;
Step 3 : Let γ be the total order of TWU ascending values on I^*;
Step 4 : Scan D to built the utility − list of each item $i \in I^*$ and build the AUBS structure;
Step 5 : Search (Ø, $I*$, minutil, AUBS);

Fig 1. A novel algorithm of High Utility Itemsets Mining.

- **input** :
  P: an itemset, Extensions of P: a set of extensions of P, the minutil threshold, the AUP structure
- **output**: the set of high − utility itemsets

- foreach itemset P x ∈ ExtensionsOfP do
  - if SUM(P_x. utilitylist. iutils) ≥ minutil then
    - output P_x;
  - end
  - if SUM(P_x. utilitylist. iutils) + SUM(P_x. utilitylist. rutils) ≥ minutil then
    - ExtensionsOfPx ← ∅;
    - foreach itemset P_y ∈ ExtensionsOfP such that y ≻ x do
      - if ∃(x, y, c) ∈ AUBS such that c ≥ minutil) then
        - P_{xy} ← P_x ∪ P_y;
        - P_{xy}. utilitylist ← Construct (P, P_x, P_y);
        - ExtensionsOfP_x ← ExtensionsOfP_x ∪ P_{xy};
      - end
    - end
    - Search (P_x, ExtensionsOfPx, minutil);
  - end
- end

Fig 2 Pruning Strategy Algorithm

Input: P: an Itemset, P_x: the extension of P with an item x, P_y: the extension of P with an item y
Output: the utility − list of P_{xy}
UtilityListOfP_{xy} ← ∅;
Foreach tuple e_x ∈ P_x. utiltylist do
  if ∃e_y ∈ P_y. utilitylist and e_x. tid = e_{xy}. tid then
    if P. utilitylist ≠ ∅ then
      Search element e ∈ P. utilitylist such that e. tid = e_x. tid;
      e_{xy} ← $(e_x.\text{tid}, e_x.\text{iutil} + e_y.\text{iutil} - e.\text{iutil}, e_y.\text{rutil})$;
    End
    Else
      e_{xy} ← $(e_x.\text{tid}, e_x.\text{iutil} + e_y.\text{iutil}, e_y.\text{rutil})$;
    End
      UtilityListOfP_{xy} ← UtilityListOfP_{xy} ∪ $\{e_{xy}\}$
  End
End

Fig. 3 Construct Procedure

4. EXPERIMENTAL WORK AND RESULTS

We performed the testing on 5 data sets from UCI repository. The data sets are accident, chess, pumbsb, mushrooms, retails.
Following table shows execution times and memory consumption of the datasets.

Table 5: Result Analysis (Execution Time Analysis)

Dataset Name: Accident				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Execution time (ms)	Execution time (ms)
1010026201	6730854	200	348954	358652
296858884	1315962	300	170483	194094
71130304	251113	400	112505	167675
31341974	130338	500	95487	127812
23899286	114688	600	99707	103194
21774504	103215	700	84218	85002
21074032	89808	800	87967	94736
20843531	75809	900	81400	83606
20712491	62573	1000	76703	81605

1. Accident Dataset

Dataset Name: Retails				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Execution time (ms)	Execution time (ms)
4715326	164976	200	21840	23425
4085421	127552	300	21778	23269
3945200	77186	400	21684	21852
3626798	39993	500	21466	26646
3064007	20070	600	19640	22058
2334006	11136	700	18688	19170
1617613	7461	800	17082	20329
1036882	5929	900	15600	16665
632786	5382	1000	14882	15870

2. Retails Dataset

Dataset Name: Chess				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Execution time (ms)	Execution time (ms)
158425388	311015	300	84742	86146
62873676	128037	400	49336	56085
35019377	68290	500	35731	36641
21820164	38282	600	28363	29440
14387424	21391	700	23097	23091
9955652	12626	800	19261	19712
7288403	8246	900	17390	18293
5681872	5711	1000	14916	18293

