Asian Journal of Electrical Sciences
ISSN: 2249-6297 (P) Vol.8, No.1, 2019, pp.15-19

DOI: https://doi.org/10.51983/ajes-2019.8.1.2338

Mapper Association Rule Reducer Mining Method (MARRMM) for the Diagnosis of Heart Disease Using Hesitation Rule Set

P. Umasankar[1] and V. Thiagarasu[2]
[1]Research Scholar, Department of Computer Science, Manonmaniam Sundaranar University, Tamil Nadu, India
[2]Associate Professor, Department of Computer Science, Gobi Arts and Science College, Tamil Nadu, India
E-Mail: pus.msu2013@gmail.com
(Received 23 November 2018; Revised 12 December 2018; Accepted 15 January 2019; Available online 23 January 2019)

***Abstract* - Association rule is one of the primary tasks in data mining that discovers correlations among items in a transactional database. The majority of vertical and horizontal association rule mining algorithms have been developed to improve the frequent items discovery step which necessitates high demands on training time and memory usage particularly when the input database is very large. In this paper, in the third work, a novel hesitation rule generation method has proposed by blending the Map Reduce concept and Association Rule Mining. In this Mapper Association Rule Reducer Mining method has proposed to generate the hesitation rule set for giving the appropriate medication to the patient who are considered as not getting heart disease.**
***Keywords:* Association Rule Mining, Map Reduce, Heart Disease, Hesitation Rule Set Generation**

I. INTRODUCTION

Since the introduction of association rule [1], it continues to be an active research area in data mining and machine learning communities. Association rule discovery is an important task that finds relationships among items in a database. The classic application for association rule is market basket analysis in which business experts aim to investigate the shopping behaviour of customers in an attempt to discover regularities. Apriori based association rule algorithms, e.g. [2][3][4], operate in two primary steps: 1) Frequent items discovery and 2) Rule generation. In the first step, they discover frequent items in multiple iterations [5] [6]. Frequent items are items which occur in the database above a certain threshold denoted by the minimum support (*MinSupp*). In Apriori based algorithms, the discovery of frequent items is accomplished by level wise search where in the first level, they count the frequencies of items having length "1" (1- items), and determine whether or not they are frequent. Then, in each subsequent level, the algorithms start with items found to be frequent in the previous level in order to produce candidate items in the current level. Apriori-like techniques normally achieve a good computational performance whenever the size of the candidate items is small. However, in circumstances where a low support threshold is given and/or rules with many attributes are required, the expected number of candidate items may be massive [7]. Therefore, passing over the database multiple times to compute the candidate items supports is a significant overhead in terms of runtime and memory usage, regardless of the method in use.

While the second step that involves generating the rules from the set of discovered frequent items is straightforward, given that frequent items and their supports are known [8] [9]. The first step of finding frequent items is a relatively a harder problem that requires extensive computation and storage [10].

Map-Reduce (MR) [11] programming model has been adopted by many search enterprises such as Yahoo, Google, and Amazon to enable building petabyte data centres comprising hundreds of thousands of nodes. These data centres are of low hardware cost and with a software infrastructure to allow for parallel processing of the stored data. MR model provides a software infrastructure to simplify writing applications that can access and process this massive data. However, the cluster setup to get the optimum performance is not a trivial problem. It needs configuration of tens of setup and dynamic job parameters which affect every task execution. In addition to the configuration parameters, more research is now focusing on developing better scheduling algorithms for certain types of applications in MR.

II. ASSOCIATION RULE MINING

Association Rule Mining (ARM) is a method to regulate the manageable association rules for predictabilities among the items in comprehensive swapping info recorded. Let I=I1, I2, ….Im be a set of m targeted attributes and T be a transaction that contains a group of objects such that T→I. D is a database with exclusive transaction files. An association rule is a repercussion of type X→Y where X and Y are attributes and X ∩ Y = ø. X is known as the antecedent event and Y is known as the consequent. So, the two important principles for association rule mining are support (S) and confidence (C), which designates how often items are in the database and how many times the item sets are presented, correspondingly. The succeeding includes some key classifications in ARM.

Definition 1: Given a collection of n transactions T= {t1, …tn} and m items I = {i1,… im}, an association rule is expressed in the form:

$$X\ (Antecedent) \rightarrow Y\ (Consequent)$$

where $X, Y \subseteq I$, $X \cap Y = \Phi$, the left hand and right-side rules are the antecedents and the consequents, respectively.

