International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 514–521

Horizontal Format Data Mining with Extended Bitmaps

Buddhika De Alwis[1], Supun Malinga[2], Kathiravelu Pradeeban[3], Denis Weerasiri[4], Shehan Perera[5]

[1,2,3,4] WSO2 Inc., Sri Lanka.
[5] Department of Computer Science and Engineering,
University of Moratuwa, Sri Lanka.

[1]*buddhikac@wso2.com*
[2]*supunm@wso2.com*
[3]*pradeeban@wso2.com*
[4]*denis@wso2.com*
[5]*shehan@uom.lk*

***Abstract*: Analysing the data warehouses to foresee the patterns of the transactions often needs high computational power and memory space due to the huge set of past history of the data transactions. With the fragmented data along with the current trend of distributed systems, most of the fundamental algorithms that are initially proposed to find the association among the itemsets in the data warehouses are inefficient either in throughput or the utilization of the resources.**

Apriori algorithm is a mostly learned and implemented algorithm that mines the data warehouses to find the associations. However, Apriori is generally not an optimized algorithm. More variations, improvements, and alternatives have been suggested to overcome the inefficiency of Apriori algorithm, either as a whole or to specific sets of data. In any case, a fraction of improvement in the algorithm often improves the mining considerably. Frequent item set mining with vertical data format has been proposed as an improvement over the basic Apriori algorithm, which mines the data sets of vertical form, opposed to the typical horizontal format data as in case of Apriori.

In this paper we are proposing an algorithm as an alternative to Apriori algorithm, which will use bitmap indices in conjunction with a horizontal format data set converted to a vertical format data structure to mine frequent itemsets leveraging efficiencies of bitmap based operations and vertical format data orientation.

***Keywords*:** Data mining, Association Rule, Apriori, Vertical format mining, Bitmap Indices, Data Analysis, Data Warehousing.

I. Introduction

Data Mining is an emerging concept or a tool in database concepts, which is young, yet powerful and promising [1], [2]. Knowledge discovery, prediction, clustering, and classifications through pattern recognition are some of the key design aspects of data mining. Most of the existing techniques on the associations are based on analysing the data warehouses - the existing bulk data of transaction history. Given the existence of a set of items, association rules enable the prediction of the existence of one or more other items based on the knowledge gathered by classifying the data warehouses.

Foreseeing the user behaviour from the customer analysis of the past years is a topic of interest in this user-oriented marketing era. Database Engineers and scientists have been working on the associations of the transactions and derived many algorithms for effective decision making. Most of the existing techniques are based on analysing the data warehouses - the existing bulk data of transaction history [1].

The choice of the algorithm to retrieve the relationship between the variables for a given application is a challenging task, which is often a compromise where the accuracy, efficiency, latency, throughput, and security matter, as resources are limited. Several algorithms leading to optimal and sub-optimal conclusion have been developed and practiced on the datasets to extract patterns and gather the association among the variables or items. Given the existence of a set of items, association rules enable the prediction of the existence of one or more other items based on the knowledge gathered by classifying the data warehouses.

Association rule learning is mostly explained by one of its common applications – retrieving the association between the items that customers purchase. As an analogy, it is referred as keeping track of the customers' baskets or market basket analysis [1]. Association rules are commonly used in mining web usage [3], intrusion detection [4], and bioinformatics [5].

Statistical bias caused by suggesting a hypothesis by non-representative data or a narrow sample to match the hypothesis is defined as data-snooping bias, which often leads to a wrong decision in scientific calculations which include a highly distributed network [6]. This leads to a wrong decision in calculations, hence these factors also should be taken in to consideration when choosing the algorithm.

Apriori algorithm [1] is considered the fundamental algorithm for mining for associations. Hybrid algorithms converging the classification and association rule mining have also been suggested [7]. With the explosive growth of the data base size, scalability for the data mining algorithms becomes crucial to be able to work with very large data sets effectively. In this paper, a hybrid frequent item-set mining method extending Apriori algorithm is proposed.

II. Preliminaries

Traditionally, the algorithms of data analysis assumed that the input database contains a relatively smaller number of records. It becomes impossible to deploy the databases fully in the main memory, with the explosion of growth in their size. Extracting data from the hard drive is considerably slower than accessing the data located in memory. Hence the data mining algorithms should be scalable to be able to work with very large databases effectively. An algorithm is called 'scalable', if sustained capacity of main memory, with an increase in the number of records in the input database, its execution time increases linearly. Hence efficient scalable algorithms for data mining in very large data sets are widely studied.

A. Market Basket Analysis

Finding frequent item sets plays an important role in data mining as the first step in determining association rules. Association rule learning is mostly explained by market basket analysis [1] – retrieving the association between the items that customers purchase. Here products or sets of items (item-sets) which occur in many transactions are found.

Table 1. Transactional data

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Each of the patterns of behaviour of buyers identified through this analysis can be used to place the goods in the shops or restructure pages in the catalogues.

A set consists of i goods, called i-set-element (i-item-set). The percentage of transactions containing a given set, called the support (provision) set. It is believed that in order for a set of interest, its support must be above a user-defined minimum, such sets are called 'frequent'. Table 1 [8] describes several transactions (T100, T200, ..), stored in a relational database. Corresponding column mentions the relevant list of item ids for the particular transaction. As an example “T200” transaction contains “I2” and “I4” item ids.

