RESEARCH ARTICLE OPEN ACCESS

MVEMFI: Visualizing and Extracting Maximal Frequent Itemsets

Maha Attia Hana*
*(Department of Electrical and Communication Department Canadian International College ElSheikh Zaid, Deputed from Faculty of computers & Information, Helwan University, Egypt

ABSTRACT
Association rule is a data mining technique that has a huge number of applications. One of the crucial steps in association rule is the extraction of frequent itemsets. This research is inspired by simple appealing visualization of itemsets frequencies in the simple well known two dimension matrix representations. This paper proposes a new procedure to extract maximal frequent itemsets called Matrix Visualization and Extraction of Maximal Frequent Itemsets. The procedure consists of two steps. The first step sets the environment to mine data while the second extracts frequent itemsets. MVEMFI procedure has been tested by three synthetic datasets and processing time has been recorded. It has been found that MVEMFI performance is not affected by the number of transactions or the density of items' occurrences in the dataset.
Keywords – Association rules, Data mining, Maximal frequent itemset

I. INTRODUCTION

Association rule is a promising tool to expand the scope of data analysis and to reveal hidden relations among data values. Association rule miming has wide spectrum of applications in the area of customer relationship management [1], financial applications [2], tax inspection [3], traffic management [4] and computing in cloud environment [5, 6], education [7, 8].

Association rules mines dataset by discovering frequent patterns, then generates a set of rules that reveals the relationship between dataset items. The first step in association rule mining process is to extract frequent itemsets and the final step is rule generation. The most common used parameters for data mining are support and confidence. There are two ways two identify frequent itemsets either through candidate generation as in Apriori [9, 10] or without candidate generation as in pattern growth methods and their modifications [11, 12, 13, 14, 15, 16]. Also, there are three types of frequent itemsets; typical frequent itemsets [9, 10], closed frequent itemsets [15, 16, 17] and maximal frequent itemsets [11, 15, 18]. A typical frequent itemset is any itemset with frequency above a specific threshold. A closed frequent itemset is a frequent itemset that doesn't have a superset with the same frequency. A maximal frequent itemset is a frequent itemset which has no proper frequent superset. Pruning the search of frequent itemsets depends on two properties. The first property states that if an itemset is infrequent, then all it supersets must be infrequent. The second property is that if an itemset is frequent, then all its subsets must be frequent. This research proposes a new non-candidate frequent itemsets generation procedure called Matrix Extraction and Visualization of Frequent Itemsets "MVEMFI". The importance of the current research is to represent and to visualize itemsets' frequencies in a naïve simple two dimension matrix notation. MVEMFI procedure starts by constructing the matrix, then it undergoes a few processing steps and evolves by extracting frequent itemsets. The paper starts by reviewing related work in section two. Section three explains MVEMFI procedure while section four explains the conducted experiments. The results are in section five and the research is concluded in section six along with suggested future work.

