High utility itemset mining using path encoding and constrained subset generation

Vamsinath Javangula, Suvarna Vani Koneru & Haritha Dasari

Peer-to-Peer Networking and Applications

ISSN 1936-6442

Peer-to-Peer Netw. Appl.
DOI 10.1007/s12083-020-00980-9

ONLINE FIRST

Springer

Peer-to-Peer Networking and Applications
https://doi.org/10.1007/s12083-020-00980-9

High utility itemset mining using path encoding and constrained subset generation

Vamsinath Javangula[1] · Suvarna Vani Koneru[2] · Haritha Dasari[3]

Received: 24 June 2020 / Accepted: 4 August 2020

Abstract
In this paper a two phase approach for high utility itemset mining has been proposed. In the first phase potential high utility itemsets are generated using potential high utility maximal supersets. The transaction weighted utility measure is used in ascertaining the potential high utility itemsets. The maximal supersets are obtained from high utility paths ending in the items in the transaction database. The supersets are constructed without using any tree structures. The prefix information of an item in a transaction is stored in the form of binary codes. Thus, the prefix information of a path in a transaction is encoded as binary codes and stored in the node containing the item information. The potential high utility itemsets are generated from the maximal supersets using a modified set enumeration tree. The high utility itemsets are then obtained from the set enumeration tree by calculating the actual utility by scanning the transaction database. The experiments highlight the superior performance of the system compared to other similar systems in the literature.

Keywords Itemset mining · High utility itemset mining · Mining weighted frequent patterns · Transaction utility mining · Utility mining

1 Introduction

Frequent itemset mining is is one of the generally investigated territories under information mining. It was primarily used for constructing rules in association rule mining [1–3]. In terms of market basket analysis,it is a technique to identify items frequently purchase together [4, 5]. This information can utilized for improving sales of product by formalizing proper marketing strategies [6].

Frequent itemsets are mined using a measure called support which is a measure of the count of the event of itemset in the exchanged performed. The frequent itemset represent items in high volume transactions, it does not convey anything about the profitability of the transactions.

Businesses often are interested in transactions that are highly profitable to them. In order to measure the profitability of the things in a exchange, a unit called utility was in the frequent itemset mining process. This process of mining frequent itemset that exhibit high utility is called HUI Mining. The utility of an itemset is made up of two components - internal and external utility [5, 7].

The problem involved in the mining process is the exponential countof itemsets processing. This is handled by the frequent itemset approaches using the monotonicity property. That is, an itemset has a chance of being frequent only if its subset is also frequent.

In case of HUIM, the high utility of a subset may not be carried over to the superset. That is, in some cases the subset might have a higher utility than its superset and vice versa. This is overcome in HUIM by using a unit called TWU [8]. It is a measure of the frequency of the itemset, thereby ensuring the monotonicity property.

This article is part of the Topical Collection: *Special Issue on Network In Box, Architecture, Networking and Applications*
Guest Editor: Ching-Hsien Hsu

✉ Vamsinath Javangula
vmsn80@yahoo.com

Suvarna Vani Koneru
suvarnavanik@gmail.com

Haritha Dasari
harithadasari9@yahoo.com

[1] CSE Department, P.B.R V I T S, Kavali, Andhra Pradesh, India

[2] CSE Department, Velagapudi Ramakrishna Siddhartha Engineering College, Kanuru, Andhra Pradesh, India

[3] CSE Department, University College of Engineering College, JNTUK, Kakinada, Andhra Pradesh, India

The approaches can be grouped under two main categories – two phase approaches [8–14] and one phase approaches [15–17]. As their names suggest, these two approaches differ in the number of phases used in identifying high utility itemsets. The two phase approach splits the process in to two stages – one for identifying potential itemsets and the second stage to pick the actual high utility itemsets from the potential itemsets identified from the earlier stage.

The two phase approach is sometimes overwhelmed by the huge volume of potential high volume itemsets generated. The single phase approaches to overcome this problem by identifying the high utility itemsets in a single phase itself. Here, an method has been proposed that builds maximal supersets without any pattern tree structure. The potential high utility itemsets are generated from the maximal supersets by constrained subset generation process using set enumeration tree.

This area is trailed by the writing study of frameworks produced for mining high utility itemsets. The following segment depicts the proposed framework. This is trailed by the exploratory assessment of the proposed framework. The last area gives the finish of this paper.

2 Literature survey

An overview of a portion of the best in class frameworks in high utility itemset mining issue space is given. In [18] a level wise itemset mining set of rules is proposed to mine high utility itemsets. In order to prune low utility items a novel pruning strategy with antimonotonic property is also proposed. The advantage in this system is that it does not require specification of minimum utility limit and is capable of mining top K high utility itemsets based on a given business strategy. [12] propose a highly efficient two phase algorithm that is capable of identifying high utility itemsets by applying a novel pruning strategy.

In [11] an efficient strategy for pruning candidate space called Isolated Item Discarding strategy is proposed. In [9] a novel strategy for performing high utility pattern mining incrementally and interactively is proposed. This approach makes use of three types of novel tree structures to facilitate the mining process – a lexicographic tree for arranging items in problem domain in lexicographic order, transaction frequency tree and a transaction weighted utilization tree. [14] propose a novel data structure –utility pattern tree for efficiently mining utility patterns.