Definition 2: Support(X) describes the proportion of transactions in T including X.
$Support\ (X)$= $(Number\ of\ Transaction\ Containing\ X)/(Total\ Number\ of\ Transactions)$, $X \in T$ (2)
Definition 3: If $Support\ (S) \geq Min\_support$ then S is known as frequent item set where $Min\_support$ is a threshold value described by users.
Definition 4: Transactions Count is $N=|T|$
Definition 5: Largest transaction length is E=Max (|ti|).
Definition 6: The rule confidence is the proportion of transactions in T including item set X which also include item set Y. Rules with both Support (X→Y) ≥ Min_Support and Confidence (X→Y) ≥ Min_Confidence are called strong rules. These thresholds values are described through customers.
$Confidence\ (X \rightarrow Y) = (Support\ (X \cup Y))/|Support\ (x)|$ (3)

III. MAP REDUCE PROGRAMMING

MapReduce is introduced by the Google [12] and under the MapReduce framework we can easily implements parallel, distributed algorithms. Google given it to Apache Software Foundation [13], now it is open source and developed under Apache Software Foundation. It is part of Hadoop [14][15] that can handle BIG Data [16] and large Applications. MapReduce [17] contains two components. One is map(), its work is filtering and sorting and other is reduce(), its work is summary operation like counting work. Input data is divided into different portion and then it sends to mapper, will do the filtering and sorting arrange data in (key, value) pair then it send to reducer. Reducer runs reduce function and calculate the output. Both map and reduce function written by the programmer as per their task.

These functions will be represented in this way:

$$Map: (key1, value1) \Rightarrow list\ (key2, value2)$$
$$Reduce: (key2, list(value2)) \Rightarrow (key3, value3)$$

IV. PROPOSED HESITATION RULE SET GENERATION FOR HEART DISEASE DIAGNOSIS BY MAPPER ASSOCIATION RULE REDUCER MINING METHOD (MARRMM)

Current association rule algorithms need to be redesigned to handle huge data problems to perform an efficient evaluation of the objectives and keep the quality of the rules obtained simultaneously. To accomplish that, an intellectual rule generation method has proposed that follows a MapReduce design to discover Association Rules from different proportions of the data. The challenge is how to discover quality association rules that represent the complete dataset through subset of rules obtained in different splits of the dataset. To fulfil this goal, the association rule algorithm is only executed in a subset (Map), thus, the fitness function evaluates only a subset of instances. The evolutionary process for each Map ends when a number of evaluations is reached (Neval) and a set of ARs is returned (RuleSet). Once all Maps are processed by the rule mining algorithm, the RuleSet from each Map is collected. Then, the global quality of the ARs of each RuleSet is evaluated using the entire dataset. To accomplish that, antecedent support, consequent support and rule support are partially calculated for each Map, and then they are aggregated to obtain the global evaluation that allow us to calculate the quality measures of the complete dataset. After that, all the ARs of each RuleSet are used to update an external set of rules henceforth named GlobalRulePool, that store the non-dominated solutions found for the entire dataset considering the quality measures previously calculated. These quality measures will be the objective functions used by the sequential association rule algorithm. Subsequently, the redundant ARs of the GlobalRulePool are removed.

Phase 1: The first phase, that follows a MapReduce design, is devoted to run an algorithm to obtain a set of ARs (RuleSet) for each subset in which the input dataset is divided.
Phase 2: The second phase, that also follows a MapReduce scheme, performs the support computation of the ARs obtained in the previous phase considering all the instances of the dataset. This phase aims at evaluating the quality of the rules over the entire dataset because it is necessary in the next phase. Note each RuleSet of the first phase has been only evaluated using the instances of the subset in which they have been trained. Therefore, in the second phase is necessary to calculate the quality of the rules over the entire dataset with the aim at selecting the best rules that present the best performance in the original dataset with all the instances and not only in the subset trained.
Phase 3: The third phase sequentially updates a global rule set denominated as GlobalRulePool that stores the non-dominated solutions found in the entire dataset. To accomplish that, the same measures used by the sequential algorithm to evaluate the quality of the ARs are calculated by the support obtained in the second phase.
Phase 4: The fourth phase builds a new rule set of GlobalRulePool following a MapReduce scheme.

