Innovative Science and Technology Publications

International Journal of Future Innovative Science and Technology
ISSN: 2454- 194X Volume - 2, Issue - 2

Manuscript Title

PERFORMANCE ANALYSIS OF DATA MINING TECHNIQUES FOR HIGH UTILITY PATTERNS DISCOVERY

V. Baby
Computer Science and Engineering
Nehru College of Engineering and Research Centre
Pampady, Thiruvilwamala, Thrissur, Kerala
babyvellayudhan@gmail.com

Dr. N. K. Sakthivel
Computer Science and Engineering
Nehru College of Engineering and Research Centre
Pampady, Thiruvilwamala, Thrissur, Kerala
vp@ncerc.ac.in

May – 2016

www.istpublications.com

International Journal of Future Innovative Science and Technology, ISSN: 2454- 194X
Volume-2, Issue-2, May - 2016 editor@istpublications.com

Performance Analysis of Data Mining Techniques for High Utility Patterns Discovery

V. Baby
Computer Science and Engineering
Nehru College of Engineering and Research Centre
Pampady, Thiruvilwamala, Thrissur, Kerala
babyvellayudhan@gmail.com

Dr. N. K. Sakthivel
Computer Science and Engineering
Nehru College of Engineering and Research Centre
Pampady, Thiruvilwamala, Thrissur, Kerala
vp@ncerc.ac.in

ABSTRACT

Discovery of High Utility Itemsets(HUI) or pattern from database is very useful in processing business. By defining a tight upper bound on the utility of candidates more conservative pruning can be achieved. High Utility Pattern growth pruning space by searching a reverse set enumeration tree with utility upper bounding is used in the direct discovery of high utility patterns. User can get concise HUIs by using Closed HUI.

Keywords- High utility pattern, closed high utility itemset, utility mining, lossless and concise representation, pattern mining

1. INTRODUCTION

Utility in high utility means importance, interestingness or profitability of the items or pattern whatever the business need. Comprehension will be very difficult for the users if the algorithm gives a large number of high utility patterns. Candidate pattern's Transaction Weighted Utilization (TWU) [2][5][7] is the transaction's utility sum.

Simplifying the utility calculation and reducing the number of candidates depends the success of high utility pattern mining. The search space can be better pruned by specifying a tighter upper bound. Pruning search space can be done either Pruning Before Candidate Generation (PBCG) or Pruning After Candidate Generation (PACG).

High Utility Pattern (HUP) [2][9][10] finds itemsets in single phase. TWU patterns are not generating in HUP. Original Utility is represented by using CAUL [2] data structure. High Utility itemsets which have the closed itemset [1] property is closed high utility itemset discovery. It finds less number of itemsets than any other algorithms. If we are considering the number of itemset then CHUD [1][6] is best.

2. RECENTLY PROPOSED HIGH UTILITY ITESET MINING TECHNIQUES

Identified recently proposed techniques as the Direct Discovery of High Utility Patterns [2] and the Closed High Utility Patterns [1] and those are the techniques for mining High utility Itemsets.

2.1 ONE SLOT GENERATION OF HUI

A linear list data structure "Chain of Accurate Utility Lists (CAUL) enables the efficient calculation of utility and estimation of tight utility upper bound. It uses reverse set enumeration tree [2]. There will be an imposed ordering Ω [2] in the tree and root does not contain anything, other nodes are labeled by an item. The path from the node to the root node is the pattern. Child nodes of particular node are the items listed before.