II. RELATED WORK

There exits many association rule mining algorithms and this section reviews some of these algorithms. Apriori Algorithm [9, 10] employs a bottom-up search that enumerates every single frequent itemset. It starts by examining the count of single k-itemsets, then identifies frequent ones and uses them to produce k+1-itemsets. Apriori repeats the last two steps until no more frequent itemsets can be found. The exponential complexity of algorithm limits its usage to short patterns. Max-Miner [11] extracts only the maximal frequent itemsets by generating all the frequent itemsets using both bottom-up and top-down traversal. After identifying maximal frequent patterns, all frequent patterns are derived by scanning database to determine their frequency. It uses subset pruning for an infrequent itemset and "look ahead" pruning for the subsets of a frequent itemset. It mines long frequent itemsets and needs several scans to database. Pincer-Search [12] identifies maximal frequent itemsets and runs both bottom-up and top-down searches at the same time. Both previously mentioned properties are used to prune candidates based on information gathered in one direction. If some maximal frequent itemset is found in the top-down direction, then this itemset can be used to eliminate its subsets as they are frequent. Also, if an infrequent itemset is found in the bottom-up direction, then it eliminates all its supersets in the top-down direction. The difference between Max-miner and Pincer-Search is that Max-miner uses a heuristic that looks ahead for the longest itemsets and reorder the items according to their frequency, so that itemset appears in the most candidate group. FP-tree [16, 17] mines closed frequent itemsets. FP-growth uses a divide-and-conquer way. It first scans dataset to get frequent itemset and in the second scan it generates FP-tree. Then, FP-growth starts to mine FP-tree starting by 1-K itemset, constructing conditional pattern base, then conditional FP-tree and mine it. FP-max [18] is a variation of FP-tree by using maximal frequent itemset. FP-max implements maximal frequent itemsets tree to keep track of all maximal frequent itemsets. FP-max reduces the number of subset frequency test operation, thus it reduces the search time. Eclat [14] uses TID(s) and their intersections to determine itemsets' frequencies. TID are stored in a bit matrix. Eclat uses a prefix tree to search in depth-first order. MAFIA [15] mines maximal or closed frequent itemsets from a transactional database using a depth-first. Mafia uses the above two properties to prune the search along with a third one. The third pruning method is based on the fact that if one transaction is a subset of another then the frequency of the former conforms to the latter. [19] Proposed HANA algorithm that has two steps. The first step uses TID(s) to count the frequency of the k-itemsets by intersecting transactions and storing the results in a matrix and then identifies the frequent itemsets. The second step generates (k-1)-itemsets for the frequent ones by introducing the concept of multiplex matrices. [20] Proposed HOUI-Mine algorithm to mine high on-shelf utility itemsets using three tables to speed processing. OS table is used to indicate the items on-shelf information. PTTU table records the transaction utility of all the transactions occurring within a time period. COSUI table records high transaction-weighted-utility of an itemset. The pruning strategy based on the on-shelf utility upper bound. The filtration mechanism for generating itemsets is also designed to prune redundant candidate itemsets early and to systematically check the itemsets. [21] Proposed association rule hiding (ARH) algorithm which deals with sensitive data. It mines the data, extracts rules, identifies sensitive rules, and then modifies the database to hide the transactions that support those sensitive rules. They compared ARH results with the k-anonymity method. They reported that ARH has decreased the data loss and time to hide itemsets compared with the k-anonymity method. [22] Proposed MFIF method that finds the maximal frequent item first by looking for transactions with maximum number of items rather than the minimal frequent itemset that starts by k equals one and increases to get k+1 frequent itemsets. If the frequency of maximal frequent itemset is greater or equal than the support, then it is a maximal frequent itemset. Otherwise, MFIF searches the corresponding subsets for a maximal frequent itemset. [23] Proposed a mining algorithm called Mining Frequent Weighed Itemsets (FWI). FWI assign different weights to all items and uses Weighted Itemset-Tidset tree (WIT-trees). Then, they proposed a Diffset strategy for both efficient computation of the weighted support of itemsets and for mining FWI.

III. MATRIX VISUALIZATION AND EXTRACTION OF MAXIMAL FREQUENT ITEMSETS "MVEMFI" PROCEDURE

The visualization of frequent itemsets in matrix notation is shown in Fig. 1.

	e	f	ef	g	eg	fg	efg
a	ae	af	aef	ag	aeg	afg	aefg
b	be	bf	bef	bg	beg	bfg	befg
ab	abe	abf	abef	abg	abeg	abfg	abefg
c	ce	cf	cef	cg	ceg	cfg	cefg
ac	ace	acf	acef	acg	aceg	acfg	acefg
bc	bce	bcf	bcef	bcg	bceg	bcfg	bcefg
abc	abce	abcf	abcef	abcg	abceg	abcfg	abcefg
d	de	df	def	dg	deg	dfg	defg
ad	ade	adf	adef	adg	adeg	adfg	adefg
bd	bde	bdf	bdef	bdg	bdeg	bdfg	bdefg
abd	abde	abdf	abdef	abdg	abdeg	abdfg	abdefg
cd	cde	cdf	cdef	cdg	cdeg	cdfg	cdefg
acd	acde	acdf	acdef	acdg	acdeg	acdfg	acdefg
bcd	bcde	bcdf	bcdef	bcdg	bcdeg	bcdfg	bcdefg
abcd	abcde	abcdf	abcdef	abcdg	abcdeg	abcdfg	abcdefg