In [13] two algorithms UPGrowth and UPGrowth+ have been proposed to efficiently mine utility pattern tree for high utility itemsets. In [10] a novel indexing mechanism has been proposed. The indexing approach helps in faster execution and utilization of lesser memory for processing. A pruning strategy to reduce the problem space has also been proposed. [17] propose a novel data structure – Maximum Item Quantity tree for processing high utility itemsets. An efficient algorithm Maximum Utility growth and pruning strategies have also been discussed for efficiently mining the maximum item quantity tree. In [19], HUIs which are more frequent are mined, using a threshold support, cutoff utility and suffix utility measures.

[16] propose a novel pruning strategy that greatly reduces the candidates generated in the mining process. It has been showed that the pruning strategy is very effective in handling sparse transactional databases. A length constraint for itemsets is proposed by [15]. In order to utilize length constraint a novel approach Length Upper Bound Reduction has been proposed. It has been shown that following this approach the length of pattern generated and the execution time has decreased considerably. In [4] a closed HUI mining system has been proposed which utilizes novel pruning methods such as chain estimated utility co-occurrence pruning, lower branch pruning, and pruning by coverage for better mining performance.

A novel utility measure along with a new data structure – pset and a new algorithm has been explored in [20] for mining HUIs in databases with item sets displaying dynamic profits. [21] propose a parallel version of the FHM+ high utility itemset miner. Though considerable research has been done in the problem domain, certain issues such as subset generation, downward closure methods for mining utility etc. still persist and needs further research.

3 Path encoded constrained subset generation approach (PECSSG)

In this paper, a two phase utility mining approach called path encoded high utility itemset mining is described. The explosion of potential high utility itemset in the mining process is contained by lazy generation of the itemsets. In the first phase rather than generating the potential high utility itemsets, potential high utility maximal supersets are generated. From the maximal supersets the HUIs are generated using a constrained subset generation process. The process flow in PECSSG is shown in Fig. 1.

To illustrate the concepts involved in the mining process, an example transaction database widely used in the literature is used as a running instance throughout this paper (Table 1).

In the Table 1 a sample database with five transactions is shown [14]. The column represents the transaction ID. The II column represents the transaction. Each alphabet in the exchange represents an item and the number following it denotes the frequency of that item in the transaction. As instance in transaction T4, 4 items of B, 3 items of C, 3 items of D and 1 item of E has been purchased. The last column represents the transaction utility i.e. the profit accrued out of the transaction.

The transaction utility (TU) is obtained by adding up the profits obtained from each of the items of the transaction. In order to calculate the TU, the profit per item is required. This is given in Table 2. For example, the profit per item obtained by transacting

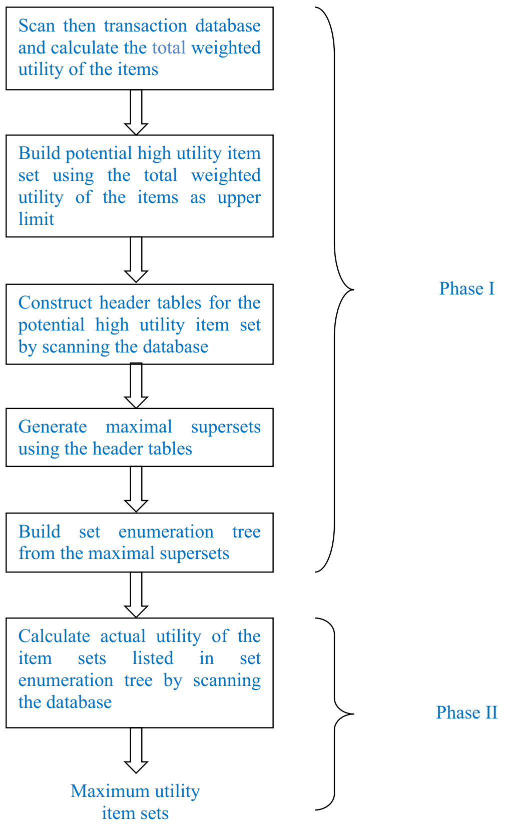
Scan then transaction database and calculate the total weighted utility of the items
Build potential high utility item set using the total weighted utility of the items as upper limit
Construct header tables for the potential high utility item set by scanning the database
Generate maximal supersets using the header tables
Build set enumeration tree from the maximal supersets
Phase I
Calculate actual utility of the item sets listed in set enumeration tree by scanning the database
Phase II
Maximum utility item sets

Fig. 1 Process Flow in PECSSG

item T is 5 units and for item X is 3 units. Thus, the utility of an item in a transaction is obtained by multiplying the profit per item with its frequency.

Problem definition Let D = {t_1, t_2, t_3, …, t_n} be a transaction database with n transactions. Let I = {i_1, i_2, i_3, …, i_m}be a finite set of items involved in the transactions in D.

Definition 1 Internal Utility U_I(i, t) = freq(i, t). For example, in the example transaction database, the internal utility of item E in transaction R2 is, U_I(E, T2) = 2.

Definition 2 External Utility U_E(i) = profit(i). For example, U_E(A) = 5.

