JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.6, Special Issue.11, Dec 2018 E-ISSN: 2347-2693

A Memory Efficient Implementation of Frequent Itemset Mining with Vertical Data Format Approach

P. Sumathi[1*], Dr. S. Murugan[2]

[1]Department of Computer Science, Nehru Memorial College (Autonomous), Tiruchirappalli, India
[2]Department of Computer Science, Nehru Memorial College (Autonomous), Tiruchirappalli, India

*Corresponding Author: sumiparasu@gmail.com

Available online at: www.ijcseonline.org

***Abstract*—** Data mining is the process of extracting the concealed information and rules from large databases. But the real world datasets are sparse, dirt and also contain hundreds of items. Frequent Pattern Mining (FPM) is one of the most intensive problems in discovering frequent itemsets from such datasets. Apriori is one of the premier and classical data mining algorithms for finding frequent patterns but it is not an optimized one. So over last two decades a remarkable variations and improvements were made to overcome the inefficiencies of Apriori algorithm such as FPGrowth, TreeProjection, Charm, LCM, Eclat and Direct Hashing and Pruning (DHP), RARM, ASPMS etc., In any case, a little enhancement in the algorithm improves the mining process considerably. Frequent itemset mining with vertical data format approach has been proposed as an improvement over the basic Apriori, which reduces the number of database scans and also uses array storage structure. This research paper has proposed a space efficient implementation of finding frequent itemsets with vertical data format using jagged array. It reduces the usage of memory by allocating exact memory. An experiment is done between the array implementation of vertical data format approach and jagged array implementation. From the experiment it is proved that the proposed jagged array implementation method utilizes the memory efficiently when compared with the traditional multidimensional array.

***Index Terms* —** Apriori, Array, Eclat, Frequent Pattern Mining, FPGrowth, Jagged Array, and Vertical Data Format.

I. Introduction

Now-a-days, volumes of data are exploding both in scientific and commercial domains. Data mining techniques are used to extract unknown information from the huge amount of data and became popular in many applications. Association Rule Mining (ARM) is one of an important core data mining techniques to discover patterns/rules among items in a large database of variable-length transactions. Its goal is to identify the groups of items that most often occurs together i.e., it focuses on finding frequent itemsets each occurring at more than a minimum support frequency (min_sup) among all transactions. It is widely used in market basket transaction data analysis, graph mining applications like substructure discovery in chemical compounds, pattern finding in web browsing, word occurrence analysis in text documents, and so on [1].

The major risks associated with finding frequent itemsets are i) computational time and ii) memory needed for the task because even with a moderate sized dataset, the search space and memory utilization of FPM is enormous, which is exponential to the length of the transactions in the dataset. Therefore, it is essential to perform FPM analysis in a space-and-time efficient way. Many researchers in this area focused on reducing computational time to find frequent patterns and this work focuses on reducing the memory utilization using jagged array storage structure in the vertical data mining algorithms.

Rest of the paper is organized as follows. Section 2 describes the review of literature. The proposed implementation method of Vertical Data Format (VDF) is illustrated in section 3. The comparison of existing and the proposed implementation methods are discussed in section 4 and finally section 5 ends with conclusion.

II. REVIEW OF LITERATURE

Improving the computational time and memory is always an issue in ARM and this section briefs the research contributions made by different researchers in this line which pawed way for the proposed implementation.

In [2], the authors have presented a VDSRP method to generate complete set of regular patterns over a data stream at a user given regularity threshold using sliding-window and VDF. It has been proved that the proposed method outperforms both in execution and memory consumption.

Ravikiran, D., et. al, have proposed a new model called RCP to mine regular sort of crimes in crime database using VDF which requires only one database scan. From the experimental results they proved that RCP is more efficient than the existing RPtree[3]. In [4], the authors have focused on the various FPM techniques, their challenges in static and stream data environment.

The authors in [6] have presented a new algorithm, which mine frequent itemsets with vertical format. They proved that the new algorithm needs a single database scan and finds new frequent item sets through 'and operation' between item sets. The new algorithm requires less storage space, and improves the efficiency of data mining.

An enhanced Apriori and Eclat has been introduced in [8], in which individual thresholds for each itemset has been used and proved that that the enhanced-Apriori algorithm outperforms Enhanced-Eclat Algorithm.

In [9], the authors have presented an improved version of Eclat called Eclat-growth algorithm based on increased search strategy. For reducing the runtime in generating an intersection of two itemsets and support degree calculation, a BSRI (Boolean array Setting and Retrieval by Indexes of transactions) method has been introduced. It has been proved by them that the Eclat-growth outperforms Eclat, Eclat-diffsets, Eclat-opt and hEclat in mining association rules.