(a)

0000000	0010000	0100000	1000000	0110000	1010000	1100000	1110000
0000001	0010001	0100001	1000001	0110001	1010001	1100001	1110001
0000010	0010010	0100010	1000010	0110010	1010010	1100010	1110010
0000011	0010011	0100011	1000011	0110011	1010011	1100011	1110011
0000100	0010100	0100100	1000100	0110100	1010100	1100100	1110100
0000101	0010101	0100101	1000101	0110101	1010101	1100101	1110101
0000110	0010110	0100110	1000110	0110110	1010110	1100110	1110110
0000111	0010111	0100111	1000111	0110111	1010111	1100111	1110111
0001000	0011000	0101000	1001000	0111000	1011000	1101000	1111000
0001001	0011001	0101001	1001001	0111001	1011001	1101001	1111001
0001010	0011010	0101010	1001010	0111010	1011010	1101010	1111010
0001011	0011011	0101011	1001011	0111011	1011011	1101011	1111011
0001100	0011100	0101100	1001100	0111100	1011100	1101100	1111100
0001101	0011101	0101101	1001101	0111101	1011101	1101101	1111101
0001110	0011110	0101110	1001110	0111110	1011110	1101110	1111110
0001111	0011111	0101111	1001111	0111111	1011111	1101111	1111111

(b)

0	16	32	48	64	80	96	112
1	17	33	49	65	81	97	113
2	18	34	50	66	82	98	114
3	19	35	51	67	83	99	115
4	20	36	52	68	84	100	116
5	21	37	53	69	85	101	117
6	22	38	54	70	86	102	118
7	23	39	55	71	87	103	119
8	24	40	56	72	88	104	120
9	25	41	57	73	89	105	121
10	26	42	58	74	90	106	122
11	27	43	59	75	91	107	123
12	28	44	60	76	92	108	124
13	29	45	61	77	93	109	125
14	30	46	62	78	94	110	126
15	31	47	63	79	95	111	127

(c)

Fig. 1 Symbolic (a), binary (b) and decimal (c) itemsets representation in matrix notation

The analysis of Fig. 1 reveals a set of interesting findings. Itemsets are represented in the matrix by their decimal values. All possible itemsets are represented either individually, as one element in the matrix, or by combing two elements. If the matrix is reordered, their will be chunks of k-itemsets in blocks. Itemsets in the last column/row are supersets of all other items in previous column/row. In analogy, all itemsets in the first column/row are subsets of all other items in succeeding column/row. The itemset in the last column and the last row is superset for any itemset in matrix. Therefore, recognizing the relation of two itemsets would be applied rationally to their corresponding column/row. MVEMFI procedure consists of two processes, Fig. 2. The first process prepares data mining settings while the second process identifies maximal frequent itemsets from the dataset.

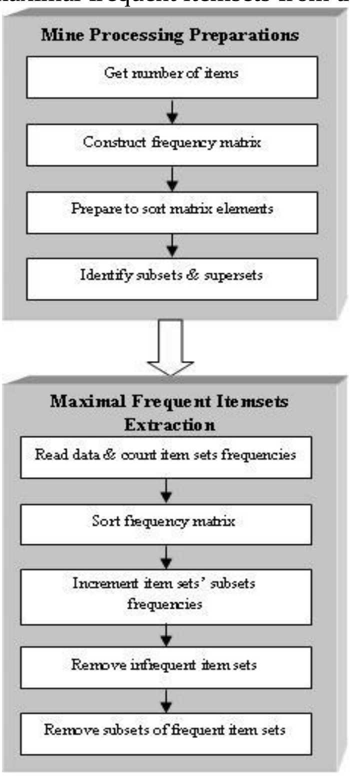


Fig. 2 MVEMFI procedure

3.1 Mine Processing Preparations Process

Mine processing preparations process consists of following four steps and the pseudo code is in Fig. 3.

