IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 15, Issue 3 (Nov. - Dec. 2013), PP 79-84
www.iosrjournals.org

A Quantified Approach for large Dataset Compression in Association Mining

[1] Pratush Jadoun , [2] Kshitij Pathak
[1]*Department of Information Technology, MIT Ujjain (M.P.) India*
[2] *Department of Information Technology, MIT Ujjain (M.P.) India*

Abstract: *With the rapid development of computer and information technology in the last several decades, an enormous amount of data in science and engineering will continuously be generated in massive scale; data compression is needed to reduce the cost and storage space. Compression and discovering association rules by identifying relationships among sets of items in a transaction database is an important problem in Data Mining. Finding frequent itemsets is computationally the most expensive step in association rule discovery and therefore it has attracted significant research attention. However, existing compression algorithms are not appropriate in data mining for large data sets. In this research a new approach is describe in which the original dataset is sorted in lexicographical order and desired number of groups are formed to generate the quantification tables. These quantification tables are used to generate the compressed dataset, which is more efficient algorithm for mining complete frequent itemsets from compressed dataset. The experimental results show that the proposed algorithm performs better when comparing it with the mining merge algorithm with different supports and execution time.*

Keywords*: Apriori Algorithm, mining merge Algorithm, quantification table.*

I. Introduction

Data compression is one of good solutions to reduce data size that can save the time of discovering useful knowledge by using appropriate methods, for example, data mining [5]. Data mining is used to help users discover interesting and useful knowledge more easily. It is more and more popular to apply the association rule mining in recent years because of its wide applications in many fields such as stock analysis, web log mining, medical diagnosis, customer market analysis, and bioinformatics. In this research, the main focus is on association rule mining and data pre-process with data compression. Proposed a knowledge discovery process from compressed databases in which can be decomposed into the following two steps [2].

The exponential growth in genomic data accumulation possesses the challenge to develop analysis procedures to be able to interpret useful information from this data. And these analytical procedures are broadly called as "Data Mining". **Data mining** (also known as Knowledge Discovery in Databases - KDD) has been defined as "The nontrivial extraction of implicit, previously unknown, and potentially useful information from data"[10] .The goal of data mining is to automate the process of finding interesting patterns and trends from a give data. Several data mining methodologies have been proposed to analyze large amounts of gene expression data. Most of these techniques can be broadly classified as Cluster analysis and Classification techniques. These techniques have been widely used to identify groups of genes sharing similar expression profiles and the results obtained so far have been extremely valuable. [3, 5, 7] However, the metrics adopted in these clustering techniques have discovered only a subset of relationships among the many potential relationship possible between the transcripts [9]. Clustering can work well when there is already a wealth of knowledge about the pathway in question, but it works less well when this knowledge is sparse [11]. The inherent nature of clustering and classification methodologies makes it less suited for mining previously unknown rules and pathways. We propose using another technique - **Association Rule Mining** for mining microarray data and it is our understanding that Association-rule mining can mine for rules that will help in discovering new pathways unknown before.

The efficiency tradeoffs between saving space and CPU (machine "thinking" time) are explored. Examples of the level of possible space savings are presented. The Goal is to provide fundamental knowledge in order to encourage deliberate consideration of the COMPRESS: option. Storage space and accessing time are always serious considerations when working with large data sets. compression, provide you with ways of decreasing the amount of space needed to store these data sets and decreasing the time in which observations are retrieved for processing. It offers several methods for decreasing data set size and processing time, considers when such techniques are particularly useful, and looks at the tradeoffs for the different strategies. Data set compression is a technique for “squeezing out” excess blanks and abbreviating repeated strings of values for the purpose of decreasing the data set’s size and thus lowering the amount of storage space it requires. Due to an increased awareness about data mining, text mining and big data applications across all domains the value of

data has been realized and is resulting in data sets with a large number of variables and increased observation size[15]. Often it takes a lot of time to process these datasets which can have an impact on delivery timelines. When there is limited permanent storage space, storing such large datasets may cause serious problems. Best way to handle some of these constraints is by making a large dataset smaller, by reducing the number of observations and/or variables or by reducing the size of the variables, without losing valuable information.