In [10], a VFFM algorithm has been developed which represents the transaction database in vertical format in the form of binary, where the attribute presence and absence is represented by 1 and 0 respectively. After one scan of transaction database for transformation it generates candidate sets and subsets similar to Apriori algorithm. The support value of each candidate itemsets is counted by intersection of every pair of frequent single items instead of database scan and proved that the VFFM outperforms Apriori.

Compressed bit vectors of frequent itemsets based on Boolean algebra named Vertical Boolean Mining (VBM) has been presented in [11] and it performs the intersection of two compressed bit vectors without making any costly decompression operation. They proved from the experiments that the VBM is better than Apriori and the classical vertical association rule mining algorithms in terms of mining time and memory usage.

A novel VDF representation called Diffset has been developed by the authors in [12], which keep track of the differences in the tid's of a candidate pattern and from which it generates frequent patterns. The method cut down the size of memory required to store intermediate results and also increased performance significantly.

From the existing literatures, it is noted that no authors have proposed a jagged array implementation of VDF approach for enhancing the memory requirement of VDF. Thus, this work implements VDF using the jagged array for efficient utilization of memory.

III. JAGGED ARRAY IMPLEMENTATION OF VERTICAL DATA FORMAT APPROACH

Frequent patterns are itemsets [set of items, such as milk and bread, that appear frequently together in a transaction data set], subsequences [buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database], or substructures [subgraphs, subtrees or sublattices] that appear in a dataset with frequency no less than a user-specified threshold (min_sup)[7]. Finding frequent patterns plays an essential role in mining associations, correlations and many other interesting relationships among data. ARM is one of the data mining techniques to discover the hidden patterns/rules among items in a large database of variable-length transactions that help in making decision and predictions [4].

Apriori Algorithm, FP-Growth and Eclat [4] are the popularly available static data mining techniques for finding frequent patterns. Apriori is the basic algorithm for mining frequent patterns which suffers from space complexity due to large number of candidate generation and also requires multiple scans of database. FP-growth uses a tree structure for mining frequent itemsets. Due to limited number of database scans and zero candidates, it is efficient as compared to Apriori. Both the Apriori and FP-growth algorithms mine frequent patterns in Horizontal Data Format (HDF) (i.e., {TID: itemset}), where TID is a transaction-id and itemset is the set of items in TID and it is shown in Table I.

TABLE I. TRANSACTION DATABASE D IN HDF

TID	List of item IDS
T1	A,B,E
T2	B,D
T3	B,C
T4	A,B,D
T5	A,C
T6	B,C
T7	A,C
T8	A,B,C
T9	A,B,C,E

But the data can also be presented in {item: TID-set} format where item is an item name and TID-set is the set of transactions containing the item called VDF. The VDF is used in Eclat algorithm that minimizes the database scan and uses set intersection of Tid’s for finding the support count for *k*-itemsets where *k*=2,3,...,n. The VDF of the transaction database D is shown in Table II. The comparisons between the Apriori, FP-Growth and Eclat with different parameters are shown in Table III. From Table III and in [4] it is observed that the FP mining algorithms which use VDF are very fast and requires less memory space when compared with HDF approaches. But, the VDF approaches use array storage structure for storing the database in memory.

TABLE II. VDF OF D

itemset	TID_set
A	T1,T4,T5,T7,T8,T9
B	T1, T2, T3, T4, T6, T8,T9
C	T3, T5, T6, T7, T8,T9
D	T2,T4
E	T1,T9

To reduce memory space further, this research work implements the VDF using jagged array. It is a special case of 2-D array and it is an array of array in which the length of each array can differ. This concept is available in JAVA, VB.NET and C#.NET. This implementation helps to reduce the memory needed considerably because in the real life grocery datasets the customers will not purchase all the items in the shop. Thus, this implementation utilizes the memory effectively.

A. *An Example*

The first part of this section shows the memory requirement for the array implementation of VDM. Let the grocery shop sells *n* (5) items viz., A, B, C, D and E and consider the transaction database *D* shown in Table I. It contains *t* (9) transactions and it is scanned first to generate VDF. The VDF of Table I is shown in Table II.