In frequent item-set mining, we derive rules based on two measurements called minimum support and the minimum confidence that reflect the usefulness and certainty of the discovered rules. Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold [1].

B. Algorithm Apriori

This algorithm defines several stages. At the i-th stage identifies all frequent i-element sets. Each stage consists of two steps: the formation of candidates (candidate generation) and the counting of candidates (candidate counting). At the step of forming the candidates, algorithm creates a lot of candidates from the i-element sets. At the step of counting candidates algorithm scans the database of transactions, computing support sets of candidates. After scanning, the algorithm discards the candidates, ensuring that less than a user-defined minimum and saves only the common i-element sets.

During the 1st phase of the selected set of candidates contains all 1-element sets. Algorithm calculates their support during the step counting candidates. Thus, after the first phase all frequent 1-element sets are known. Reasoning in a straight line, a candidate can "burn" all pairs of goods. However, Apriori reduces the number of sets of candidate sifting - a priori - those candidates who may not be frequent, based on information received at previous stages of the information on which of the sets are the most abundant. Screenings are based on the simple assumption that if the set is frequent, all its subsets must also be frequent. Thus, before the counting of candidates step algorithm can reject any candidate set, a subset of which is not frequent. This process is continued until the number of frequent n-item-sets becomes zero, where n determines the no. of children in the item-set.

Consider the database presented in Table 1. Suppose that the minimum support count threshold is 2. That is, to be a frequent item-set, there should be at least two transactions, consisted of the particular item-set. In the first stage, all the products individually are sets of candidates and counted during the counting step, the candidate. At the second stage, the candidate may be only a couple of items, each of which is frequently encountered. For example, initially all the sets of single items ({I1}, {I2}, {I3}, {I4} and {I5}) have a support count of 6, 7, 6, 2, and 2 respectively. So initially all five items become frequent item-sets.

Thus, the second stage of the algorithm will form the following list of sets of candidates. Table 2 shows the item sets along with their support counts. Now frequent 2-item-sets are {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4} and {I2,I5} as the other item-sets don't have the minimum support count.

Table 2. Item sets and the support counts

Itemset	*Sup. Count*
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Likewise this process is continued until the no of frequent n-item-sets become zero, where n determines the no. of children in the item-set.
Apriori counts not only the provision of all frequent sets, but also ensuring those sets of candidates, which could not be discarded a priori. The set of all sets of candidates, which are rare, but whose software calculates the Apriori algorithm, called the negative region (negative border). Thus, the set belongs to a negative area if it can not be attributed to the frequent, but so are its subsets.

C. Algorithm Optimization

Many optimizations to the primary data mining algorithms are proposed over the time. They focus on a narrow sub set of datasets or cater the mining of the data broadly.

1) Pruning
Mannila et al. proposed pruning techniques to reduce the size of candidate set [9] as an optimization. Pruning strategy for the algorithm is based on a characteristic: a frequent item-set is a set, if and only if all its subsets are frequent.

2) Dynamic hashing and pruning
Park et al. suggested a hash based algorithm (DHP), which generates candidate large itemsets efficiently, while reducing the size of the transaction data base [10]. Number of candidate itemsets generated by DHP is smaller than that generated using existing algorithms, making it efficient for large itemsets, specifically for the large 2-itemsets.

3) Parallel Data mining
Park et al., extended DHP into a parallel data mining algorithm, facilitating parallel generation of the candidate itemsets and parallel determination of large itemsets [11].

4) Transaction Reduction
Agrawal et al. proposed this algorithm to reduce the number of transactions scanned in the future iterations [12], [13]. A transaction that does not contain any frequent k-itemsets can not contain any frequent (K +1) itemsets, hence can be removed.

5) Partition
Savasere et al. designed a Partition-based highly parallel algorithm [14], which logically divides the data base into several disjoint blocks. Each considered separately a sub-block and it generates all the frequent sets. The generated frequency of collection is used to generate all possible frequent sets, and followed by the final calculation of the degree of support of these itemsets. Sub-block size is chosen to allow each sub-block be placed in the main memory, each stage scanned only once.
This algorithm is highly parallel, and can be allocated to each sub-block of a processor that generates a frequency set. Frequency set is resulting, after the end of each cycle. The processor to generate the overall communication between the candidate k-itemsets. Usually here the communication process is a major bottleneck in algorithm execution time; while on the other hand, each individual processor generates frequent set time is also a bottleneck. Other methods are shared between multiple processors in a hash tree to generate frequent sets. More information on the generated set of parallel frequency method in the literature [14] found.

6) Sampling
A subset of the sample is taken for mining in the sampling based mining techniques, in the algorithm proposed by Mannila et al. [9]. Toivonen further developed this algorithm [15], where the rules are produced with the first extracted sample from the whole dataset. The remaining of the dataset is used to authenticate the dataset that is chosen. Sampling based algorithms significantly reduced the I /O costs, nevertheless with inaccurate results often and distortions in the data, known as data-skew.

7) Dynamic itemset counting
Dynamic itemset counting technology [16] proposed by Brin et al. marks the starting point of the database divided into blocks. The results of algorithm need fewer database scans than Apriori.

8) Equivalence CLASS Transformation (ECLAT)
Algorithm developed by Zaki [17] is used to mine the data sets efficiently using vertical data formats.