3. Chess Dataset

Dataset Name: Pumsb				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Execution time (ms)	Execution time (ms)
270539846	7585663	20	79416	86697
241434858	5891132	30	71479	74258
188380197	3789518	40	56192	56192
128266953	2082279	50	42589	45676
78569135	1010554	60	30647	35030
45895752	457195	70	25180	28129
27702812	207587	80	19479	19890
18367292	104208	90	19288	19847
13356254	62077	100	16376	16729
10363367	43430	110	15555	15900

4. Pumsb Dataset

Dataset Name: Mushrooms				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Execution time (ms)	Execution time (ms)
117250901	1295377	20	27942	28807
93200126	925787	30	24015	24408
59518065	554707	40	16414	17511
30345690	283773	50	9599	9599
12983561	125890	60	5380	5547
5322994	48658	70	3505	4681
2376007	16790	80	2214	3255
1302291	5405	90	1730	1962
798860	1808	100	1379	1812
535812	860	110	1647	1650

5. Mushrooms Dataset

Table 6: Result Analysis (Memory Occupation Analysis)

Dataset Name: Accident				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Memory	Memory
1010026201	6730854	200	44.11343384	45.30254364
296858884	1315962	300	35.94010925	36.5364151
71130304	251113	400	41.3348999	43.74705505
31341974	130338	500	43.80355072	44.74591827
23899286	114688	600	35.93977356	37.77521723
21774504	103215	700	35.93977356	37.3758316
21074032	89808	800	35.93977356	36.74591827
20843531	75809	900	35.93977356	36.74591827
20712491	62573	1000	35.93977356	36.98177338

1. Accident Dataset

Dataset Name: Retails				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Memory	Memory
4715326	164976	200	28.1545105	29.4540105
4085421	127552	300	26.87570953	27.27700557
3945200	77186	400	20.72692871	26.33441925
3626798	39993	500	33.33231354	30.33231354
3064007	20070	600	26.20373535	29.69876099
2334006	11136	700	23.9582901	25.83473969
1617613	7461	800	25.07624054	20.4046936
1036882	5929	900	21.68154144	24.33560181
632786	5382	1000	23.04615021	23.29427338

2. Retails Dataset

Dataset Name: Chess				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Memory	Memory
158425388	311015	300	7.72845459	7.924464781
62873676	128037	400	4.800796509	5.14151001
35019377	68290	500	13.93788147	14.42671204
21820164	38282	600	11.03689575	12.04354858
14387424	21391	700	9.978912354	11.28218842
9955652	12626	800	11.12553406	12.12864685
7288403	8246	900	11.67134857	11.82299194
5681872	5711	1000	10.36638641	10.83495175

3. Chess Dataset

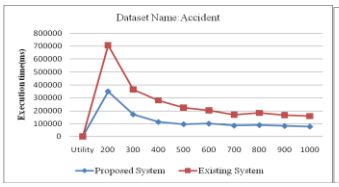
Dataset Name: Pumsb				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Memory(MB)	Memory(MB)
270539846	7585663	20	15.61930847	16.16085052
241434858	5891132	30	16.07501221	16.31917267
188380197	3789518	40	13.7053299	13.95844574
128266953	2082279	50	16.48575592	17.03975677
78569135	1010554	60	13.7053299	14.34442902
45895752	457195	70	16.57945251	16.88559265
27702812	207587	80	13.70309448	13.93407288
18367292	104208	90	13.70309448	13.97099609
13356254	62077	100	13.70309448	13.83698578
10363367	43430	110	13.7053299	13.88599701