II. Related Work

The Apriori [1] algorithm is one of the classical algorithms in the association rule mining. It uses simple steps to discover frequent itemsets. Apriori algorithm is given below, Lk is a setof k-itemsets. It is also called large k-itemsets. Ck is a set of candidate k-itemsets. How to discover frequent itemsets? Apriori algorithm finds out the patterns from short frequent itemsets to long frequent itemsets. It does not know how many times the process should take beforehand. It is determined by the relation of items in a transaction. The process of the algorithm is as follows: At the first step, after scanning the transaction database, it generates frequent 1-itemsets and then generates candidate 2-itemsets by means of joining frequent 1-itemsets. At the second step, it scans the transaction database to check the count of candidate 2-itemsets [1]. It will prune some candidate 2-itemsets if the counts of candidate 2-itemsets are less than predefined minimum support. After pruning, the remaining candidate 2-itemsets become frequent 2-itemsets which are also called large 2-itemsets. It generates candidate 3-itemsets by means of joining frequent 2-itemsets. Therefore, CK is generated by joining large (K-1)-itemsets obtained in the previous step. Large K itemsets are generated after pruning. The process will not stop until no more candidate itemset is generated.

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)     C_k = apriori-gen(L_{k-1});  // New candidates
4)     forall transactions t ∈ D do begin
5)        C_t = subset(C_k, t);  // Candidates contained in t
6)        forall candidates c ∈ C_t do
7)           c.count++;
8)     end
9)     L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

Since most data occupy a large amount of storage space, it is beneficial to reduce the data size which makes the data mining process more efficient with the same results. Compressing the transactions of databases is one way to solve the problem. [1] Proposed a new approach for processing the merged transaction database. It is very effective to reduce the size of a transaction database. Their algorithm is divided into data preprocess and data mining. Sub-process transforms the original database into a new data representation. It uses lexical symbols to represent raw data. Here, it's assumed that items in a transaction are sorted in lexicographic order. Another sub-process is sorting all the transactions to various groups of transactions and then merges each group into a new transaction. For example, *T*1= {A, B, C, E} and *T*2 = {A, B, C, D} are two transactions. *T*1 and *T*2 are merged into a new transaction *T*3= {A2, B2, C2, D1, E1}.

```
// Phase one
  D^M = compressed database
  L_1^M = {large 1-itemsets in compressed database};
  for(k = 2; L_{k-1}^M ≠ Ø; k++) do begin
    C_k^M = merging-gen(L_{k-1}^M);
    forall transactions T_M^* ∈ D^M do begin
      C_t^M = subset(C_k^M, t^*);  // Candidate contained in t^*
       forall candidates c^* ∈ C_t^M do begin
          c^*.count = c^*.count + min-frequency(c^*); // the smallest item frequency in Candidate
    end
      L_k^M = {c^* ∈ C_k^M | c^*.count ≥ minsup}
    end
    Answer = ∪_k L_k^M;
// Phase two
D = original database
    forall transactions T_M ∈ D do begin
        L_t^M = subset(L_k^M, t) // Large itemset contained in t
    forall large itemsets l^* ∈ L_k^M do begin
        l^*.count++;
    end
    L_k = {l^* ∈ L_k^M | l^*.count ≥ minsup}
    Answer = ∪_k L_k;
```

II. Proposed Method

The description of the proposed algorithm focuses on compressing the dataset through building a quantification table. The dataset is sorted in lexicographical order and then this sorted dataset is use to create desired number of groups of the dataset, now we built the quantification table for each group and then by reading the postfix items from the table we get the compressed dataset. Finally, an example is provided to show the processes of our method. To simplify the description, it assumes the items in each transaction are presented in a lexicographical order.