9) Vertical format data mining
Apriori algorithm mines frequent patterns from a horizontal data format which represents the items categorized into particular transactions, where vertical data format represents data as transactions categorized into particular items.
Hence, for a particular item, there is a set of corresponding transaction ids. Instead of horizontal data format, Apriori can be extended to use vertical data format for efficient mining. Table III shows the transactional data represented in Table I, in the vertical format.
As can be seen from the Table 3, the vertical format data mining only has to parse the dataset once to get the itemsets. For the itemset generation from the 2nd itemset, it only needs to refer the previous itemset. This eliminates the need to parse through the dataset each time to count the frequency of itemsets, for each round. Hence, relative to the algorithms developed for the use on databases with the horizontal layout of data, the algorithms developed for the vertical representation tend to be more optimal.

Table 3. Transactional data represented in the vertical format

Item ID	*List of TIDs*	*Support Count*
I1	T100, T400, T500, T700, T800, T900	6
I2	T100, T200, T300, T400, T600, T800, T900	7
I3	T300, T500, T600, T700, T800, T900	6
I4	T200, T400	2
I5	T100, T800	2

10) VIPER
VIPER [18], [19] is a general purpose algorithm, with no special requirement for the underlying database, based on

the vertical format representation. VIPER also uses compressed bit-vectors (referred to as 'snakes') to store data, with optimized generation of snakes, intersection, counting, and storage. VIPER is one of the algorithms motivated by the vertical format mining, as vertical format mining is more efficient than its horizontal format mining.

III. Horizontal Format Data Mining with Extended Bitmaps

We propose the algorithm 'Horizontal Format Data Mining with Extended Bitmaps', for mining a data set in a horizontal format, by converting it to a data structure of vertical format along with bitmaps indices to store the itemset data.

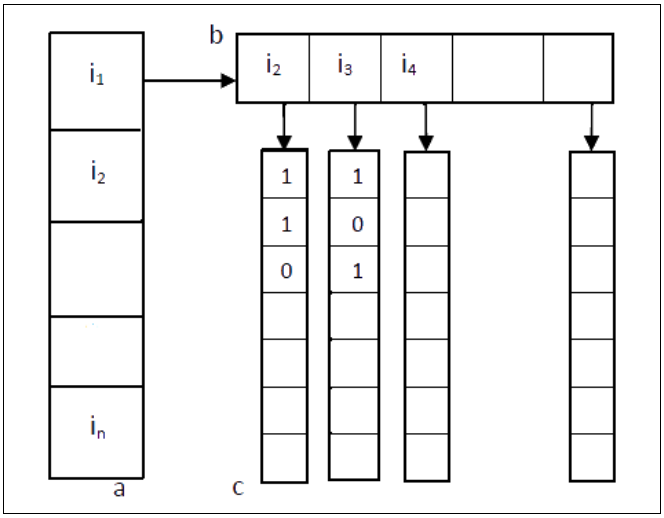
A. Parsing the dataset to create 1st itemset and bitmaps

The first and only parsing of the dataset is done here.

- Creating the master array.
- Creating bitmaps relevant to each item in master array.

Here bitmaps are used to store the individual bits compactly, exploiting the bit level parallelism effectively. In a bit map, 1 indicates the existence.

As in Figure 1, masterArray is an array of Item objects of all items in the dataset. Under each item, there is an array of other items occurring together with it in transactions which we call from, which we refer from now on as AssociatedItems. Under each AssociatedItem, we store a bit vector indicating the presence of the AssociatedItem for every transaction where the Item is found. Basically, length of bit vectors give the number of transactions that its root Item in masterArray is found. If the AssociatedItem is in the transaction, bit vector value for that transaction will be 1 (true), otherwise 0 (false).



a – masterArray; i_n – Item n;
b – associatedItemArray; c – BitSet

Figure 1. Main data storage structure of the extended vertical data mining.

B. Remove redundant associated items

Here for better memory utilisation, the redundant duplicate mappings (AssociatedItems) are pruned from the main data structure such as, Item i_1 mapped to AssociatedItem i_2 and Item i_2 mapped to AssociatedItem i_1 in the masterArray.

C. Pruning itemsets according to apriori property

Association mining is carried out solely in this step, in three major phases. The core logic of Apriori property is used; but implementation is done using the manipulation of bit vectors.

1) 1st frequency itemset extraction

The items not satisfying the minimum support are removed from the masterArray [1].

2) 2nd frequency itemset extraction

Here we iterate through the associatedItemArrays for each Item in masterArray. For each AssociatedItem, the cardinality of its bit vector is compared with the minimum support value. AssociatedItems having less than the minimum support are removed.

3)Extracting frequency itemsets above two

For mining frequency item sets of three or above, we start intersecting bit vectors of AssociatedItems, compare result bit vectors' cardinality with minimum support value, and remove the AssociatedItems as Apriori property suggests. This will take place for increasing frequency itemset values and stop when no more frequent item sets are present. In this recurring step, two main data structures [Figure 2, Figure 3] are used.