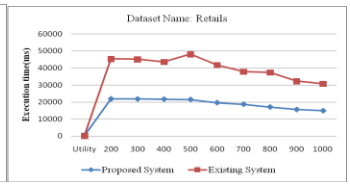
4. Pumsb Dataset

Dataset Name: Mushrooms				
			Proposed System	Existing System
Candidate Count	HUICount	Utility	Memory(MB)	Memory(MB)
117250901	1295377	20	13.33829498	14.82988739
93200126	925787	30	11.25697327	12.12731934
59518065	554707	40	8.578224182	9.4402771
30345690	283773	50	7.4401474	7.4401474
12983561	125890	60	9.510559082	10.03915405
5322994	48658	70	7.614753723	8.352684021
2376007	16790	80	5.979194641	6.473640442
1302291	5405	90	7.970489502	8.596801758
798860	1808	100	6.234306335	6.351547241
535812	860	110	7.966888428	8.351547241

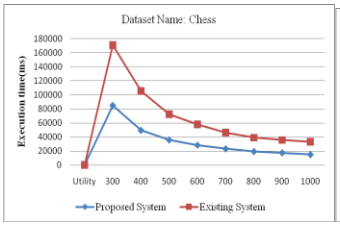
5. Mushrooms Dataset

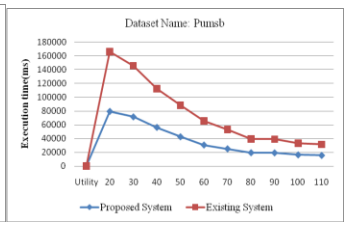
Fig. 5 Result Analysis (Execution Time Analysis)



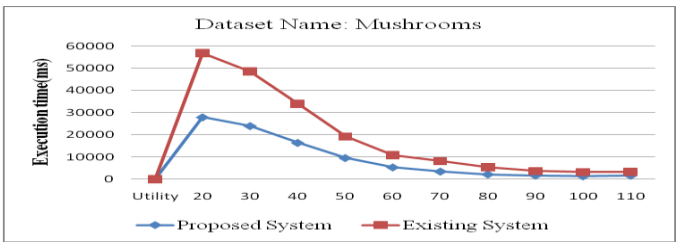


Accident Dataset Retails Dataset



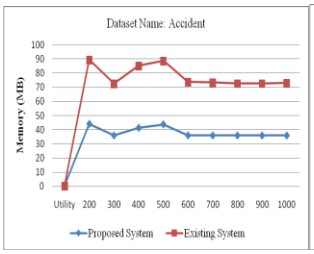


Chess Dataset Pumsb Dataset

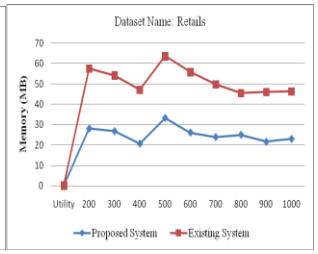


Mushrooms Dataset

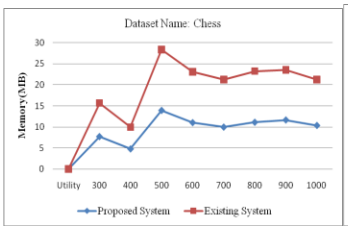
Fig. 6 Result Analysis (Memory Occupation Analysis)



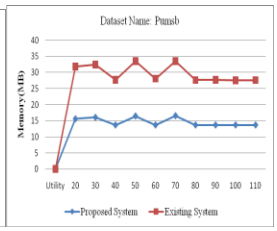
Accident Dataset



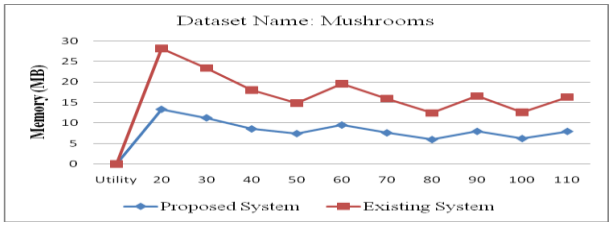
Retails Dataset



Chess Dataset



Pumsb Dataset



Mushrooms Dataset

5.CONCLUSIONS AND FUTURE WORK:

In this paper, we have proposed a novel data structure, utility-list, and developed an efficient algorithm, HUIMiner, for high utility itemset mining. Utility-lists provide not only utility information about itemsets but also important pruning information for HUI-Miner. However, most candidate itemsets are not high utility and are discarded finally. HUI-Miner can mine high utility itemsets without candidate generation, which avoids the costly generation and utility computation of candidates. We have studied the performance of HUI-Miner in comparison with the state of-the-art algorithms on various databases. In future, we compare the performance of this algorithm with other existing algorithms of utility mining.