Definition 3 Net utility is given by U(i, t) = U_E(i) x U_I(i, t). For example, U(D, T3) = U_E(D) x U_I(D, T3) = (2 × 6) = 12

Table 1 Example transaction database

TID	Transactions	Transaction Utility (TU)
R1	{T:1, V:1, W:1}	8
R2	{T:2, V:6, X:2, Z:5}	27
R3	{T:1, U:2, V:1, W:6, X:1, Y:5}	30
R4	{U:4, V:3, W:3, X:1}	20
R5	{U:2, V:2, X:1, Z:2}	11

Table 2 Profit per item table

Items	T	U	V	W	X	Y	Z
Profit/Item	5	2	1	2	3	1	1

Definition 4 Utility of an item i in the database D is given by U_D(i) as in Eq. 1,

$$U_D(i) = \sum_{t \in D} U(i,t) \tag{1}$$

For example, U_D(G) = U(G, R2) + U(G, R5) = (1 x 5) + (1 x 2) = 7

Definition 5 An itemset S is a set of items in I such that |S| < = |I|. If the number of items in S is |S| = k, then S is called as a k-itemset. For example, {T, V, X} is a 3-itemset.

Definition 6 Utility of an itemset S in D is given by U_D(X) as in Eq. 2,

$$U_D(S) = \sum_{i \in S \wedge S \subseteq t \wedge t \in D} U(i,S) \tag{2}$$

For example, U_D({X, Y}) = U({X, Y}, T3) = 8.

Definition 7 For an itemset S if the utility of the itemset is greater than or equal to minUtility, where minUtility is a user specified utility threshold, the itemset is called a high utility itemset.

Since the number of itemsets to be generated has exponential complexity, to prune the search space a measure transaction weighted utility, TWU was proposed in [8].

Definition 8 Transaction utility is given by TU(t) as in Eq. 3,

$$TU(t) = \sum_{i \in t} U(i,t) \tag{3}$$

For example, the transaction utility of R1 is the sum of the transaction utilities of the items T, V, and W i.e. 5 + 1 + 2 = 8.

Definition 9 The transaction weighted utility is given by TWU_D(S) as in Eq. 4,

$$TWU_D(S) = \sum_{S \subseteq t \wedge t \in D} TU(t) \tag{4}$$

For example, TWU_D({X, Z}) = TU(T2) + TU(T5) = 27 + 11 = 38.

Table 3 TWU of the items in example database in decreasing order

Items	T	U	V	W	X	Y	Z
TWU	65	61	96	58	88	30	38

Table 4 revised TWU table for minUtility = 40

Items	V	X	T	U	W
TWU	96	88	65	61	58

3.1 Phase one

The main stage contains ascertaining the TWU of everything in the exchange and utilizing it to manufacture the potential high utility maximal supersets. This includes two outputs of the exchange database. The principal output of the database is done to figure the TWU of the things in the exchange. At that point, the things whose TWU esteems are less the base utility limit is expelled from the TWU table and the table is arranged in the diminishing request of TWU values. For instance, the TWU of the things in the model exchange database is indicated Table 3 and the reconsidered and arranged TWU table for least utility limit of 40 is appeared in Table 4.

The TWU of an item in Table 3 is calculated as given by Eq. 4 in Definition 9.

As can be seen in Table 4 items Y and Z has been removed as its TWU 30 and 38 are less than the threshold value of 40. The second scan of the database is done to build the potential high utility maximal supersets. In order to build the maximal supersets, a data structure similar to the header table as in [14] is used. The header table is indexed by the items whose TWU is greater than or equal to the user specified utility limit. Each row in the table is used for storing the itemsets ending with item indexing that row and their TWU. Unlike the other approaches discussed in the literature [14], the header table is built without constructing any pattern tree.

The prefix information of a particular item is stored as a bitcode [22] which is later used for constructing the maximal supersets. In order to facilitate the construction of the header table, each transaction in the database is modeled as a path of nodes, where each node in the path contains information about the items making up the transaction. The information contained in each node is as follows:

- Item name
- TWU of the itemset until that position of the item in the transaction
- Prefix: bitcode representing the prefix information of the itemset until that position of the item in the transaction.

The algorithm for constructing the header table is shown in Algorithm 1.

```
Algorithm 1: Construction of Header table
Inputs: D: Transaction database file
        TWU: Transaction weighted utility table
        minUtility: Minimum utility threshold
Output: htab: Header table
1. htab.index = TWU.items
2. for each t in D do
3.      pathList = empty list
4.      for each i whose TU(i) >= minUtility in t do
5.              node = new info_node(item)
6.              node.name = item
7.              add node to pathList
8.      sort pathList in descending order of TWU.items
9.      prefix = zeros
10.     for each node in pathList do
11.             index  = position of node.item in TWU
12.             prefix[index] = 1
13.             node.prefix = prefix
14.             if node not in htab[node.item] then
15.                     add node to htab[node.item]
16.             else
17.                     htab[node.item].nodeUtility += TU(t)
```

Each of the transaction in D is read and the items whose TU is greater than the minUtility in the transaction are stored as nodes. The list of nodes is then sorted in decreasing order of its TWU. The list of nodes generated corresponding to transaction R1 is shown Fig. 2.

The prefix attribute of the node conveys information regarding the items traversed so far in the current transaction. This is encoded using bitcodes as in [22]. The bitcode is single dimensional bit array having a length same as the size of the revised TWU table. The bitcode of an item is generated by putting a 1 in the bit array corresponding to the position of the item in the revised TWU table. The bitcodes for the items in the example transaction are shown in Table 5.