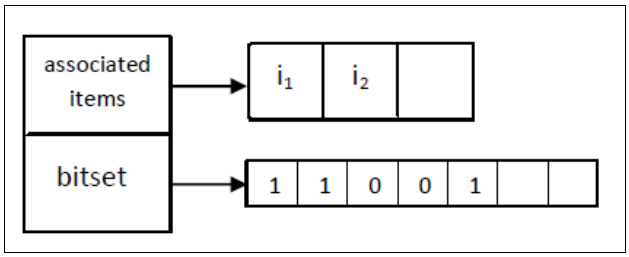


Figure 2. Pattern structure

Pattern structure Figure 2 is used to store each resultant bit vector from intersection of bit vectors and the itemset ids for those intersected bit vectors' belonging items. As Pattern is also a representation of item sets, we can use it as a frequent itemset itself.

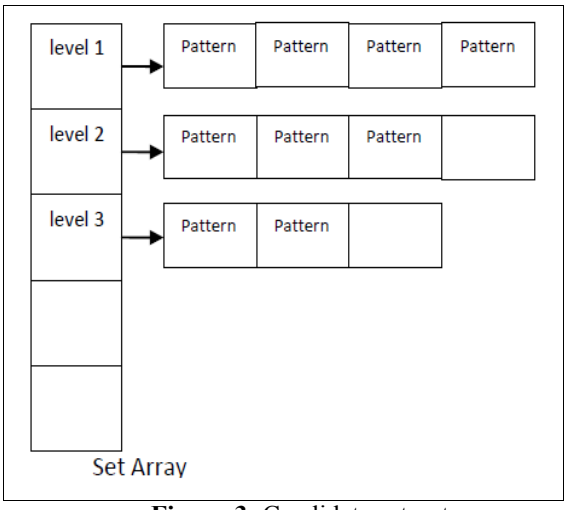


Figure 3. Candidates structure

Candidates structure Figure 3, is used to store the frequent item sets of each level of itemset size. Here each candidate level is an array of patterns. Each level indicates the

frequency itemset level. Hence frequent itemsets of length one will be under level one, frequent itemsets of length two will be under level two and so on. Once the above steps are completed, the frequent item sets are retrieved by iterating through the candidate structure.

IV. Analysis

We discussed the algorithm, "Horizontal format data mining with extended bitmaps". Here we are going to analyze the algorithm with a simple data set.

Table 4. Sample data set for the analysis

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I1, I2
T400	I2, I5

Table 4 shows the data set to be considered for this analysis. Here we are taking a very simple data set for the ease of analysis. In this analysis, we will use the proposed algorithm to find the frequent item sets with the minimum support = 2.

A. Building the master array and the bitmaps

We first build the master array with all the items in the sample data set. Then go through the first transaction
T100 = {I1, I2, I5}.
The associated item array is built with the items in the first transaction. I2 and I5 are linked to I1 in the master array, as the associated items in the associated item array for I1. Similarly, I5 is stored in the associated item array of I2, as shown in Figure 4.

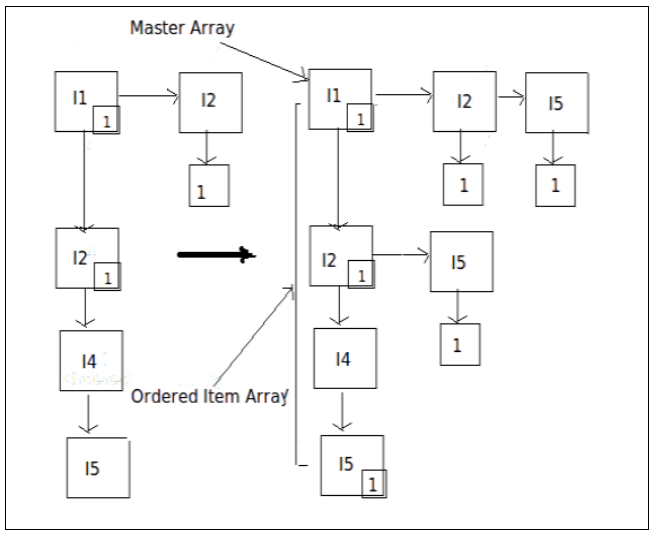


Figure 4. Data structure after going through T100

We avoid the redundancy by storing I1 in the associated item array of I2 for the same transaction, as {I1, I2} and {I2, I1} are identical for the transaction. Hence associated item arrays built with T100: {I1, I2, I5} and {I2, I5}.
Under each associated item in the respective associated item arrays, a bit is used to indicate the existence of the item.

The items in the transaction are counted and stored in the master array. Here I1, I2, and I5 are indicated by '1' in the master array, to show that the item exists once.
Similarly for the next transaction T200 = {I2, I4}, I2 is updated with the count of 2, and I4 with 1, in the master array.
Associated Item Array of I2 is updated. Here I4 is linked to I5 in the array of I2. {I2, I5, I4}.
A '0' is used under I4 of the associated item array of I2 to indicate that for T100, only I5 was associated to I2. Similarly the '0' for T200 under the bitmap of I5, in the associated item array of I2 show that I5 wasn't associated with I2.
From Figure 5, we can read that,
T100 = {I1, I2 (1), I5 (1)} = {I1, I2, I5}
T100 = {I2, I5 (1), I4 (0)} = {I2, I5}
T200 = {I2, I5 (0), I4 (1)} = {I2, I4}