TABLE III. COMPARISON BETWEEN STATIC DATA MINING TECHNIQUES FOR FINDING FREQUENT PATTERNS [5]

Comparison Parameters	Apriori	FP-Growth	ECLAT
Technique	Breadth first search and Apriori property (for pruning)	Divide and conquer	Depth first search & intersection of T-id’s
Database Scan	scanned for each time a candidate item set is generated	Two times	Few times
Drawback(s)	1. Requires large memory space. 2. Too many candidate item set.	FP-tree is expensive to build and consumes more memory	It requires the virtual memory to perform the transaction.
Advantage(s)	1. Easy to implement. 2. Use large item set property	Database is scanned two times	1. No need to scan the database each time 2. fast
Data format	Horizontal	Horizontal	Vertical
Storage structure	Array	Tree (FP-tree)	Array
Time	More execution time	Execution time is less than Apriori	Execution time is less than Apriori

The support count (SC) for each item is the number of transaction-id's that it contains i.e. the SC of A, SC_A=count(A)=6. Similarly, SC_B=7, SC_C=6, SC_D=2 and SC_E=2. Let the min_sup be 2. The frequent 1-itemset contains {A, B, C, D, E}. The VDF is actually stored in the memory as 2-D array, where number of rows (*r*) = items in the grocery shop and number of columns(*c*) = *t*. Here *r*=5 and *c*=9. The memory required for storing 1-itemset in VDF format is

$$TM_1 = (r \times c \times sizeof(tid)) + (sizeof(item_{11}) \times r) \tag{1}$$

Where $item_{11}$ is the first item in the frequent 1-itemset, tid is the transaction-id and sizeof is a built-in function which says the number of bytes required for the argument.

Here each tid requires 2 bytes and $item_{11}$ requires 1 byte of memory respectively. All items say A, B, C, D and E sold in the grocery shop are frequent 1-itemsets. Therefore the VDF requires (5×9×2)+(5×1) = 95 bytes of memory i.e., TM_1 = 95 bytes. Suppose if there are some in-frequent items in 1-itemsets, they can be removed which saves memory considerably. The number of bytes of memory removed from 1-itemset is computed as

$$rbytes_1 = (rr_1 \times c \times sizeof(tid)) + (rr_1 \times sizeof(item_{11})) \tag{2}$$

Where, rr_1 is the number of rows to be removed as in-frequent. Therefore the total bytes of memory for frequent 1-itemset is

$$M_1 = TM_1 - rbytes_1 \tag{3}$$

Here $M_1 = 95 - 0 = 95$ bytes. Similarly, in iteration 2, the possible 2-itemsets combinations are generated from frequent 1-itemsets and it is {AB, AC, AD, AE, BC, BD, BE, CD, CE, DE}. Suppose if there are n items in 1-itemset, the possible two item combinations are $(n \times n-1)/2$ say tc_2. Among them, the numbers of itemset combinations say x may be in-frequent which need not be placed in VDF. Therefore, the memory required for frequent 2-itemset shown in Table IV is

$$TM_2 = ((tc_2 - x) \times c \times sizeof(tid)) + (sizeof(item_{21}) \times (tc_2 - x)) \quad (4)$$

Where, $item_{21}$ is the first item in the frequent 2-itemset. In this example, the combinations viz., AD,CD,CE and DE are in-frequent and based on equation (4), the VDF requires ((10 - 4) × 9 × 2) + (2 × (10 - 4)) = 108 + 12 = 120 bytes. Similarly from Table IV, the 3-itemset combinations are {ABC, ABD, ABE, ACE, BCD, BCE, BDE} and the combinations ABD, ACE, BCD, BCE and BDE are in-frequent, therefore the frequent 3-itemset requires ((7-5)×9×2)+(7-5)×3)=42 bytes of memory and the VDF of 3-frequent itemsets is shown in Table V. The process is repeated until no frequent itemsets are found.

TABLE IV. VDF OF 2-ITEMSETS

Itemset	TID_set
AB	T1,T4,T8,T9
AC	T5,T7,T8,T9
AE	T1,T9
BC	T3,T6,T8,T9
BD	T2,T4
BE	T1,T9

Therefore, the total memory required for VDF using 2-D array is

$$TM = M_1 + \sum_{i=2}^{itemset_i \neq \emptyset} TM_i \quad (5)$$

Where M_1 is calculated using (3) and TM_i are calculated using the equation (6) shown below.

$$TM_i = ((tc_i - x) \times c \times sizeof(tid)) + (sizeof(item_{i1}) \times (tc_i - x)) \quad (6)$$

Where, tc_i and x *are* the number of items and in-frequent items in the candidate i-frequent itemset. For the above example TM = 95+120+42 =257 bytes of memory. If the same is implemented using jagged array, the memory requirement is reduced considerably. The format of jagged array representation for candidate 1-itemset is shown in Table VI and all items in it are frequent which forms frequent 1-itemset.