The following gives the step by step procedure for the above-mentioned phases

1. Phase 1: The Step by Step Procedure of Retrieving all the Instances of the Map

Phase 1: The step by step procedure the retrieving all the instances of the Map

Split and runs the association rule algorithm in the instances of Map partition.

Input: *data that represents the dataset, each row is an Array of values.*

Output: *The rules discovered by the AR algorithm*

1.1: *RuleSet ←*

1.2: *map partitions instances ∈ data*

1.3: *{instances will contain all instances in each partition}*

1.4: *associationRules ← algorithm.run(instances)*

1.5: *end map*

2. Phase 2: Explains the Step by Step Procedure that Computes the Support of Each Association Rule

Mapper Stage

Input: *RuleSet a set of rules discovered by the association rule algorithm*

Output: *(key, value) pair, where key indicates ruleId of each QAR of RuleSet evaluated in the instance corresponding to the offset key and value contains the support of the antecedent (supA), support of the consequent (supC) and support of the rule (supR) of each rule in such instance.*

2.1: *supports ←*

2.2: *map partitions instances ∈ data*

2.3: *{instances will contain all instances in each partition}*

2.4: *for each Rule ∈ RuleSet do*

2.5: *{ruleId, {supA, supC, supR}} ← computeSupports(instances,Rule)*

2.6: *end for*

2.7: *end map*

Reducer Stage: It computing the support of the model:

Input: *supports a (key, value) pair, where key is the id of a rule and value is the support of the antecedent (supA), support of the consequent (supC) and support of the rule (supR) of such rule for each instance.*

Output*: (key, value) pair, where key is the id of each rule of RuleSet and value contains the global support of the antecedent, consequent and the rule considering all the instances.*

3.1: *Rules ← supports.reduce{(a, b) =>*

3.2: *a.supA + = b.supA*

3.3: *a.supC += b.supC*

3.4: *a.supR += b.supR*

3.5*:}*

Phase 3: Step by Step Procedure

Input: *(key, value) pair, where key is the id of each rule of RuleSet and value contains the global support of the antecedent, consequent and the rule considering all the instances.*
Output: *GlobalRulePool the external set of rules that contains the non-dominated solutions found for the entire dataset.*
3.1: *for RuleRS ∈ RuleSet do*
3.2: *if RuleRS satisfies Minimum Thresholds then*
3.3: *for RuleGP ∈ GlobalRulePool do*
3.4: *if ruleRS dominates ruleGB then*
3.5: *remove ruleGB from GlobalRulePool*
3.6: *add ruleRS to GlobalRulePool*
3.7: *end if*
3.8: *if ruleRS and ruleGB are non-dominated solutions then*
3.9: *add ruleRS to GlobalRulePool*
3.10: *end if*
3.11: *end for*
3.12: *end if*
3.13: *end for*
3.14: *GlobalRulePool ← removeRedundant(GlobalRulePool)*

Phase 4: The fourth Stage Builds a New Rule Set of a GlobalRulePool by Following a MapReduce Scheme

Input: *data that represents the dataset, each row is an Array of values.*
Output: *newData a new dataset with all uncovered instances.*
4.1: *newData ← data.filter{instance =>*
4.2: *keep ← TRUE*
4.3: *for Rule ∈ GlobalRulePool do*
4.4: *if instance is satisfied by Rule then*
4.5: *keep ← FALSE*
4.6: *end if*
4.7: *end for*
4.8: *keep*
4.9: *}*

V. RESULTS AND DISCUSSION

We have evaluated the proposed Mapper Association Rule Reducer Mining Method (MARRMM) on local standalone cluster installed in our machine. Algorithm is implemented by using Java and MapReduce 2.0 (YARN) library. Figure 1 depicts the execution time of the ARM and proposed Mapper Association Rule Reducer Mining Method (MARRMM) with various minimum support values for given original dataset. The original dataset has pre-processed (by using the proposed TFIE-PSFS) and classified by the ANN. From the proposed TFIE-PSFS [18], the original dataset has reduced and it is named as reduced dataset since it gives more accuracy and reduced error rates than the existing methods.