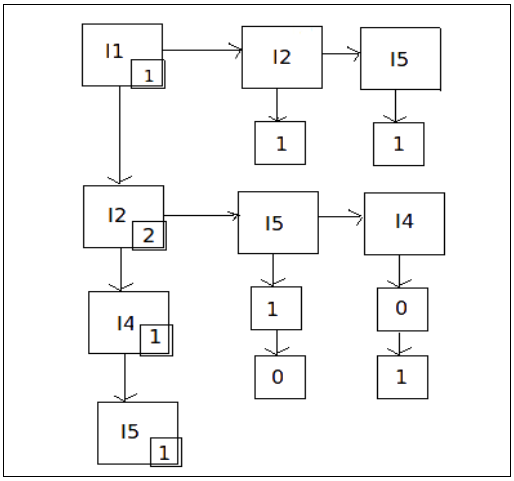


Figure 5. Data structure after T200

Figure 6 shows the data structure in our notation, updated with T300 = {I1, I2}.

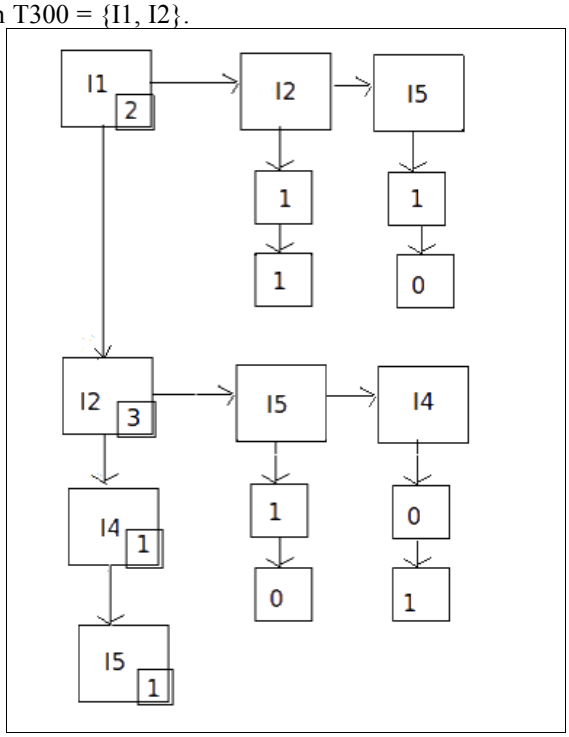


Figure 6. Data structure after T300

Figure 7. shows the final data structure, with T400, where all the master array is updated with all the transactions, and the relevant bit maps are created for each item in the master array. Reading the associated item arrays for each item in the master array shows the below.
I1, I2 (1), I5 (1) = {I1, I2, I5}
I1, I2 (1), I5 (0) = {I1, I2}
I2, I5 (1), I4 (0) = {I2, I5}
I2, I5 (0), I4 (1) = {I2, I4}
I2, I5 (1), I4 (0) = {I2, I5}

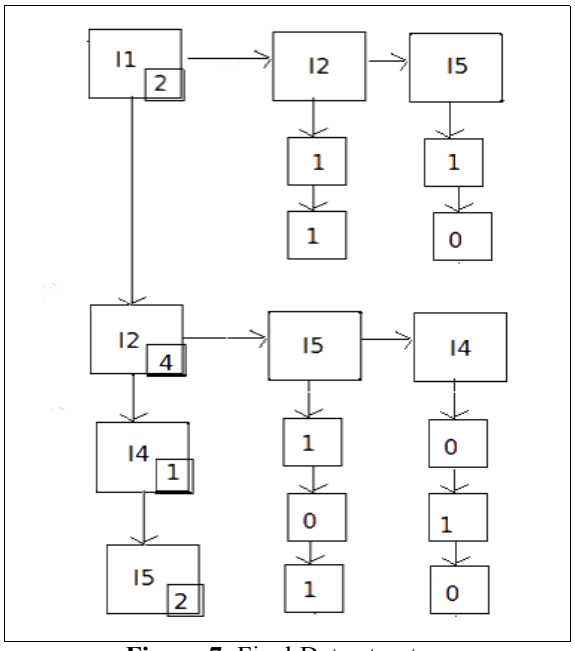


Figure 7. Final Data structure

B. Pruning itemsets

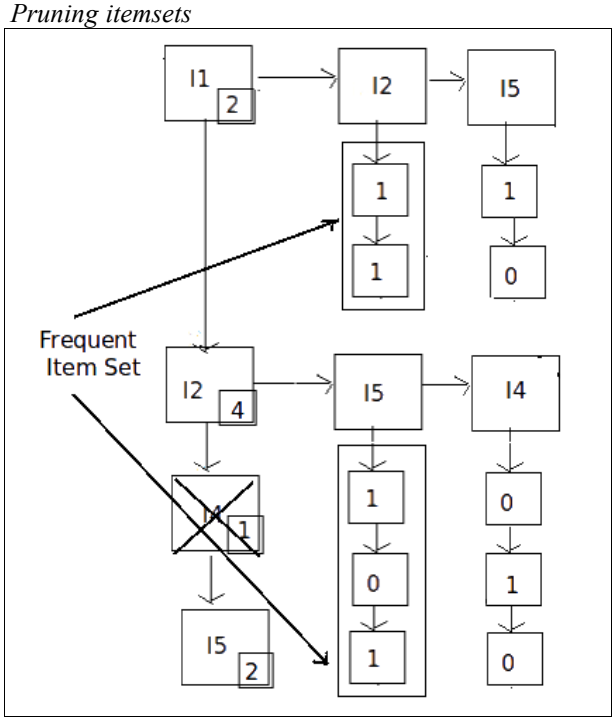


Figure 8. Frequent itemset

As shown in Figure 8, since our minimum support is 2, I4 is removed from further calculations, as it doesn't satisfy the minimum support level.