Step 1: Determine the number of items and divide number of items in almost two equal values; No_row_items and No_column_items.

Step 2: Construct a two dimensional matrix where number of rows and columns equal to the number of itemsets that is generated from No_row_items and No_column_items, respectively. Fig. 1 shows symbolic, binary and decimal itemsets representations in matrix notation.

```
Process 1: Mine Processing Preparations
Input:   number of items
Output:  No_row_items, No_column_items, Frequency matrix,
         Row subsets, Row supersets, Column subsets, Column
         supersets
Steps:
  1.     No_row_items = round (No_items/2)
         No_column_items = No_items – No_row_items

  2.     Constrruct frequency matrix ( 2 ^ No_row_items, 2 ^
         No_column_items)

  3.     Sort rows' indices according to number of bits set to 1
         Sort columns' indices according to number of bits set to
         1

  4.     For each row index
             Get Row's index subsets
             Get Row's index supersets
         End
         If (not (No_row_items == No_column_items))
             For each row index
                 Get Column's index subsets
                 Get Column's index supersets
             End
         Else
             Column subsets = Row subsets
             Column supersets =Row supersets
         End
```

Fig. 3 Pseudo code for process 1

Step 3: Prepare to reorder the itemsets's elements in rows and columns according to the number of items present in the itemset. For example, the matrix in Fig. 1 is reordered as shown in Fig. 4.

Step 4: Identify the subsets and the supersets for each itemset in the first row and the first column. The indices of the subsets and the supersets for any row/column are the same. So, determining the subsets and the supersets for one row and one column is sufficient. Moreover, if No_row_items and No_column_items are equal, then get the subsets and supersets for one dimension only. The results of this step are shown in Fig. 5.

Column index of subsets for itemsets in a row = {{}, {1}, {1}, {1,2,3}, {1,2,4}, {1,3,4}, {1,2,3,4,5,6,7} }

Column index of supersets for itemsets in a row = {{1,2,3,4,5,6,7,8}, {5,6,8}, {5,7,8}, {6,7,8}, {8}, {8}, {8}, {} }

Row index of subsets for itemsets in a column = { {}, {1}, {1}, {1}, {1}, {1,2,3}, {1,2,4}, {1,3,4}, {1,2,5}, {1,3,5}, {1,4,5},{1,2,3,4,6,7,8}, {1,2,3,5,6,9,10}, {1,2,4,5,7,9,11}, {1,3,4,5,8,10,11}, {1,2,3,4,5,..15} }

Row index of supersets for itemsets in a column = {{6,7,9,12,13,14,16}, {6,8,10,12,13,15,16}, {7,8,11,12,14,15,16}, {9,10,11,13,14,15,16}, {12,13,16}, {12,14,16}, {12,15,16}, {13,14,16}, {13,15,16}, {14,15,16}, {16}, {16}, {16}, {16}, {} }

Fig. 4 Symbolic itemsets sorted by the number of items

	e	f	g	ef	eg	fg	efg
a	ae	af	ag	aef	aeg	afg	aefg
b	be	bf	bg	bef	beg	bfg	befg
c	ce	cf	cg	cef	ceg	cfg	cefg
d	de	df	dg	def	deg	dfg	defg
ab	abe	abf	abg	abef	abeg	abfg	abefg
ac	ace	acf	acg	acef	aceg	acfg	acefg
bc	bce	bcf	bcg	bcef	bceg	bcfg	bcefg
ad	ade	adf	adg	adef	adeg	adfg	adefg
bd	bde	bdf	bdg	bdef	bdeg	bdfg	bdefg
cd	cde	cdf	cdg	cdef	cdeg	cdfg	cdefg
abc	abce	abcf	abcg	abcef	abceg	abcfg	abcefg
abd	abde	abdf	abdg	abdef	abdeg	abdfg	abdefg
acd	acde	acdf	acdg	acdef	acdeg	acdfg	acdefg
bcd	bcde	bcdf	bcdg	bcdef	bcdeg	bcdfg	bcdefg
abcd	abcde	abcdf	abcdg	abcdef	abcdeg	abcdfg	abcdefg

Fig. 5 Itemsets' subsets and supersets

3.2 Maximal Frequent Itemsets Extraction Process

Maximal frequent itemsets extraction process consists of the following five steps and the pseudo code is in Fig. 6.