Procedure:
Step 1: Read the original database.
Step 2: Sort the database in lexicographical order.
Step 3: Group the sorted database.
Step 4: Identify maximum length Transaction in each groups.
Step 5: Built quantification table
Step 6: Read postfix items from each quantification table
Step 7: Combine postfix items in table.
Step 8: Compressed datasets.
Step 9: Discover frequent itemset

Figure 1 shows an example of the proposed algorithm. We have a dataset shown in Figure (a) with transaction id and itemsets. Figure (b) shows the sorted dataset.

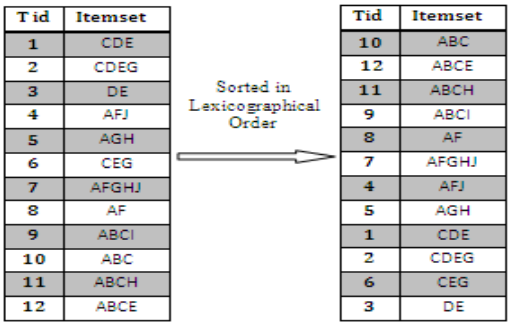

Figure 1: Example of Proposed Algorithm

Now the given sorted dataset is grouped into desired number of groups as shown in Figure 3 with group id and item set.

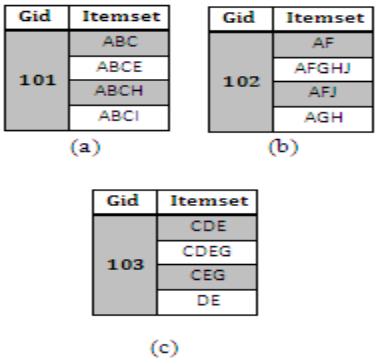

Figure 2: Groups of sorted dataset

Now we built quantification table for all these groups .The approach starts working from the left –most item, called prefix item. After finding the length of the input transaction as *n*, it records the count of the item sets appearing in the transaction under the respective entries of length Ln, Ln-1,.. L1. A quantification table is composed of these entries where each Li contains a prefix-item and its support count. Quantification table of each group is shown in
Figure 3.

L4	L3	L2	L1
A3	A4	A4	A4
	B3	B4	B4
		C3	C4
			E1
			I1

(a)

L5	L4	L3	L2	L1
A1	A1	A3	A4	A4
	F1	F1	F2	F3
		G1	G2	G2
			H1	H2
				J2

(b)

L4	L3	L2	L1
C1	C3	C3	C3
	D1	D3	D3
		E2	E4
			G2

(c)

Figure 3: Quantification Tables

Now by reading the postfix items of each quantification table we get the compressed dataset.

G id	Compressed Dataset
101	A4 B4 C4 E1 H1 I1
102	A4 F3 G2 H2 J2
103	C3 D3 E4 G2

Figure 4: Compressed Dataset

```
                Array of transactions,
p: set of items,
smin: int) : int
var i: item;
s: int;
n: int;
b; c; d: array of transactions;
begin
n := 0;
while a is not empty do
b := empty; s := 0;
i := a[0].items[0];
while a is not empty
and a[0].items[0] = i do
s := s + a[0].wgt;
remove i from a[0].items;
if a[0].items is not empty
then remove a[0] from a and append it to b;
else remove a[0] from a; end;
c := b; d := empty;
 while a and b are both not empty do  merge step
if a[0].items > b[0].items
then remove a[0] from a and append it to d;
else if a[0].items < b[0].items
then remove b[0] from b and append it to d;
else b[0].wgt := b[0].wgt +a[0].wgt;
remove b[0] from b and append it to d;
remove a[0] from a;
end;
end;
while a is not empty do
remove a[0] from a and append it to d; end;
while b is not empty do
```

Figure 5: Proposed Algorithm

Figure 5 describes the proposed algorithm. The overall architecture of the algorithm is shown in the figure 6.