For each AssociatedItem, the cardinality of its bit vector is compared with the minimum support value. AssociatedItems having less than the minimum support are removed. Hence only {I1, I2} and {I2, I5} remain.

Now we have to extract the frequency itemsets above two. We start intersecting the bit vectors of AssociatedItems. Intersecting {I1, I2} with {I1, I5} and {I2, I5} with {I2, I4}. We compare result bit vectors' cardinality with minimum support value 2. Here, as depicted in Figure 9, the intersection shows that {I1, I2, I5} appears once, and {I2, I5, I4} doesn't appear as a frequent set.

Since these do not satisfy the minimum support level of 2, we stop here.

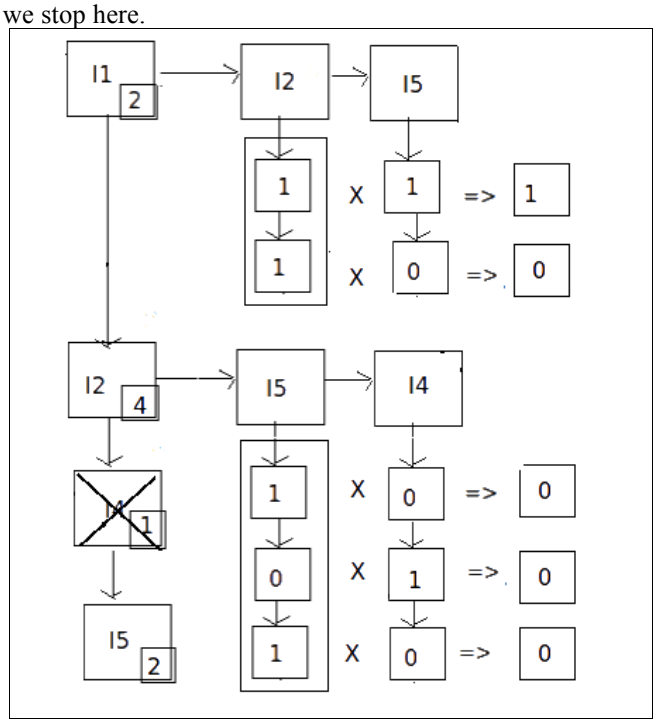


Figure 9. Final Frequent Item Sets

From the algorithm, we derive that {I1, I2} and {I2, I5} are the frequent sets that satisfy the minimum support.

V. Experimental Study

The algorithm was implemented and benchmarked in a system with 2 GB memory and 2.5 Ghz core 2 Duo Processor, against an implementation of Apriori algorithm for the datasets chosen from Frequent Itemset Mining Dataset Repository [20], along with different values of minimum support, as both the algorithms mine the datasets in horizontal form. Here both Apriori and the Horizontal Format Data-mining with Extended Bitmaps algorithm are designed for the databases having the horizontal layout.

Table 5. T= 10; I = 4; D = 10K

Minimum Support (%)	T10I4D10K	
	Apriori (sec)	HFDM-EB (sec)
0.75	1934.4	10.6
1	1365.4	10.5
2	238.4	10.4
5	1.6	9.6

Table 5, Table 6, and Table 7 compare the performance of the algorithms. The minimum support vs time taken are plotted for (T = 10, I = 4, D = 10K), (T = 10, I = 4, D = 100K), and (T = 40, I = 10, D = 100K), which are shown in Figure 10, Figure 11, and Figure 12 respectively. Time is shown in the log scale.

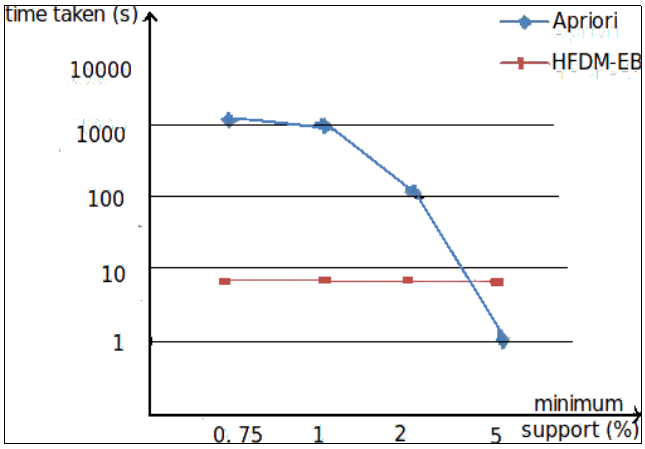


Figure 10. T= 10; I = 4; D = 10K

Table 6. T = 10; I = 4; D = 100K

Minimum Support (%)	T10I4D100K	
	Apriori (sec)	HFDM-EB (sec)
0.75	21039.1	134.2
1	12600.5	132.1
2	2365.1	131.7
5	1.3	132.1