Step 1: Read the data set and convert each transaction itemset into decimal value. Then, for each transaction calculate the matrix element indices and increments its count.

Step 2: Sort the matrix according to step#3, process#1.

Step 3: Increment all subsets of each itemset in the matrix, using the discovered supersets. This enables the calculations of k-itemset frequency.

Step 4: Remove infrequent itemsets. An infrequent itemset has a frequency less than the support value.

Step 5: Remove all subsets of frequent items, using the discovered subsets and supersets in step#4, process#1. The removal starts with subsets of frequent itemsets in columns, then in rows as the number of columns maybe less than those of rows.

Process 2: Maximal Frequent Itemsets Extraction

Input: *Data set, Support, No_row_items, No_column_items, Frequency matrix, Row subsets, Row supersets, Column subsets, Column supersets*

Output: *Maximal Frequent itemsets*

Steps:

1. *Read Tid from Dataset*

2. *For each Tid*
   Dec_Tid = decimal (Tid)
   Column index = floor (Dec_Itemset / (2^row_items))
   Row index = Dec_Itemset % (2^row_items)
   Frequency_matrix(Row index, Column index)++
   End

3. *For each row*
   For each column
   Remove Frequency_matrix(row,column) < support
   End
   End

4. Remove all subsets of frequent itemsets in *Frequency_matrix(row,column)*

Fig. 6 Pseudo code for process 2

For example, if the number of items is five {a, b, c, d, e}, then items are divided into three items {a, b, c} and two other items {d, e}. The itemsets for {a, b, c} are represented in matrix row elements and {d, e} are represented in matrix column elements. Next, the Frequency matrix is constructed by initiating No_rows = 8 and No_column = 4. In step 3, both rows and columns values are sorted according to the number of items present in each itemset, Fig. 7. In the last step, the subsets and the supersets for each itemset in a row and a column is identified. Then, step#1, process#2 starts by reading data and converting it into decimal number, Table 1. Next, the matrix element index for each transaction is calculated and the corresponding matrix element is incremented, Fig. 8(a).

	D	E	DE
A	AD	AE	ADE
B	BD	BE	BDE
C	CD	CE	CDE
AB	ABD	ABE	ABDE
AC	ACD	ACE	ACDE
BC	BCD	BCE	BCDE
ABC	ABCD	ABCE	ABCDE

Fig. 7 Sorted Symbolic itemsets representation

Table 1 Dataset Sample

T No.	Binary Itemset					Decimal Value
T1	1	0	1	0	0	20
T2	0	1	1	0	1	13
T3	1	1	0	0	1	25
T4	1	1	0	1	0	26
T5	1	0	1	1	0	22
T6	1	1	0	0	0	24
T7	1	0	0	0	0	16

T8	1	1	1	1	0	30
T9	0	0	0	1	1	3
T10	0	0	1	1	1	7
T11	0	0	0	1	1	3
T12	0	0	1	1	1	7
T13	0	0	0	0	1	1

For example, in the set of transactions {AC, CE, ABDE}, their decimal values are {5, 20, 27}. The matrix indices for each one is {(5,0), (4,2), (3,3)} and their element frequency in matrix is incremented by one. In step 2, Frequency_matrix is sorted, Fig. 8(b).

$$\begin{pmatrix} & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

(a) (b)

Fig. 8 Frequency_matrix (a) before sorting (b) after sorting

In step 3, all subsets of each itemset in Frequency_matrix are incremented, using the discovered row's supersets and column's supersets. Fig. 9 shows Frequency_matrix after updates.

$$\begin{pmatrix} & 1 & 3 & 4 \\ 5 & 1 & 0 & 1 \\ 4 & 0 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 4 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} & 5 & 7 & 4 \\ 7 & 2 & 1 & 1 \\ 7 & 2 & 3 & 2 \\ 6 & 2 & 3 & 1 \\ 4 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 4 & 1 & 2 & 1 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