Algorithm 1: Direct Discovery of HighUtilityPattern

1 Construct transaction set which contains the Pattern and ordering and external utility
2 Reverse set enumeration tree root
3 DFS(node, transaction set of the pattern, minimum Utility, given ordering)

Subroutine:DFS (node, transaction set of the pattern, Minimum Utility, given ordering)

4 **if** utility of pattern of node≥ minimum utility **then** output pattern of the node

5 W←{i|i<pattern of node and the utility sum of full prefix extension of the transactions(union of {i}and pattern of node)≥ minimum utility
6 **if** closure (pattern of node, W, minimum utility) is Satisfied
7 **then** output nonempty subsets of WUpattern of node
8 **else if** singleton(pattern of node,W, minimum utility) is satisfied
9 **then** output WU pattern of node as a HUP
10 **else foreach** item i ϵ W in Ω **do**
11 **if** basic upper bound ≥minimum utility
12 **then** C← the child node of the current node for i
13 transaction set of pattern of node←project (transaction set of pattern of the current node, i)
14 DFS(C, transaction set(pattern of C, minimum utility,Ω)
15 **end foreach**

Closure is defined as for pattern X and set W of items with X intersection W is a null set, the utility of (S U X) ≥ minimum utility, for all S (is a subset or equal to W) and S is not a null set. If the minimum utility ≤ utility of W union X which is less than the sum of minimum utility and the sum of the utility of the item in w included in the transaction set of X then utility S union X < minimum utility for all S, subset of W. This property is known as Singletone.

For calculating the utilities and upper bound of prefix extension of pattern CAUL is used. It contains two division utility list and summary table. All the items in transaction t which is relevant in growing prefix extension of pattern the utility is stored in the utility list. For each distinct relevant item j to grow the prefix extension of pattern an entry is maintained in the summary table. It is denoted as quintuple, summary[j] = (support[j], utility[j],sum of full prefix extension of {j} union pattern of the current node,

Algorithm 2: PsudoProject(CAUL of pattern of P,i)

1 **foreach** relevant item j < i **do**
2 summary[j]←0
3 **end foreach**
4 **foreach** utility list t threaded by link[i] **do**
5 utility of pattern of N ← utility of pattern of P + utility of i in transaction t
6 calculate sum for all transaction together
7 **foreach** relevant item j belongs to t and j < I by Ω **do**
8 s[j]←s[j]+1
9 u[j]←u[j]+utility of j in t+utility of pattern of node N in transaction t
10 sum ← sum + utility of j in t
11 basic upper bound for j ← basic upper bound for j + sum
12 **end foreach**
13 **foreach** relevant item j belongs to t and j < i by Ω **do**
14 sum of full prefix itemsetof j ← sum of full prefix itemset of j + sum
15 thread t into the chain by link[j]
16 **end foreach**
17 **end foreach**

basic upper bound, link[j]). Basic upper bound for a pattern X is the sum of the utility of the full prefix extension of X with respect to each transaction in the transaction set. In the utility list the occurrences of the same item are linked by a chain threading that is the link[j].

CAUL of a pattern in the transaction set is calculated efficiently by using pseudo projection [4]. In the reverse enumeration tree node N and its parent node P, pattern of N = {i} union pattern of parent. CAUL keeps the original utility information for each transaction. CAUL can determine whether X is a high utility pattern before X is enumerated. Currently being enumerated pattern only keeps in main memory. The optimization is done by considering γ as the maximum number of rounds for irrelevant item filtering and ø for materialization threshold for space-time tradeoff.

2.2 CLOSED HUI DISCOVERY

An itemset is closed if there is no superset which has same support count. A high utility itemset is closed if there should not be any proper superset having the same utility. CHUD is an efficient depth-first search algorithm. It uses Itemset - TidsetpairTree(IT-Tree) [1][8] to find CHUIs. Each node consists of an itemset X, Tidset g(X), two ordered sets of items PREV-SET(X) and POST-SET(X) and estimated utility. The

TU-Table stores the transaction utility with transaction id.

CHUD [1] first scans the database and convert into a vertical database. While creating vertical database, it creates a global TU-Table. Promising items are stored in an ordered list like an increasing order of support at the same time unpromising items are removed from the global TU-Table [1]. From single promising item, CHUD generates candidates by recursively joining items to the existing for forming larger candidates. For each item a_k in O, CHUD creates a node and items a_1 to a_{k-1} into PREV-SET($\{a_k\}$) and items a_{k+1} to an into POST-SET($\{a_k\}$).