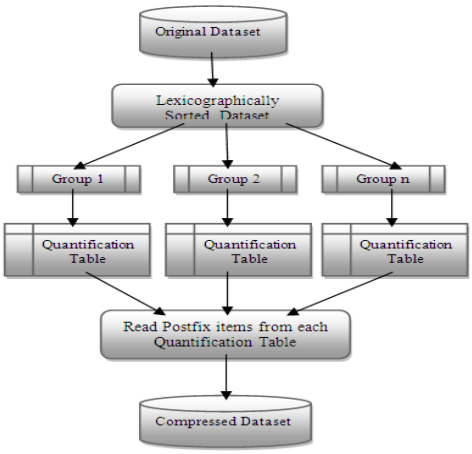

Figure 6: Architecture of Proposed Algorithm

III. Experimental Results

We implement the algorithm in Microsoft Visual Studio 2010 to evaluate the performance of our design and all experiments run on a PC of Intel Pentium 4 3.0GHz processor with DDR 400MHz 4GB main memory. Synthetic datasets are generated by using the items for our experiment. To the best of our knowledge, no research work has been dedicated to discovering frequent items for large data compression. We compare our approach with mining merge approach to show its effectiveness for the overall system performance evaluation. The table I representing the minimum support and execution time for mining merge and proposed algorithm.

TABLE IAlgorithms Comparison

Minimum support	Time taken to execute (In seconds) **Mining merge Algorithm**	Time taken to execute (In seconds) **Proposed Algorithm**
2	165	117
3	141	95
4	125	83
5	107	65

The performance of algorithms are analyzed in our experiment from Fig 6, it shows that the proposed algorithm performs better than the mining merge algorithm because it is possible to reduce more I/O time in the proposed algorithm. In general, the performance of using a quantification table is better than without using it. The proposed algorithm takes less time to compress the data than the mining merge algorithm for the varying minimum support.

Graph representing the comparison of mining merge and proposed algorithm when minimum support varying. Algorithms are analyzed on the basis of minimum support and execution time. The graph shows that for every value of support the proposed algorithm takes less time to execute than the mining merge algorithm. Graph shows that performance of proposed algorithm is better than mining merge algorithm.

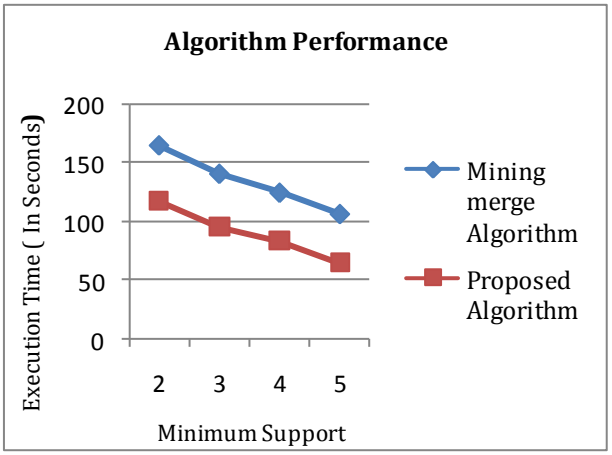

Figure 7: Proposed Algorithm

IV. Conclusion

We have proposed an innovative approach to generate compressed dataset for efficient frequent data mining. The effectiveness and efficiency are verified by experimental results on large dataset. It can not only reduce number of transactions in original dataset but also improve the I/O time required by dataset scan and improve the efficiency of mining process.

References

[1]. Jia -Yu Dai, Don – Lin Yang, Jungpin Wu and Ming-Chuan Hung, " Data Mining Approach on Compressed Transactions Database" in PWASET Volume 30, pp 522-529, 2010.

[2]. M. C. Hung, S. Q. Weng, J. Wu, and D. L. Yang, "Efficient Mining of Association Rules Using Merged Transactions," in WSEAS Transactions on Computers, Issue 5, Vol. 5, pp. 916-923, 2008.

[3]. D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, "MAFIA: A maximal frequent itemset algorithm," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, pp. 1490-1504, 2008.