The bitcode for item X is '01000' (Table 5) as X is positioned 2nd in the revised TWU table. The prefix attribute of the current node is obtained by copying the prefix of the immediate predecessor in the transaction and putting a 1 in the corresponding position of the current item in the revised TWU table – path encoding. Thus path encoding helps us in identifying the nodes traversed before reaching the last item in the item set being processed. For example,

R1: V, T, W

V.prefix = 10000, since C is the first item in R1

T.prefix = 10100, i.e. put 1 in the position of T in C.prefix

W.prefix = 10101, i.e. put 1 in the position of W in A.prefix.

Thus by perusing the prefix of a node, information regarding the current item as well as the previous items in the transaction can be obtained. These nodes are then added to the

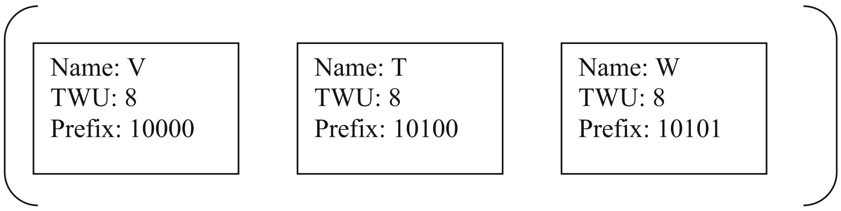

Fig. 2 Nodes generated from R1 in D

Table 5 Bitcodes of items whose TWU > = 40 in D

Items	Bitcode				
	V	X	T	U	W
V	1	0	0	0	0
X	0	1	0	0	0
T	0	0	1	0	0
U	0	0	0	1	0
W	0	0	0	0	1

header table indexed by the item names in D. The first time an item is encountered, the node generated is added to the header table. The next time it is encountered, the nodes TWU value is incremented by its current TWU value. That is,

For R1: V, V.TWU = 8.

For R2: V, V.TWU = 8 + 27 = 35.

Thus each time an item is encountered its TWU value is updated accordingly. The final header table obtained after scanning all the transactions in D is shown in Table 6.

As can be seen from the header table, each entry in the table corresponds to an item and the itemsets ending with that item. For example, the entry for item V shows that there is only one itemset ending in V as it is the item with the maximum TWU. The entry for T shows that there are two itemsets ending in T {V,T} and {V, X, T}. The constructed header table is then used for identifying the potential high utility maximal supersets. The algorithm for identifying the maximal supersets is shown in Algorithm 2.

```
Algorithm 2: Identification of potential high utility maximal supersets
Inputs: TWU: Transaction weighted utility table
        htab: Header table
        minUtility: Minimum utility threshold
Output: phms: Potential high utility maximal supersets
1.  phms = empty
2.  for each itemI in reverse(TWU.items) do
3.      nodeList = htab[itemI]
4.      ms = empty
5.      for each itemJ in TWU.items do
6.          if itemJ in pms then
7.              add itemJ to ms
8.          else
9.              nodes = empty
10.             for each node in nodeList do
11.                 if node.prefix contains itemJ then
12.                     add node to nodes
13.             if sum(nodes.utility) >= minUtility then
14.                 add itemJ to ms
15.     add ms to phms
```

Table 6 Header table for transactions in D with minUtility = 40

Index	Node list [item name, TWU, prefix]
V	[V, 96, 10,000]
X	[X, 88, 11,000]
T	[T, 8, 10,100], [T, 57, 11,100]
U	[U, 30, 11,110], [U, 31, 11,010]
W	[W, 8, 10,101], [W, 30, 11,111], [W, 31, 11,011]

Table 7 Maximal superset

Item	Maximal Superset
V	The node V itself
X	Node V and node X
T	Node V, Node X, Node T
U	Node V, Node X, Node U
W	Node V, Node X, Node U and Node W

Definition 10 An itemset starting with an item i_j and ending with an item i_k and TWU($\{i_j \ldots i_k\}$) > = minUtility is called an maximal superset of i_k if there exists no superset of $\{i_j \ldots i_k\}$ ending with i_k whose TWU > TWU($\{i_j \ldots i_k\}$).

The maximal superset of an item i_k is obtained by forming itemsets by adding all i_j's before i_k such that TWU(i_j) > = TWU(i_k) and TWU($\{i_j, i_k\}$) > = minUtility. The items are processed in the increasing order of the TWU values for forming the maximal superset i.e. the item with the least TWU is chosen first. This is obtained by perusing the prefix patterns of the node list in the header table of the corresponding item. For example, the nodelist corresponding to item D in the header table are as follows:

D -> [W, 8, 10101], [W, 30, 11111], [W, 31, 11011]

The TWU({V,W}) can be obtained by summing the TWU of nodes whose prefix has a 1 in the position of V. That is,

TWU({V, W}) = 8+30+31= 69 > 40, likewise

TWU({X, W}) = 30 + 31=61 > 40

TWU({T, W}) = 8 + 30 = 38 < 40

TWU({U, W}) = 30 + 31= 61 > 40

So the maximal superset of W = {V, X, U, W}.

When calculating the maximal superset of next item U in order, the items in the maximal superset of W coming before U can be added to the maximal superset of U by default. Hence the only item to be checked is T.

U -> [U, 30, 11110], [U, 31, 11010]

TWU({T, U}) = 30 < 40

So the maximal superset of U = {V, X, U}.