(a) row frequency updates (b) column frequency updates

Fig. 9 Frequency_matrix after incrementing all itemsets' subsets

Next, infrequent itemsets are removed using a support value equals to one, Fig. 10. Finally, all frequent subsets of a frequent super itemset are removed starting by column then row, Fig. 11. At the last step, the final maximal frequent itemsets are identified along with their frequency. Maximal frequent itemsets are {abc, ad, cd, bde}

$$\begin{pmatrix} & 5 & 7 & 4 \\ 7 & 2 & 0 & 0 \\ 7 & 2 & 3 & 2 \\ 6 & 2 & 3 & 0 \\ 4 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 4 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 10 Frequent itemsets

$$\begin{pmatrix} & 0 & 0 & 4 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 2 & 3 & 0 \\ 4 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

(a) (b)

Fig. 11 Subsets removal of frequent itemsets in (a) Column and (b) Row

IV. EXPERIMENT

Three data sets are used to examine this work which has been used in [19]. Three datasets {A, B, C} are generated synthetically. The number of items in each set is 24 and the average number of items presence per transaction per data set is {2, 7 and 13} with 2000 transactions per data set.

V. RESULTS

Table 2 and Fig. 12 illustrate the processing time of MVEMFI algorithm for the three datasets for different transactions. Table 3 shows the mean and the standard deviation for each dataset separately and for the three datasets. MVEMFI is characterized by having almost a constant time of processing with mean value equals to 715.57 and standard deviation of 5.03.

Table 2 MVEMFI procedure processing time

Transaction No.	Dataset A	Dataset B	Dataset C
50	696.80	702.18	706.06
100	699.24	702.70	710.45
150	702.19	703.36	710.94
200	705.40	707.64	711.32
250	705.43	708.63	724.63
500	717.31	725.26	732.60
1000	723.29	733.80	733.09
2000	735.97	738.33	736.96

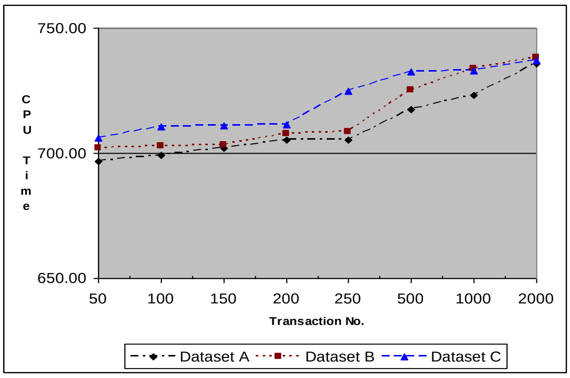


Fig. 12 CPU time for MVEMFI procedure

Table 3 Statistical measures for MVEMFI procedure

Statistics	Dataset A	Dataset B	Dataset C	Dataset A, B
Mean	710.71	715.24	720.76	715.57
Standard Deviation	13.59	14.87	12.41	5.03

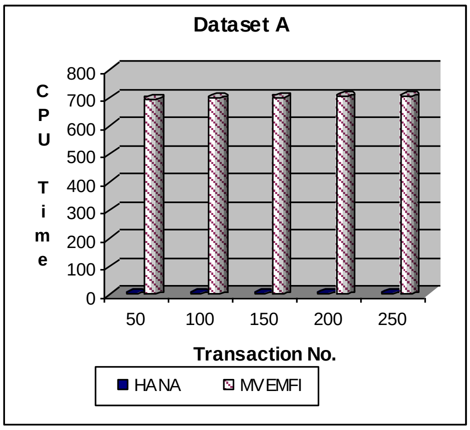
Table 4 and Fig. 13 show the comparison between MVEMFI procedure and HANA algorithm in terms of the processing time. It is noticeable that HANA algorithm outperformed MVEMFI in case of small transaction number and low density datasets. On the other hand, MVEMFI procedure outperformed HANA algorithm in case of high density datasets even in small number of transactions.