ALGORITHM: CHUD

Input: D: the database;absolute minimum utility
Output: complete set of CHUIs

01. **InitialDatabaseScan**(D)
02 **RemoveUtilityUnpromisingItems**(O, GTU)
03 **foreach** item a_k belongs O**do**
04 {Create node for a_k
05 **CHUD_Phase-I**(node for a_k,GTU,absolute minimum utility)
06 **Removing the Exact utilities of items from the Global TU-Table**
07 **CHUD_Phase-II**(D,absolute minimum utility

Phase I produce all the candidates containing the item a_k but no item $a_i < a_k$, then Removing the Exact utilities of items from the Global TU-Table. In Phase II absolute utility of pattern is no less than the absolute minimum utility then the pattern is outputted as CHUI then removes the isolated items of level k. CHUD discovered items are not maintained in the main memory, directly the item is outputted after that it will discard candidates with maximum item utility less than the minimum utility threshold. A candidate can be discarded from Phase II if its estimated utility[1] or maximum item utility is less than the absolute utility [1].

Phase-I of CHUD find the complete set of potential CHUIs. If the set X is not subsumed by other itemsets then closure of X, discarding candidate with the maximum item utility of each calculated. Phase-I explore the superset of X by appending items from POST-SET(X) to X which forms the search space of closed candidates for the second phase. So CHUD gets a very less number of HUIs in Phase-I.

Phase-II gets inputs as promising item's database, potential CHUIs and absolute minimum utility and produces all CHUIs by calculating absolute utility for X and is greater than that of absolute minimum utility then the set X is outputted. In CHUD, the exact utilities of items were removed from the Global TU-Table, the minimum item utility of an item of items were removed from the Local TU Table, the maximum item utility of an item candidates were discarded, these are the strategies used.

3. IMPLEMENTATION OF D2HUP AND CHUD

Techniques D2HUP and CHUD were implemented in Java by using the datasets Foodmart, Mushroom, retail those are described in Table 1.

TABLE 1. DATASET CHARECTERISTICS

Dataset	#Transactions	#Distinct items	Avg.trans. length
Foodmart	4141	1559	4.4
Mushroom	8124	119	23
Retail	5133	16470	13

The datasets used for implementation include Foodmart, Mushroom and Retail as shown in the Table 1. Example for the sparse dataset is Foodmart. Example for dense dataset is Mushroom. Both are real life data sets. Retail is mixed dataset. Algorithms were executed in Windows 8.1 operating system and Intel i3 processor at 1.7GHz. So many datasets like BMS, Connect, Chess, Accident, Kosarak, pumsb utilities have been used for testing. In those D2HUP was taking a long time for output, but CHUD has shown output within seconds. First started running by giving utility values like 10000, 5000, 1000 then identified the maximum utility for each dataset and taken 1%, 10%, 80% of the maximum utility as the minimum utility.

4. RESULTS AND PERFORMANCE ANALYSIS

Table 1. Shows sample outputs got by running with the three real life datasets by setting the minimum utility as 1%, 10% and 80%. If the minimum utility is low, then the CHUD gives a very less number of patterns. Of Mushroom which is a dense data set it gets almost

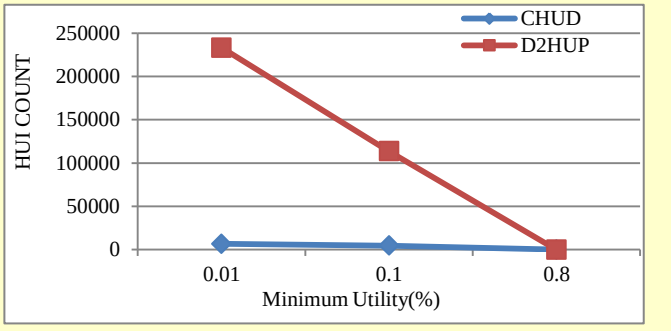
5000 times less number of itemsets than D2HUP. It means that much of repeated utilities with itemsets are available with D2HUP. So if the user wants less number of itemset then CHUD is the best technique. It will be efficient for the requirement of business calculations. When the utility is high, then both algorithms are performing almost equally, with the Mushroom dataset it gives again almost 190 times lesser numbers of itemsets.