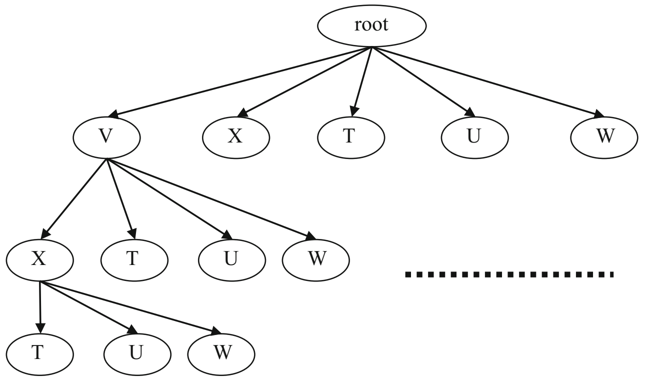

Fig. 3 A portion of the enumeration tree of maximal supersets in D

Table 8 Dataset and their characteristics

Dataset	No. of Transactions	No. of Items	Average transaction Length
Chess	3196	75	36
Mushroom	8124	120	23
Retail	88,162	16,470	10.3
Accidents	340,183	468	33.8

The identified maximal supersets are then used for generating the subsets using a set enumeration tree as in [22]. The subset enumeration procedure has been modified to operate with multiple maximal supersets – constrained subset generation. The revised procedure is capable of combining mulitiple supersets in to a single enumeration tree. The algorithm for subset generation is shown in Algorithm 3.

```
Algorithm 3 Enumeration tree construction
Inputs: phms: Potential high utility maximal supersets
Output: pseq: Potential sequences
1. pseq = empty
2. root = seqnode()
3. root.name = -1
4. root.children = empty
5. root.itemset = empty
6. current = root
7. for each ms in phms do
8.      for each item in ms do
9.           if item not in current.children then
10.               child = seqnode()
11.               child.name = item
12.               add item to child.itemset
13.               add item to pseq
14.               add child to current.children
15.               ms = ms – item
16.               subsetgen(child, ms, pseq)
17. # subsetgen: sunset generation sub procedure
18.Parameters: Node: sequence node
19.          ms: maximal superset passed from main procedure
20.          pseq: Potential sequences
21.par.itemset = Node.itemset
22. for each item in ms do
23.      if item not in Node.children then
24.           child = seqnode()
25.           child.name = item
26.           add par.itemset + item to child.itemset
27.           add child to Node.children
28.           ms = ms – item
29.           subsetgen(child, ms, pseq)
```

The maximal supersets generated for the transactions in D are shown in Table 7. A part of the enumeration tree generated for the supersets in Table 7 is shown in Fig. 3.

For example, as can be seen in Table 7, the maximal superset of T is {V, X, T} generated as defined in Definition 10 earlier.

The nodes of the enumeration tree contain the following information:

- Name: item name
- Children: The children of the node
- Itemset: set of items in the path of the node

Thus, itemsets that are part of the maximal superset can be obtained directly from the enumeration tree. For example, node T at the third level of the enumeration tree represents the itemset {V, X, T} i.e. the set of items from root to T.

3.2 Phase two

In this stage, the database is filtered for the third an ideal opportunity to figure the genuine utility of the potential high utility itemsets distinguished in stage one from the list tree. The genuine utility of the itemsets are contrasted and the base utility edge and