Table 4 HANA algorithm processing time

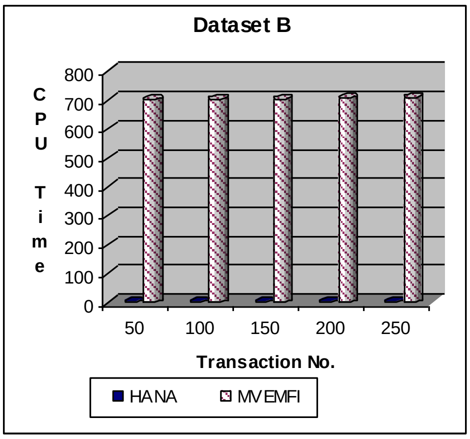
Transaction No.	Dataset A	Dataset B	Dataset C
50	0.14	2.11	74
100	0.34	9.09	1444
150	0.61	25	7740
200	0.97	54	139660
250	1.34	87	373395

VI. CONCLUSION AND FUTURE WORK

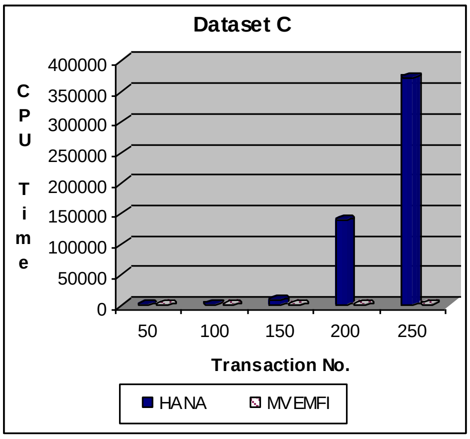
This paper proposes MVEMFI procedure to discover frequent items without candidate generation using matrix notation. It can inferred that the variation in processing times for all runs is small regardless of number of transaction or the density of the items in the datasets which is better than a solution with high variability in the processing time.



(a)



(b)



(c)

Fig. 13 Comparison between MVEMFI procedure and HANA algorithm (a) dataset A (b) dataset B (c) dataset C

Also, MVEMFI is characterized by using matrix notation which is a simple well known data type. HANA and MVEMFI algorithms interchange the best performance as HANA uses transaction identifiers to extract frequent itemsets which are not the case with MVEMFI. HANA algorithm extracts all frequent itemsets while MVEMFI extracts maximal frequent itemsets. This explains why HANA algorithm outperformed MVEMFI in case of datasets with fewer items per transaction as well as when the number of transactions is few. There are several advantages of the MVEMFI algorithm. The above two characteristics are also two advantages that covers predictability, stability and simplicity. The next advantage is that it divides the number of items into two portions which limits the explosive nature of itemsets generation. Also, three of processing steps when identified for one matrix element are applied to the containing column/row. Those steps are reorder itemsets' elements, identify subsets and supersets. The last advantage is that the generation of full frequent itemsets is minimized as it is performed after the removal of infrequent ones.

Future work may include adapting MVEMFI procedure to extract typical or closed frequent itemsets. Further work may include hardwire MVEMFI procedure and upon successful performance, it maybe a solution for stream data mining. Another suggestion is to represent frequent itemsets by higher matrix dimension and access the cost and the benefit of adding more dimensions to the matrix.

References

[1] E. Ngai, L. Xiu, and D. Chau, Application of data mining techniques in customer relationship management: A literature review and classification, *Expert Systems with Applications, 36(2)*, 2009, 2592–2602.
[2] M. Mak, G. Ho, and S. Ting, A financial data

mining model for extracting customer behavior, International *Journal of Engineering Business Management*, Wai Hung Ip (Ed.), ISBN: 1847-9790, InTech, DOI:10.5772/50937. Available from: http://www.intechopen.com/journals/international_journal_of_engineering_business_management/a-financial-data-mining-model-for-extracting-customer-behavior

[3] Q. Zhu, L. Guo, J., N. Xu, and W. Li, Research of tax inspection cases – choice based on association rules in data mining", *Proceedings of the Eighth International IEEE Conference on Machine Learning and Cybernetics*, Hebei, China, 2009, 2625 - 2628.