The three parameters analyzed with the three datasets include the number of HUI derived, the memory usage and the execution time of the techniques. The memory usage is high

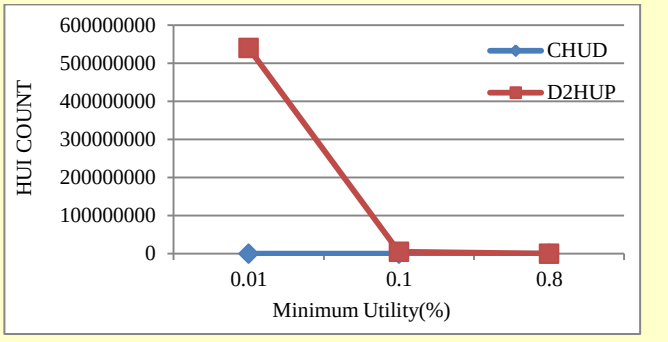
TABLE 2. SAMPLE OUTPUT

Min.Utility	Data set	Technique	Parameters		
			HUI count	*Mem. (MB)*	*Time (Sec.)*
1%	Foodmart	D2HUP	233180	17.5	1.37
		CHUD	6679	13.32	5.9
	Retail	D2HUP	3311	114.23	3.6
		CHUD	3098	239.02	601
	Mushroom	D2HUP	539520109	28.29	1608.19
		CHUD	107103	12.9	52.8
10%	Foodmart	D2HUP	113663	11.2	1.45
		CHUD	4474	7.82	4.7
	Retail	D2HUP	51	50.71	3.73
		CHUD	51	72.62	3.73
	Mushroom	D2HUP	5234758	27.97	124.73
		CHUD	13447	10.57	34.2
80%	Foodmart	D2HUP	11	6.7	0.09
		CHUD	11	14.46	1.6
	Retail	D2HUP	2	18.36	1.04
		CHUD	2	14.4	1.34
	Mushroom	D2HUP	757	24.45	1.18
		CHUD	4	10.79	4.16

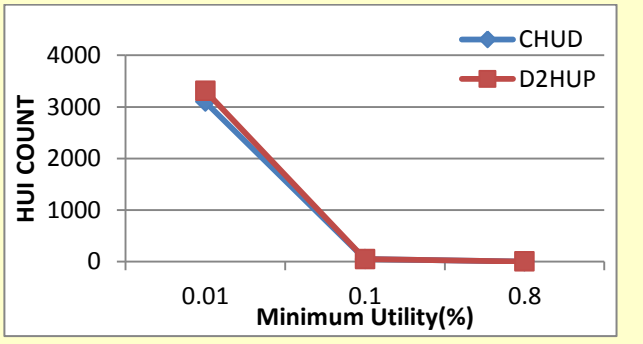
for D2HUP with the Foodmart as well as the Mushroom dataset. Only for Retail CHUD took more memory. When the minimum utility is 1%. For Foodmart when the utility is high that is about 80% CHUD took more memory. For Mushroom CHUD took two times less memory than the D2HUP.Mining itemsets without repetition is very important and it will help the managers reduce the workload for identifying the selling items with more profit. Closed items are less in number and it doesn't contain the repeated itemsets so for finding the unique high utility itemsets CHUD is very useful. The direct discovery of high utility patterns finds more itemsets than Closed HUI discovery. In all datasets CHUD displays the less number or equal to D2HUP. Closed Itemsets will not be having a superset with the same utility that is why CHUD is always retrieving less number of high utility itemsets.