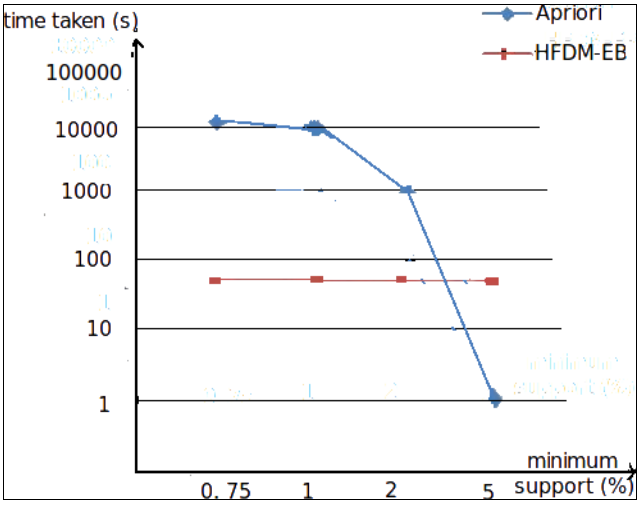


Figure 11. T= 10; I = 4; D = 100K

Table 7. T = 40; I = 10; D = 100K

Minimum Support (%)	T40I10D100K	
	Apriori (sec)	HFDM-EB (sec)
5	232	10.9
10	19	10.8
20	0.5	10.6
40	0.3	10.4

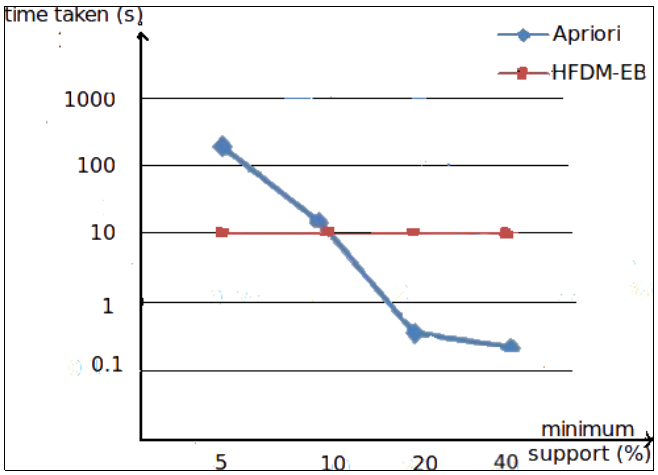


Figure 12. T = 40; I = 10; D = 100K

The time taken to build the bitmap is independent of the number of frequent item sets. But Apriori's time drops drastically when the number of frequent items is low. Thus Apriori was having a higher performance at high support levels where number of frequent item sets found is low.

VI. Conclusion and future work

Large companies for decades accumulated data on their customers, suppliers, products and services. Due to high rate of development of e-commerce working in Web start-ups can turn into a huge enterprise in a matter of months, rather than something those years. And, as a consequence, will grow rapidly and their databases.

Data mining, also called 'knowledge discovery in databases' [21] provides organizations with the tools developed to analyse the large collection of information to find trends, patterns and relationships that can help in making strategic decisions. In this paper we have proposed an efficient algorithm for Association Rule Mining, which recovers the associations as the Apriori on a data set in horizontal format, utilizing the bitmaps.

We have implemented the algorithm in Java, which may be more efficient, if implemented in C or a lower level language, so that we can control the memory allocation, in the most optimal way for the algorithm, as we want.

For each data item, a bitmap is created for each associated item. If there are n associated items for a data item, then the number of candidate sets generated is n(n-1). So there can be redundant bitmaps created for the same data item pairs. Currently, redundant pairs are pruned after creating the vertical format. But, if there is a dynamic pruning mechanism to prune redundant data item pairs while creating the vertical format memory can be well optimized.

The algorithm lends well to Map Reduce like distributed data mining since mining of each data item is independent of others. Each master array index is self contained, and hence can be mined in parallel. So this algorithm can be enhanced to work in a distributed environment with or without a shared memory. Here data structure generation becomes the Map phase, where the result accumulation becomes the Reduce phase, as in Map-Reduce.

For some larger data sets that are having many items per transaction, the algorithm fails to withstand due to utilizing prohibitive amount of memory. It can be mitigated by using

compressed bitmap [22] implementation instead of plain bitmaps, so the memory is utilized better.

References

[1] Jiawei Han and Micheline Kamber. “Data Mining, Concepts and Techniques,” 2nd Edition, 2006.

[2] Abraham Silberschatz, Henry F.Korth, and S.Sudarshan. “Database System Concepts,” 5th Edition. McGraw-Hill, Inc., New York, San Francisco, Washington, DC, 2005.

[3] Jaideep Srivastava , Robert Cooley , Mukund Deshpande, Pang-Ning Tan. “Web Usage Mining: Discovery and Applications of Usage,” Department of Computer Science and Engineering, University of Minnesota. ACM SIGKDD Explorations Newsletter, v.1 n.2, January 2000.

[4] Wenke Lee , Salvatore J. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th conference on USENIX Security Symposium*, p.6-6, January 26-29, 1998, San Antonio, Texas.

[5] Elisabeth Georgii, Lothar Richter, Ulrich Rückert and Stefan Kramer. “Analyzing microarray data using quantitative association rules,” Bioinformatics 2005, 21(Suppl 2):ii123-ii129.