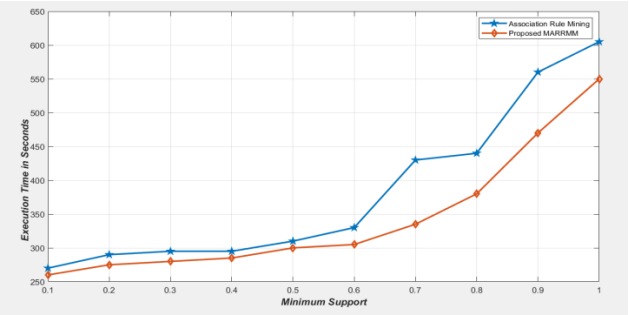

Fig. 1 Execution time of the proposed MARRMM and ARM with varying minimum support for original dataset

Figure 2 depicts the execution time of the ARM and proposed MARRMM with various minimum support value for obtained optimal dataset. From the Figure 2, it is clear that the execution time for optimal dataset is reduced than the *original* dataset with minimum support value.

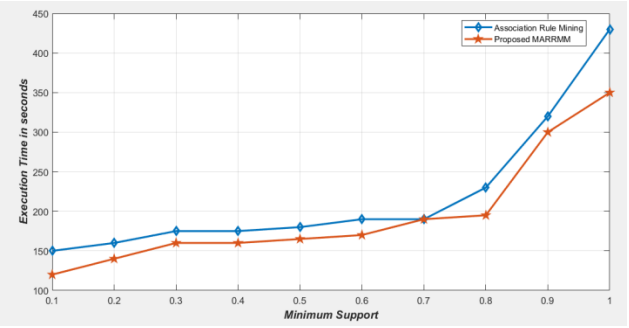

Fig. 2 Execution time of the proposed MARRMM and ARM with varying minimum support for optimal dataset

Figure 3 gives the performance analysis of the proposed MARRMM with using both original and optimal datasets. The Figure 3 represents the elapsed time (in seconds) for each phase in MapReduce, total and actual execution of the proposed method at the minimum support of 0.1. From the Figure 3, it is observed that the proposed algorithm with optimal dataset operates in less elapsed time and its total execution time is reduced than the proposed algorithm with original dataset. The total execution time is less than the actual execution time for proposed algorithm with optimal dataset than the proposed algorithm with original dataset.

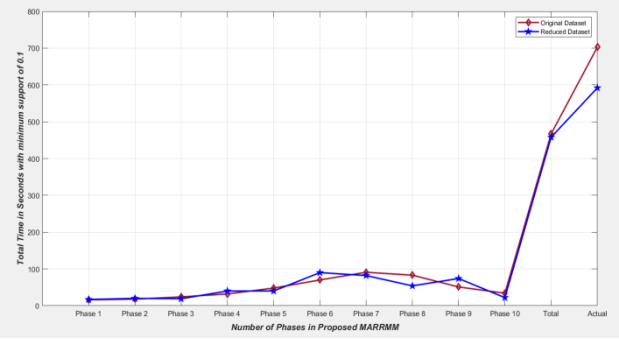

Fig. 3 Actual execution time, Elapsed time in seconds for each MR phase and total time (in seconds) of all phases with minimum support of 0.1

Figure 4 gives the performance analysis of the proposed MARRMM with using both original and optimal datasets. The Figure 4 represents the elapsed time (in seconds) for each phase in MapReduce, total and actual execution of the proposed method at the minimum support of 0.5. From the Figure 4, it is observed that the proposed algorithm with optimal dataset operates in less elapsed time and its total execution time is reduced than the proposed algorithm with original dataset. The total execution time is less than the actual execution time for proposed algorithm with optimal dataset than the proposed algorithm with original dataset.

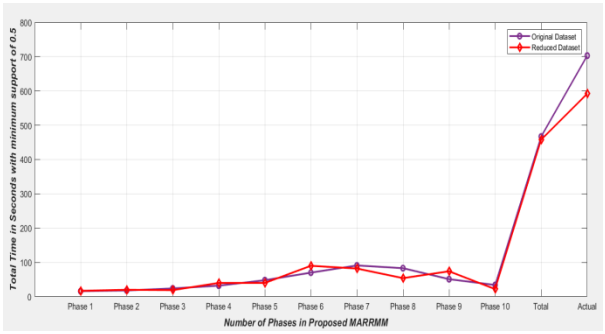

Fig. 4 Actual execution time, Elapsed time in seconds for each MR phase and total time (in seconds) of all phases with minimum support of 0.5