a) Foodmart



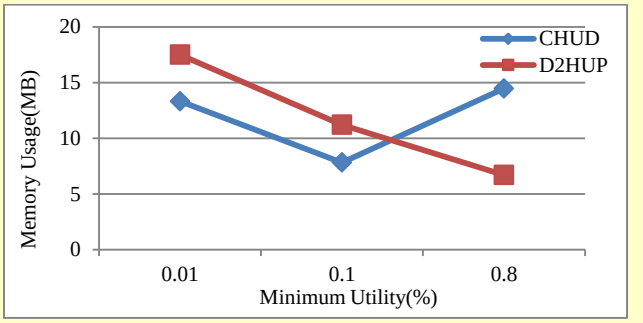
b) Mushroom



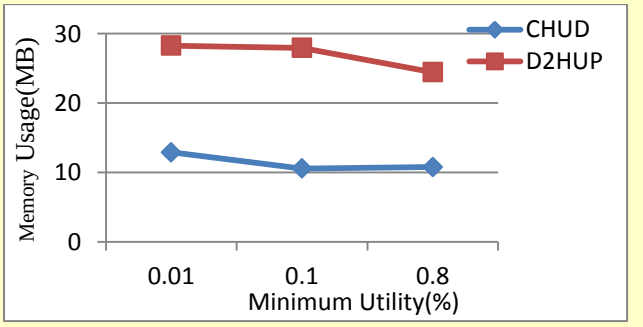
c) Retail

Fig. 1. HUI count vs. Minimum Utility(%)

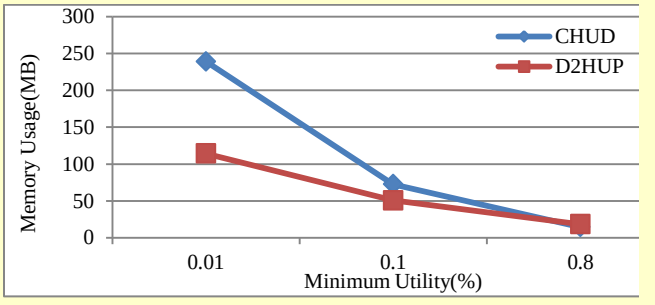
The time taken for the execution is less with CHUD for dense dataset like Mushroom and it is very less for the less value of the minimum utility. For Mushroom CHUD is 30 times faster than D2HUP and for retail D2HUP is performing well than CHUD. If we want to retrieve high utility itemset in the dense large dataset, then for less number of itemsets those contributing more to the profit CHUD is the best solution. For big dataset like BMS, Connect, which have high values as the item utility then also CHUD is the best.