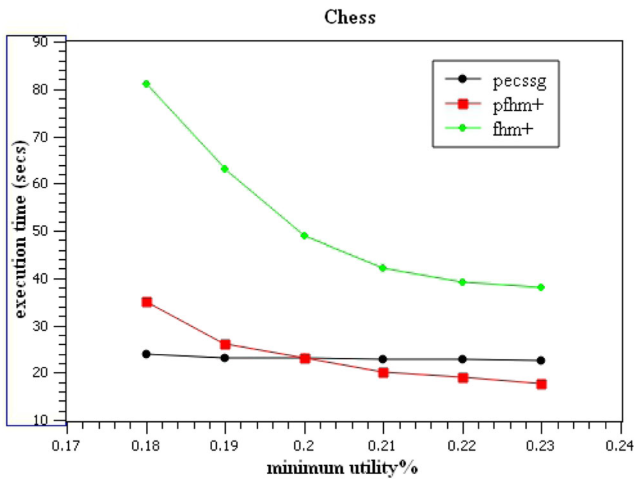

Fig. 4 System performance with respect to chess dataset

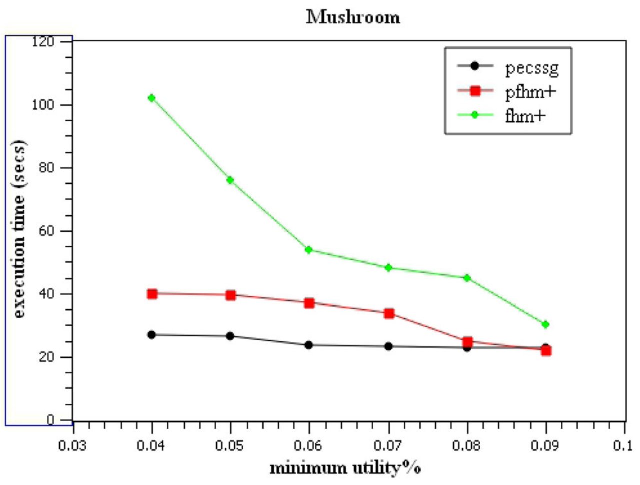

Fig. 5 System performance with respect to mushroom dataset

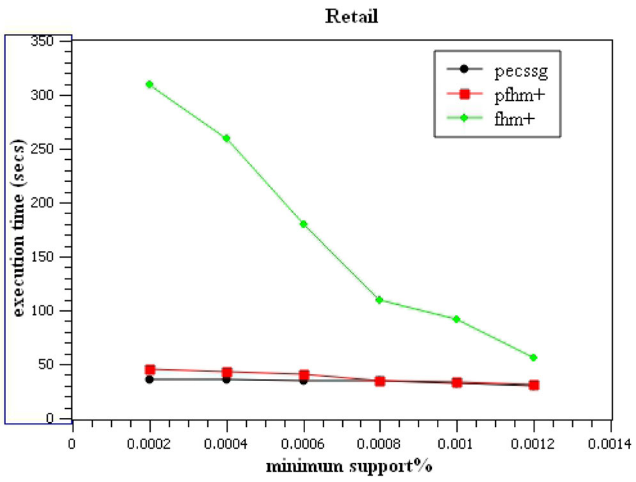

Fig. 6 System performance with respect to retail dataset

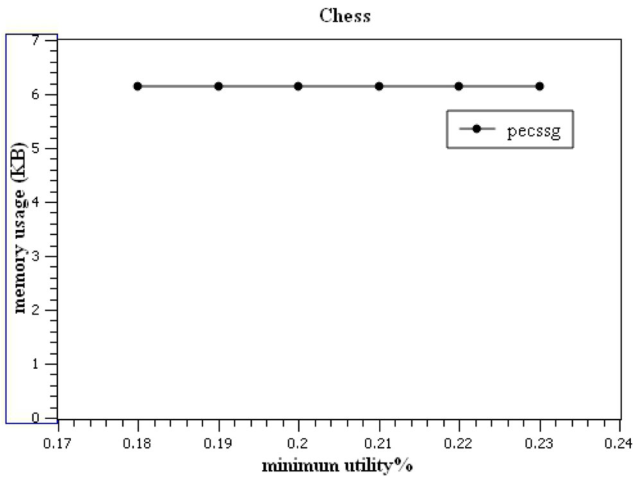

Fig. 8 Memory usage for chess dataset

the HUIs are distinguished. The calculation for distinguishing the HUIs is appeared in Algorithm 4.

```
Algorithm 4 Identification of HUIs
Inputs: phui: Potential high utility itemsets
        minUtility: Minimum utility threshold
Output: hui: High utility itemsets
1. hui = empty
2. for each itemset in phui do
3.      for each tr in filename do
4.           calculate the actual utility of the itemset
5.      if itemset.utility >= minUtility then
6.           add itemset to hui
```

4 Experimentation and results

The experimentation was conducted using four real time datasets – chess, mushroom, retail and accidents. The dataset and their characteristics are shown in Table 8. The experimentation was performed using intel core i3 system with 3GB RAM. The results from the experimentation were compared with two current state of the art systems in the literature - pfhm+ and fhm + as they had outperformed other similar systems discussed in the literature.

The performance of the proposed system (path encoded constrained subset generation – pecssg) with respect to the datasets is shown in Fig. 4 to Fig. 7. The superior performance of the system with respect to datasets of diverse characteristics establishes the robustness and scalability of the proposed system. It was found that the system performed relatively well with respect to the current state of the art systems in the literature.

In Figs. 4 & 5, it can be seen that the proposed strategy pecssg has completely outperformed other state of the art systems with an execution time of less than 30 s for both chess and mushroom dataset.

In Figs. 6 & 7, the proposed strategy pecssg has excelled with great margin with respect to fhm + and by a small margin with respect to pfhm+ for both retail and accident datasets. Overall for all four datsets the proposed strategy pecssg is better than the other state of the art systems.