TABLE V. VDF OF 3-ITEMSETS

itemset	TID_set
ABC	T8,T9
ABE	T1,T9

TABLE VI. JAGGED ARRAY REPRESENTATION OF 1-ITEMSET

itemset	TID_set						
A	T1	T4	T5	T7	T8	T9	
B	T1	T2	T3	T4	T6	T8	T9
C	T3	T5	T6	T7	T8	T9	
D	T2	T4					
E	T1	T9					

The memory required for candidate 1-itemset TM_1 is calculated as

$$TM_1 = \sum_{\forall item \in \{itemset_1\}} SC_{item} \times sizeof(tid) + sizeof(item) \quad (7)$$

As in two-D representation, there may be x in-frequent items in candidate 1-itemset say {in-frequent} = {$item_1$, $item_2$, …,$item_x$} then the memory for {in-frequent} be saved by removing it and the amount of memory removed is computed as shown in equation (8).

$$rbytes_1 = \sum_{\forall item \in \{in-frequent\}} SC_{item} \times sizeof(tid) + sizeof(item) \quad (8)$$

Therefore the total memory required for frequent 1-itemset in jagged representation is computed using (3) with the values computed using (7) and (8) respectively. Similarly, the jagged array representation of frequent 2-itemset shown in Table VII requires TM_2 - $rbytes_2$ memory space where TM_2 and $rbytes_2$ are calculated by using (9) and (10) respectively.

$$TM_2 = \sum_{\forall item \in \{itemset_2\}} SC_{item} \times sizeof(tid) + sizeof(item) \quad (9)$$

$$rbytes_2 = \sum_{\forall item \in \{in-frequent\}} SC_{item} \times sizeof(tid) + sizeof(item) \quad (10)$$

The jagged representation of frequent 3-itemset is shown in Table VIII which requires TM_3 - $rbytes_3$ memory. This process continues until no more frequent itemsets are found. For this case the candidate 4-itemset is null and the algorithm terminates. Therefore, the total memory required for the jagged implementation is calculated using equation (11).

$$TM = \sum_{i=1}^{itemset_i \neq \emptyset} TM_i - rbytes_i \quad (11)$$

Where, TM_i and $rbytes_i$ are calculated using (12) and (13) respectively.

$$TM_i = \sum_{\forall item \in \{itemset_i\}} SC_{item} \times sizeof(tid) + sizeof(item) \quad (12)$$

$$rbytes_i = \sum_{\forall item \in \{in-frequent_i\}} SC_{item} \times sizeof(tid) + sizeof(item) \quad (13)$$

TABLE VII. JAGGED ARRAY REPRESENTATION OF 2-ITEMSET

itemset	TID_set			
AB	T1	T4	T8	T9
AC	T5	T7	T8	T9
AE	T1	T9		
BC	T3	T6	T8	T9
BD	T2	T4		
BE	T1	T9		

TABLE VIII. JAGGED ARRAY REPRESENTATION OF 3-ITEMSET

itemset	TID_set	
ABC	T8	T9
ABE	T1	T9

For this example, the jagged representation requires

TM_1 = (6×2 +1)+(7×2+1)+(6×2+1)+(2×2+1)+(6×2+1)
= 13+15+13+5+5=51 bytes

$rbytes_1$ = 0

Therefore M_1=51- 0 = bytes

TM_2 = (4×2+2) +(4×2+2)+(1×2+2)+ (2×2+2) +(4×2+2) +(2×2+2) +(2×2+2) +(0×2+2) +(1×2+2)+ (0×2+2)
=10+10+4+6+10+6+6+2+4+2=60 bytes

$rbytes_2$ = (1×2+2)+(0×2+2)+(1×2+2)+(0×2+2)=12 bytes

Therefore M_2 requires = 60 - 12 = 48 bytes of memory.

Similarly, M_3 requires 14 bytes and therefore, the jagged representation for this example requires

$TM=M_1+M_2+M_3$ =51+48+14=113 bytes of memory which is less than 50% in the original array representation.

IV. RESULTS AND DISCUSSION

From the example discussed in section 3.1, the jagged implementation has several advantages. They are

1. No memory space is wasted as in 2-D array because jagged array allocates space only to the transactions in which the items occurs.
2. Minimizes the memory space required than the original array implementation because for the above example the array implementation requires 257 bytes of memory, where as it is 113 bytes when using jagged implementation i.e., it requires less than 50% of memory when compared with the array representation.