[4] W. Cheng, X. Ji, C. Han, and J. Xi, The mining method of the road traffic illegal data based on rough sets and association rules", *IEEE International Conference on Intelligent Computation Technology and Automation*, Changsha, China, 2010, 856 - 859.

[5] L. Li, and M. Zhang, The Strategy of Mining Association Rule Based on Cloud Computing, *International Conference on Business Computing and Global Informatization (BCGIN)*, Shanghai, China, 2011, 475-478.

[6] J. Li, P. Roy, S. Khan, L. Wang, and Y. Bai, Data mining using clouds: an experimental implementation of Apriori over MapReduce. *The 12th IEEE International Conference on Scalable Computing and Communications (ScalCom 2012)*, Changzhou, China, 2012.

[7] V. Kumar, and A. Chadha, Mining association rules in student's assessment data, *International Journal of Computer Science Issues, 9(5)*, 2012, 211-216.

[8] R. Prasad, and D. Babu, Mining association rules with static and dynamic behavior of learner in the Internet, *International Journal of Computer Applications, 37(4)*, 2012, 26-30.

[9] R. Agrawal, T. Imielinski, and A. Swami, Database Mining: A Performance Perspective, *IEEE Transactions on Knowledge and Data Engineering, 5(6)*, 1993, 914-925.

[10] R. Agrawal, and R. Srikant, Fast Algorithms for Mining Association Rules, *International Confeence Very Large Data Bases*, Santiago, Chile, 1994, 487-499.

[11] R. Bayardo, Efficiently mining long patterns from databases, *ACM Sigmod Record, 27(2)*, 1998, 85-93.

[12] D. Lin, and Z. Kedem, Pincer-search: a new algorithm for discovering the maximum frequent set. *In Advances in Database Technology — EDBT'98 · Lecture Notes in Computer Science, 1377*, 1998, *103-119*.

[13] G. Grahne, and J. Zhu, High performance mining of maximal frequent itemsets", *Proc. SIAM Int'l Conf. High Performance Data Mining*, CA, USA, 2003, 135−143.

[14] C. Borgelt, Efficient implementations of Apriori and Eclat. *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations FIMI 03*, Florida, USA, 2003.

[15] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, MAFIA: a maximal frequent itemset algorithm, *IEEE Transactions on Knowledge and Data Engineering, 17(11)*, 2005, 1490–1504.

[16] J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation, *Proceeding of the ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, TX, USA, 2000, 1-12.

[17] J. Han, J. Pei, Y. Yin, and R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, *Data Mining and Knowledge Discovery, 8*, 2004, 53–87.

[18] G. Grahne, and J. Zhu, Efficiently using prefix-trees in mining frequent itemsets, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, Melbourne, Florida, USA, 2003, 98-115.

[19] M. Hana, HANA algorithm: a novel algorithm for frequent itemsets generation, *International Journal of Intelligent Computing and Information Science, 7(2)*, 2007, 41-53.

[20] G. Lan, G. C., T. Hong, and V. Tseng, A three-scan mining algorithm for high on-shelf utility itemsets, *ACIIDS'10 Proceedings of the Second international conference on Intelligent information and database systems: Part II*, 2010, 351-358.

[21] R. Sugumar, A. Rengarajan, and M.Vijayanand, Extending K-anonymity to privacy preserving data mining using association rule hiding algorithm, *International Journal of Advanced Research in Computer Science and Software Engineering, 2(6)*, 2012. Available online at: www.ijarcsse.com.

[22] H. Jnanamurthy, H. Vishesh, V. Jain, P. Kumar, and R. Pai, Top down approach to find maximal frequent itemsets using subset creation, *Computer science & information technology, 2(4)*, 2012, 445-452.

[23] Vo, B., Coenen, F., and Le, B, A new method for mining frequent weighted itemsets based on WIT-trees, *Expert Systems with Applications, 40(4)*, 2013, 1256-1264.