References

1. Pillai, Jyothi, and O. P. Vyas. "User centric approach to itemset utility mining in Market Basket Analysis." *International Journal on Computer Science and Engineering* 3.1 (2011): 393-400.
2. Bhattacharya, Sudip, and Deepty Dubey. "High Utility Itemset Mining." *International Journal of Emerging Technology and Advanced Engineering* 2.8 (2012): 476-481.
3. Tseng, Vincent S., et al. "Efficient algorithms for mining high utility itemsets from transactional databases." *Knowledge and Data Engineering, IEEE Transactions on* 25.8 (2013): 1772-1786.
4. Londhe, Smita R., Rupali A. Mahajan, and Bhagyashree J. Bhoyar. "Overview on Methods for Mining High Utility Itemset from Transactional Database." *International Journal of Scientific Engineering and Research (IJSER)* 1.4.
5. Murali, Sadak, and Kolla Morarjee. "A Survey A Survey on Efficient Algorithm for Mining High Utility or Mining High Utility Itemsets."
6. Reddy, B. Adinarayana, O. Srinivasa Rao, and M. H. M. Prasad. "An Improved UP-Growth High Utility Itemset Mining." *arXiv preprint arXiv:1212.0317* (2012).
7. Tseng, Vincent S., et al. "Efficient algorithms for mining high utility itemsets from transactional databases." *Knowledge and Data Engineering, IEEE Transactions on* 25.8 (2013): 1772-1786.
8. Shaik, Nazeer, and N. L. Prasanna. "AN EFFICIENT ALGORITHM (FUFM) FOR MINING FREQUENT ITEM SETS."
9. Podpecan, Vid, Nada Lavrac, and Igor Kononenko. "A fast algorithm for mining utility-frequent itemsets." *CONSTRAINT-BASED MINING AND LEARNING* (2007):

10. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Data Bases*, pp. 487-499, 1994.
11. Frequent itemset mining implementations repository, http://fimi.cs.helsinki.fi/
12. Aakansha Sexena,Sohil Gandhiya," A Survey on Frequent Pattern Mining Methods Apriori,Eclat, Fpgrowth",2014 IJDER|Volume 2, issue|ISSN 232-9939
13. Varsha Mashoria,Anju Singh,"Literature Survey On Various Frequent Pattern Mining Algorithm",vol 3,issue1(jan 2013),pp58-64.
14. 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.
15. Yeh J. S., Li, Y. C., Chang C. C.: A Two-Phase Algorithm for Utility-Frequent Mining. To appear in Lecture Notes in Computer Science, International Workshop on High Performance Data Mining and Applications, 2007.
16. Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm," Proc. Utility-Based Data Mining Workshop,2005.
17. Han, Jiawei, Micheline Kamber, and Jian Pei. *Data mining, southeast asia edition: Concepts and techniques*. Morgan kaufmann, 2006.
18. Liu, Mengchi, and Junfeng Qu. "Mining high utility itemsets without candidate generation." *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012.
19. Hong, Tzung-Pei, Cho-Han Lee, and Shyue-Liang Wang. "Effective utility mining with the measure of average utility." *Expert Systems with Applications*38.7 (2011): 8259-8265.
20. H. Yao, H. Hamilton, and C. Butz, "A Foundational Approach to Mining Itemset Utilities from Databases," *The 4th SIAM International Conference on Data Mining*, pp. 211-225, 2004.