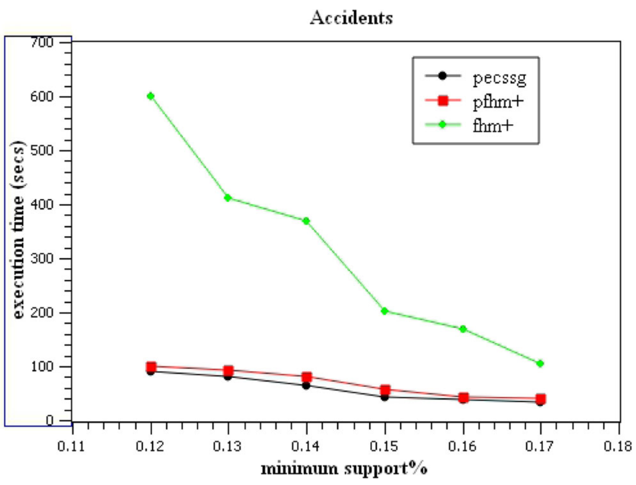

Fig. 7 System performance with respect to accidents dataset

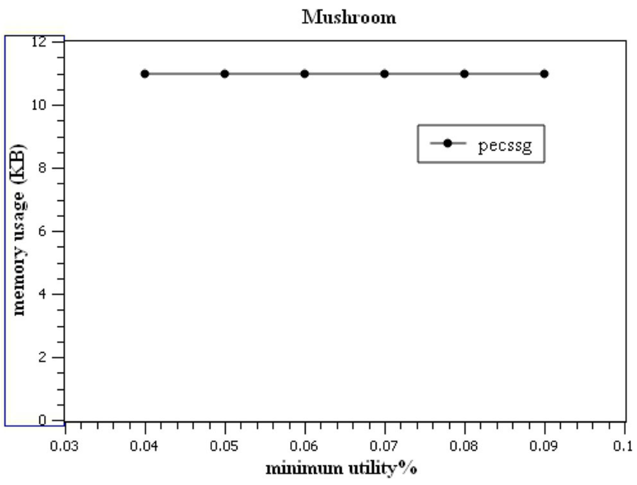

Fig. 9 Memory usage for mushroom dataset

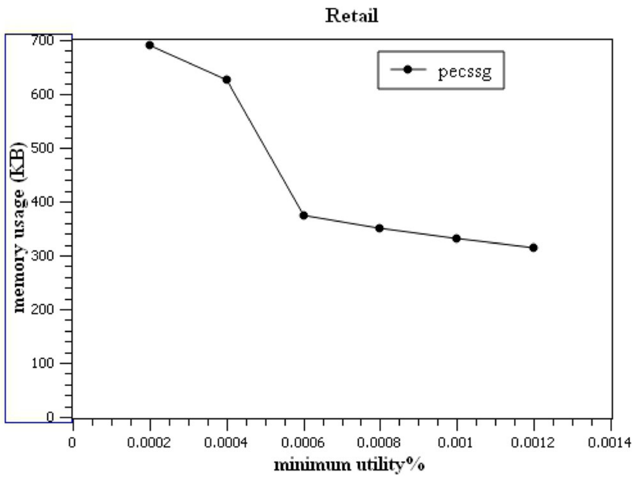

Fig. 10 Memory usage for retail dataset

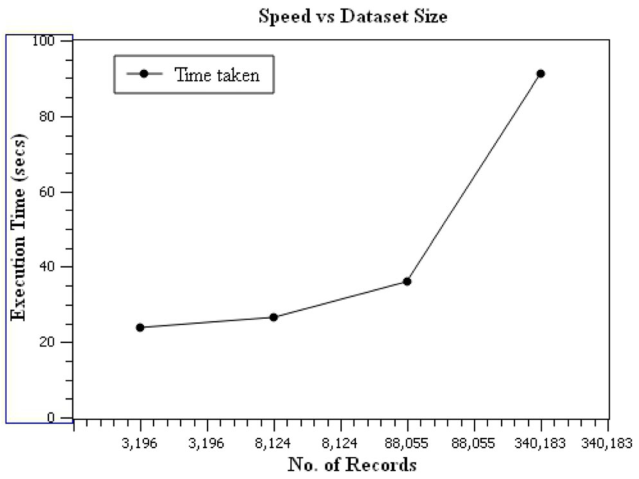

Fig. 12 Variation of Execution time with respect to Dataset Size

The memory usage by the algorithm for various minimum utility values for each of the datasets is shown in Figs. 8, 9, 10 and 11.

It can be seen that for most of the datasets Figs.8 to 11, the memory usage is almost constant for the range of utilities tested. The only exception to this is the retail dataset for which the memory usage is almost linear with respect to the range of utilities tested. Thus, the memory usage degrades gracefully with changes in the utilities.

The graphs in Fig. 12 and Fig. 13 exhibit the performance of the system with respect to increase in data size.

As can be seen for Fig. 12 and Fig. 13, the variation in memory used and execution time too is almost linear to the dataset size. The performance degradation if any with respect to dataset size is very gradual.

5 Conclusion and future work

Here, a high utility itemset mining approach using a header table of maximal superset has been discussed. It was shown that the maximal supersets can be constructed without using any tree structure. The necessary prefix information was stored as bit code in the nodelist representing a transaction. It was also shown how the bitcodes helped in the construction of maximal supersets.

Using a constrained subset generation approach, potential high utility itemsets was generated from the maximal supersets using a set enumeration tree structure. Its construction was modified to build a single set enumeration tree from multiple maximal supersets. Our proposed method highlights the enhanced performance of the framework compared to other similar frameworks in the literature. The results obtained exhibit the superior performance of the system over pfhm+ and fhm + strategies that had outperformed the other state of the art systems. In future, pruning strategies and more efficient set enumeration procedures could be experimented with, to improve the presentation of the framework further.