a) Foodmart

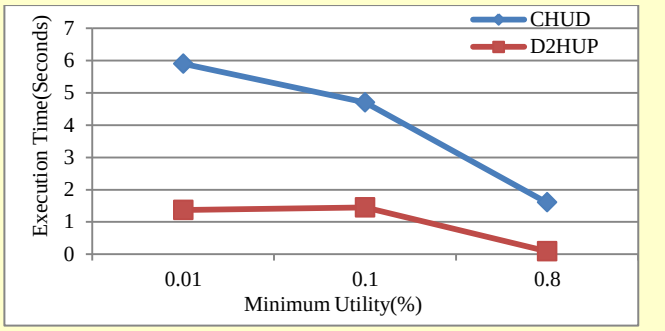


b) Mushroom

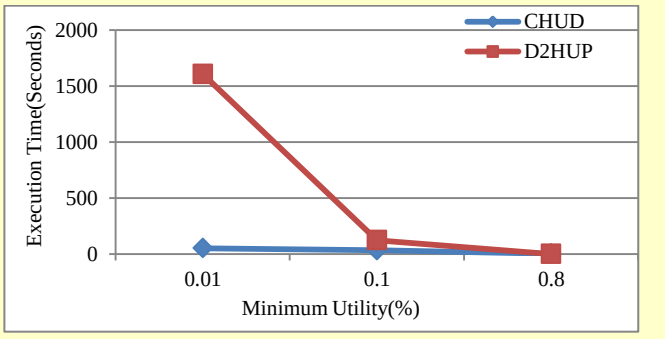


c) Retail

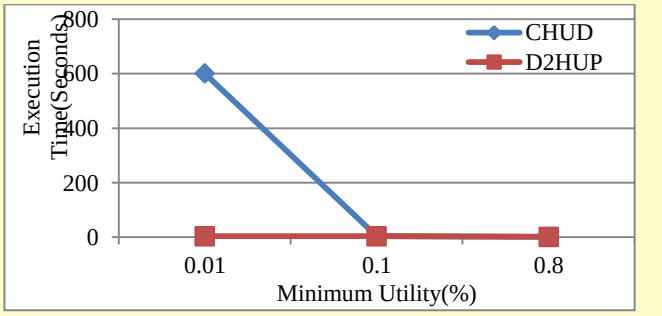
Fig. 2.Memory Usage vs. Minimum Utility(%)



a) Foodmart



b) Mushroom



c) Retail

Fig. 3. Execution time vs. Minimum Utility(%)

5. CONCLUSION AND FUTURE WORK

Mining by specifying the minimum utility is very difficult for the users because users may not be knowing the highest utility so users have to do trial and error in getting the desired result and it is time taking and sometimes they will not get the desired result, so for that user can specify the number of results or the number of patters or itemsets they want for example K. Mining top-k high utility itemsets[3] will be more useful so the future work is for implementing top-k CHUD.

REFERENCES

[1] . Vincent S. Tseng, Cheng-Wei Wu, Philippe Fournier-Viger, and Philip S. Yu, "Efficient Algorithms for Mining the Concise and Lossless Representation of High Utility Itemsets," IEEE Trans. Knowl. Data Eng.,vol. 27, no. 3, pp. 726– 739, Mar. 2015.

[2] . Junqiang Liu, Ke Wang, and Benjamin C.M. Fung, "Mining High Utility Patterns in One Phase without Generating Candidates," IEEE Trans. Knowl. Data Eng, DOI10.1109/TKDE.2015

[3] . Vincent S. Tseng, Cheng-Wei Wu, Philippe Fournier-Viger,and Philip S. Yu, "Efficient Algorithms for Mining Top-K High Utility Itemsets," IEEE Trans. Knowl. Data Eng., DOI10.1109/TKDE.2015.

[4] . G.-C. Lan, T.-P. Hong, and V. S. Tseng, "An efficient projection based indexing approach for mining high utility itemsets,"*KAIS*, vol. 38, no. 1, pp. 85–107, 2014.

[5] . V. S. Tseng, C.-W. Wu, B.-E.Shie, and P. S. Yu, "UP-Growth: An efficient algorithm for high utility itemset mining," in Proc. ACMSIGKDD Int. Conf. Knowl. Discov. Data Mining, 2010, pp. 253–262.

[6] . C.-W Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng, "Efficient mining of a concise and lossless representation of high utility itemsets," in Proc. IEEE Int. Conf. Data Mining, 2011, pp. 824–833.

[7] . V. S. Tseng, B.-E.Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE TKDE*, vol. 25, no. 8, pp. 1772–1786, 2013.

[8] . C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," Expert Syst. Appl., vol. 38, no. 6, pp 7419–7424, 2011.

[9] . M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *CIKM*. ACM, 2012, pp. 55–64.

[10] . J. Liu, K. Wang, and B. Fung, "Direct discovery of high utility Itemsets without candidate generation," in *ICDM*. IEEE, 2012, pp. 984–989.