[4]. D. Xin, J. Han, X. Yan, and H. Cheng, "Mining Compressed Frequent-Pattern Sets," in Proceedings of the 31st international conference on Very Large Data Bases, pp. 709-720, 2007.

[5]. G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, pp. 1347-1362, 2005.

[6]. M. Z. Ashrafi, D. Taniar, and K. Smith, "A Compress-Based Association Mining Algorithm for Large Dataset," in Proceedings of International Conference on Computational Science, pp. 978-987, 2003.

[7]. Ashrafi and K. Smith, "Data Compression-Based Mining Algorithm for Large Dataset," in Proceedings of International Conference on Computational Science, 2003.

[8]. D. I. Lin and Z. M. Kedem, "Pincer-search: an efficient algorithm for discovering the maximum frequent set," IEEE Transactions on Knowledge and Data Engineering, Vol. 14, pp. 553-566, 2002.

[9]. E. Hullermeier, "Possibilistic Induction in Decision-Tree Learning," in Proceedings of the 13th European Conference on Machine Learning, pp. 173-184, 2002.

[10]. Cheung, W., "Frequent Pattern Mining without Candidate generation or Support Constraint." Master's Thesis, University of Alberta, 2002.

[11]. Huang, H., Wu, X., and Relue, R. Association Analysis with One Scan of Databases. Proceedings of the 2002 IEEE International Conference on Data Mining. 2002.

[12]. Beil, F., Ester, M., Xu, X., Frequent Term-Based Text Clustering, ACM SIGKDD, 2002

[13]. Zaki, M. J., Parthsarathy, S., Ogihara, M., and Li, W. New Algorithms for Fast Discovery of Association Rules. KDD, 283-286. 1997. Agarwal, R., Aggarwal, C., and Prasad, V.V.V. 2001

[14]. Goulbourne, G., Coenen, F., and Leng, P. H. Computing association rule using partial totals. In Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, 54-66. 2001.

[15]. Pei, J., Han, J., Nishio, S., Tang, S., and Yang, D. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proc.2001 Int.Conf.on Data Mining. 2001.

[16]. Orlando, S., Palmerini, P., and Perego, R. Enhancing the Apriori Algorithm for Frequent Set Counting. Proceedings of 3rd International Conference on Data Warehousing and Knowledge Discovery. 2001.

[17]. Grahne, G., Lakshmanan, L., and Wang, X. 2000. Efficient mining of constrained correlated sets. In Proc. 2000. Int. Conf. Data Engineering (ICDE'00), San Diego, CA, pp. 512–521.

[18]. Dong, G. and Li, J. 1999. Efficient mining of emerging patterns: Discovering trends and differences. In Proc. 1999 Int. Conf. Knowledge Discovery and Data Mining (KDD'99), San Diego, CA, pp. 43–52.

[19]. Bayardo, R.J. 1998. Efficiently mining long patterns from databases. In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98), Seattle, WA, pp. 85–93.

[20]. D. W. L. Cheung, S. D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," in Proceedings of the 15th International Conference on Database Systems for Advanced Applications, pp. 185-194, 1997.

[21]. Brin, S., Motwani, R., and Silverstein, C. 1997. Beyond market basket: Generalizing association rules to correlations. In Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'97), Tucson, Arizona, pp. 265–276.

[22]. Brin, S., Motwani, R., Ullman Jeffrey D., and Tsur Shalom. Dynamic itemset counting and implication rules for market basket data. SIGMOD. 1997.

[23]. U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," Communications of the ACM, Vol. 39, pp. 27-34, 1996.

[24]. Piatetsky-Shapiro, G., Fayyad, U., and Smith, P., "From Data Mining to Knowledge Discovery: An Overview," in Fayyad, U., Piatetsky-Shapiro, G., Smith, P., and Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining AAAI/MIT Press, 1996, pp. 1-35.

[25]. Rakesh Agrawal and John C. Shafer, "Parallel Mining of Association Rules", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 962-969,December 1996