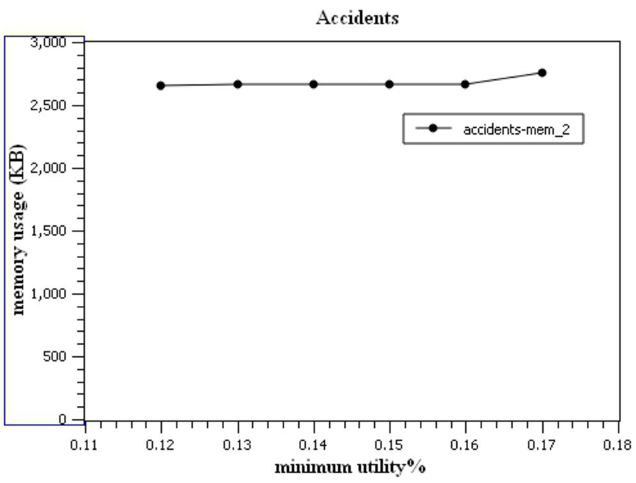

Fig. 11 Memory usage for accidents dataset

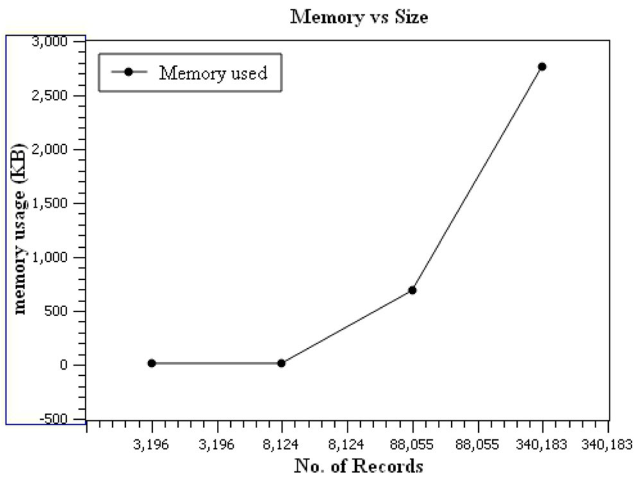

Fig. 13 Variation of memory used with respect to Dataset size

References

1. Agrawal R, Srikant R (1994) “Fast algorithms for mining association rules.” In Proc. 20th int. conf. very large data bases. VLDB 1215: 487–499
2. Agrawal, R., Imieliński, T., & Swami, A. (1993). “Mining association rules between sets of items in large databases.” In Acm sigmod record 22 (2). ACM: 207–216
3. Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques. Elsevier, New York
4. Dam T, Li K, Fournier-Viger P et al (2019) CLS-miner: efficient and effective closed high-utility itemset mining. Front Comput Sci 13: 357–381. https://doi.org/10.1007/s11704-016-6245-4
5. Han J, Cheng H, Xin D, Yan X (2007) Frequent pattern mining: current status and future directions. Data Min Knowl Disc 15(1): 5586
6. Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min Knowl Disc 8(1):53–87
7. Sethi, K. K., & Ramesh, D. (2017). “HFIM: a Spark-based hybrid frequent itemset mining algorithm for big data processing.” The Journal of Supercomputing: 1–17
8. Yao, H., Hamilton, H. J., & Butz, C. J. (2004). “A foundational approach to mining itemset utilities from databases.” In Proceedings of the 2004 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics: 482–486
9. Ahmed CF, Tanbeer SK, Jeong BS, Lee YK (2009) Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans Knowl Data Eng 21(12):1708–1721
10. Lan GC, Hong TP, Tseng VS (2014) An efficient projection-based indexing approach for mining high utility itemsets. Knowl Inf Syst 38(1):85–107
11. Li YC, Yeh JS, Chang CC (2008) Isolated items discarding strategy for discovering high utility itemsets. Data Knowl Eng 64(1):198–217
12. Liu Y, Liao WK, Choudhary AN (2005) A two-phase algorithm for fast discovery of high utility Itemsets. In PAKDD 3518:689–695
13. Tseng VS, Shie BE, Wu CW, Philip SY (2013) Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans Knowl Data Eng 25(8):1772–1786
14. Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010). “UP-Growth: an efficient algorithm for high utility itemset mining.” In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining ACM: 253–262
15. Fournier-Viger, P., Lin, J. C. W., Duong, Q. H., & Dam, T. L. (2016). “FHM+: faster high-utility itemset mining using length upperbound reduction.” In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer International Publishing: 115–127
16. Krishnamoorthy S (2015) Pruning strategies for mining high utility itemsets. Expert Syst Appl 42(5):2371–2381
17. Yun U, Ryang H, Ryu KH (2014) High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates. Expert Syst Appl 41(8):3861–3878
18. Chan, R., Yang, Q., & Shen, Y. D. (2003). “Mining high utility itemsets.” In Data Mining ICDM Third IEEE International Conference on IEEE: 19–26
19. Uday KR, Yashwanth RT, Fournier-Viger P, Toyoda M, Krishna RP, Kitsuregawa M (2019) Efficiently Finding High Utility-Frequent Itemsets Using Cutoff and Suffix Utility. In: Yang Q, Zhou ZH, Gong Z, Zhang ML, Huang SJ (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2019. Lecture notes in computer science, vol 11440. Springer, Cham
20. Nguyen LT, Nguyen P, Nguyen TD, Vo B, Fournier-Viger P, Tseng VS (2019) Mining high-utility itemsets in dynamic profit databases. Knowl-Based Syst 175:130–144
21. Sethi KK, Ramesh D, Edla DR (2018) P-FHM+: parallel high utility itemset mining algorithm for big data processing. Procedia Comput Sci 132:918–927
22. Arybarzan N, Bidgoli B, Reshnehlab M (2018) negFIN: an efficient algorithm for fast mining frequent itemsets. Expert Syst Appl 105: 129–143

Vamsinath Javangula is currently working as Asst. Professor, Dept of C.S.E, P.B.R V I T S College. He is having nearly 15 years of teaching experience. He published 15 papers in various conferences and journals. Areas of interest are Data Mining, Deep learning, Machine learning.He is currently Pursuing Ph.D in Department of CSE, JNTU Kakinada.

Suvarna Vani Koneru is working as Professor in Department Computer Science and Engineering,Velagapudi Ramakrishna Siddhartha Engineering College and 20+ experiences in Teaching .Her areas of interest are Bio-informatics, Machine learning, Deep learning.

Haritha Dasari is working as Professor in Computer science and Engineering Department at JNTU Kakinada. She has 19+ years of experience. Her research interest is on Image Processing, Data Structures, Software Engineering and Networking.