Thus, it is finalized that the jagged implementation saves memory significantly and also fast when compared with the horizontal data format approaches.

V. CONCLUSION

From the literatures, it is observed that there is always a trade-off between the computational time and memory in generating frequent itemsets. It is also found that the vertical data format approaches reduces the database scans and finds the support counts by intersection. Though it is best, the array storage structure implementation used by VDF requires more memory because it takes the assumption that each item may fall almost in all transactions. But in real world grocery datasets, each transaction will not contain all items and each item may not present in all transactions. So to reduce the memory consumption, this research work used the jagged array representation for efficient usage of memory and from the experiments it is proved that the proposed implementation approach reduces more than 50% of memory when compared with original 2-D array implementation. In future, this work can be extended to the test real world grocery datasets of more dimensions.

REFERENCES

[1]. Liu, Y., Liao, W. K., Choudhary, A. N., & Li, J. (2008). Parallel Data Mining Algorithms for Association Rules and Clustering, In Intl. Conf. on Management of Data, pp.1-25.

[2]. Kumar, G. V., Sreedevi, M., & Kumar, N. P. (2012). Mining Regular Patterns in Data Streams Using Vertical Format. *International Journal of Computer Science and Security (IJCSS)*, *6*(2), pp.142-149.

[3]. Ravikiran, D., & Srinivasu, S. V. N. (2016). Regular Pattern Mining on Crime Data Set using Vertical Data Format. *International Journal of Computer Applications*, *143*(13).

[4]. Singla, V. (2016). A Review: Frequent Pattern Mining Techniques in Static and Stream Data Environment. *Indian Journal of Science and Technology*, *9*(45), pp.1-7.

[5]. Ishita, R., & Rathod, A. (2016). Frequent Itemset Mining in Data Mining: A Survey. *International Journal of Computer Applications*, *139*(9).

[6]. Guo, Y. M., & Wang, Z. J. (2010, March). A vertical format algorithm for mining frequent item sets. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on* (Vol. 4, pp. 11-13). IEEE.

[7]. Han, J., Kamber, M. Data Mining Concepts and Techniques, *Morgan Kaufmann Publishers*, 2006.

[8]. S.Sharmila, Dr. S.Vijayarani. (2017). Frequent Itemset Mining and Association Rule Generation using Enhanced Apriori and Enhanced Eclat Algorithms, International Journal of Innovative Research in Computer and Communication Engineering, 5(4), pp. 679- 6804.

[9]. Zhiyong Ma, Juncheng Yang, Taixia Zhang and Fan Liu. (2016). An Improved Eclat Algorithm for Mining Association Rules Based on Increased Search Strategy, *International Journal of Database Theory and Application*, 9(5), pp.251-266.

[10]. C.Ganesh, B.Sathiyabhama and T.Geetha. (2016). Fast Frequent Pattern Mining Using Vertical Data Format for Knowledge Discovery, International Journal of Emerging Research in Management &Technology, 5(5), pp.141-149.

[11]. Hosny M. Ibrahim, M.H. Marghny and Noha M.A. Abdelaziz. (2015). Fast Vertical Mining Using Boolean Algebra, *International Journal of Advanced Computer Science and Applications*, 6(1), pp.89-96.

[12]. Mohammed J. Zaki amd Karam Gouda. (2003), Fast Vertical Mining Using Diffsets SIGKDD '03, ACM.

Authors Profile

P.Sumathi received her B.Sc and M.Sc degrees in Computer Science from Seethalakshmi Ramaswami College, affiliated to Bharathidasan University, Tiruchirappalli, India in 2001 and 2003 respectively. She received her M.Phil degree in Computer Science in 2008 from Bharathidasan University. She is presently working as an Assistant Professor in the Department of Computer Science, Vysya College, Salem, India. She is currently pursuing Ph.D., degree in Computer Science in Bharathidasan University. Her research interests include Data structures, Database and Data Mining techniques.

S.Murugan received his M.Sc degree in Applied Mathematics from Anna University in 1984 and M.Phil degree in Computer Science from Regional Engineering College, Trichirappalli in 1994. He is an Associate Professor in the department of Computer Science, Nehru Memorial College (Autonomous), affiliated to Bharathidasan University since 1986. He has 32 years of teaching experience in the field of Computer Science. He has completed his Ph.D., degree in Computer Science with the specialization in Data Mining from Bharathiyar University in 2015. His research interest includes Data and Web Mining. He has published many research articles in the National and International journals.