[6] Ioannidis, John P. A. (August 30, 2005). "Why Most Published Research Findings Are False". PLoS Medicine (San Francisco: Public Library of Science) 2 (8). doi:10.1371/journal.pmed.0020124. ISSN 1549-1277.

[7] Behrouz Minaei-Bidgoli1, William F. Punch. “Using Genetic Algorithms for Data Mining - Optimization in an Educational Web-based System,” Genetic Algorithms Research and Applications Group (GARAGe). Department of Computer Science & Engineering. Michigan State University. Online: www.lon-capa.org/papers/v90-gapaper.pdf *[Accessed: 25th December, 2009]*

[8] Wei-Qing Sun, Cheng-Min Wang, Tie-Yan Zhang, and Yan Zhang. 2009. “Transaction-item association matrix-based frequent pattern network mining algorithm in large-scale transaction database.” W. Trans. on Comp. 8, 8 (Aug. 2009), 1327-1336.

[9] H. Mannila, H. Toivonen, and A. Verkamo. Efficient algorithm for discovering association rules. AAAI Workshop on Knowledge Discovery in Databases, pp 181-192, Jul. 1994.

[10] J.S. Park, M.S. Chen, and PS Yu. “An effective hash-based algorithm for mining association rules”. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp 175-186, May 1995.

[11] J.S. Park, M.S. Chen, and P.S. Yu. “Efficient parallel data mining of association rules,” *4th International Conference on Information and Knowledge Management*, Baltimore, Maryland, Nov. 1995.

[12] R. Agrawal, and R. Srikant. “Fast algorithms for mining association rules,” in *Proceedings of. 1994 Int. Conf. Very Large Databases (VLDB'94)*, Sep. 1994.

[13] J. Han and Y. Fu. “Discovery of multiple-level association rules from large databases,” in *Proceedings of. Int. Conf. Very Large Databases (VLDB'95)*, pp 402-431, Sep. 1995.

[14] A. Savasere, E. Omiecinski, and S. Navathe. “An efficient algorithm for mining association rules in large databases,” in *Proceedings of the 21st International Conference on Very Large Database*, pp 432-443, Sep. 1995.

[15] H. Toivonen. “Sampling large databases for association rules,” in *Proceedings of the 22nd International Conference on Very Large Database*, Bombay, India, pp 134-145, Sep. 1996.

[16] S. Brin, R. Motwani, JD Ullman, and S. Tsur. “Dynamic itemset counting and implication rules for market basket data,” in *Proceedings of ACM SIGMOD International Conference On the Management of Data*, pp 255-264, May 1997.

[17] M. J. Zaki. “Scalable algorithms for association mining,” IEEE Trans. Knowledge and Data Engineering, 12:372–390, 2000.

[18] Shenoy, P. and Haritsa, J. and Sudarshan, S. and Bhalotia, G. and Bawa, M. and Shah, D. (2000) “VIPER: A Vertical Approach to Mining Association Rules.” In: ACM SIGMOD International Conference on Management of Data (SIGMOD 2000), May 16-18, 2000 , Dallas, Texas.

[19] Shenoy, P. and Haritsa, J. and Sudarshan, S. and Bhalotia, G. and Bawa, M. and Shah, D. “Turbo-Charging Vertical Mining of Large Databases”.

[20] Frequent Itemset Mining Dataset Repository. [Online]. Available: http://fimi.cs.helsinki.fi/data/ [Accessed: 25th December, 2009]

[21] UM Fayyad et al., Eds. “Advances in Knowledge Discovery and Data Mining,” AAAI/MIT Press, Menlo Park, Calif., 1996. Advances in Knowledge Discovery and Data Mining, AAAI / MIT Press, Menlo Park, Calif., 1996.

[22] Kesheng Wu, Ekow Otoo and Arie Shoshani. “Optimizing Bitmap Indices with Efficient Compression.” ACM Transactions on Database Systems, v31, pages 1 - 38, March, 2006.

Author Biographies

Buddhika De Alwis received his B.Sc (Hons) in Computer Science and Engineering from University of Moratuwa in 2010. He is currently a software engineer at WSO2 Inc. His research interests include data mining, distributed and cloud computing.

A. Supun Malinga is a software Engineer at WSO2 Inc. He received his B.Sc (Hons) in Computer Science and Engineering from University of Moratuwa. His research interests include Distributed computing, SOA, middleware.

Kathiravelu Pradeeban obtained his B.Sc (Hons) in Computer Science and Engineering from the University of Moratuwa. He is a software engineer at WSO2 Inc., currently working with WSO2 Cloud Middleware Platform. His research interests include distributed computing, data mining, and SOA web services.

W.A. Denis Dhananjaya Weerasiri is a software engineer at WSO2 Inc. He received his B.Sc (Hons) in Computer Science and Engineering from University of Moratuwa in 2010. His research interests include cloud computing, distributed computing and business process management.

Amal Shehan Perera is a senior lecturer in computer science at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. He served as a senior visiting research scholar at North Dakota State University in 2009. He has multiple research publications and a co-owner of a US patent. His areas of interest include data mining, database systems, and software engineering. He won the ACM KDD Cup 2006 for data mining in collaboration with Prof. William Perrizo. He obtained his MSc and PhD from North Dakota State University under the guidance of Prof. Perrizo and his BSc from University of Colombo, Sri Lanka.