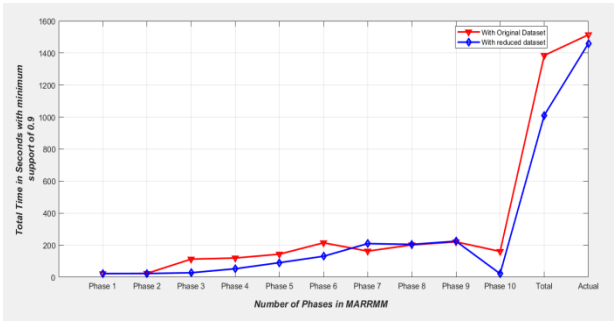

Fig. 5 Actual execution time, Elapsed time in seconds for each MR phase and total time (in seconds) of all phases with minimum support of 0.9

Figure 5 gives the performance analysis of the proposed MARRMM with using both original and optimal datasets. The Figure 5 represents the elapsed time (in seconds) for each phase in MapReduce, total and actual execution of the proposed method at the minimum support of 0.9. From the Figure 5, it is observed that the proposed algorithm with optimal dataset operates in less elapsed time and its total execution time is reduced than the proposed algorithm with original dataset. The total execution time is less than the actual execution time for proposed algorithm with optimal dataset than the proposed algorithm with original dataset.

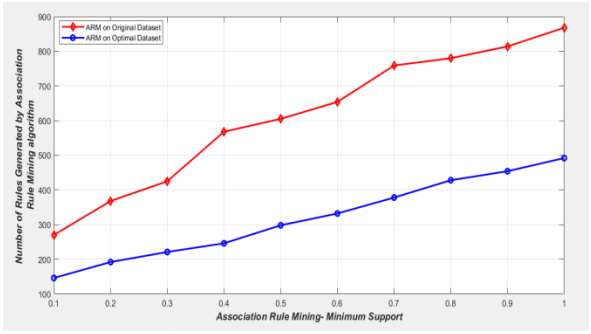

Fig. 6 Number of Rules obtained by ARM for Original Heart disease dataset and optimal dataset with various support value

Figure 6 depicts the number of rules generated by Association Rule Mining algorithm on original dataset and optimal dataset with various minimum support values. From the figure 6 it is clear that the optimal dataset gives least number of rules than the ARM on original dataset.

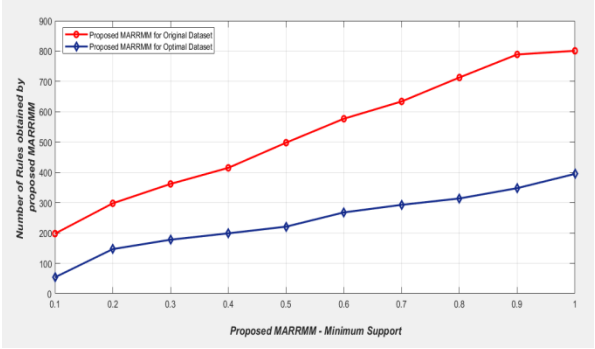

Fig. 7 Number of Rules obtained by proposed MARRMM for original dataset and optimal dataset with various minimum support value

Figure 7 depicts the number of rules generated by proposed MARRMM algorithm on original dataset and optimal dataset with various minimum support values. From the figure 7 it is clear that the optimal dataset gives least number of rules than the proposed MARRMM on original dataset.

VI. CONCLUSION

In this paper, Mapper Association Rule Reducer Mining Method (MARRMM) has proposed to generate the hesitation rule set from the reduced heart disease dataset. The results revealed that MARRMM is more efficient than Apriori in mining the frequent itemsets because of fast and parallel intersection among the itemsets in the dataset. The quality of the rules is ensured by MapReduce framework. The four stages in the framework used to give the global support rules for hesitation information to give appropriate medication to the patient. The effectiveness of the proposed MARRMM has evaluated in the result and discussion part with original heart disease dataset as well as reduced heart disease dataset.

REFERENCES

[1] Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami, "Mining association rules between sets of items in large databases", *Acm sigmod record*. ACM, Vol. 22, No. 2, 1993.

[2] Liu, Bing, Wynne Hsu, and Yiming Ma, "Mining association rules with multiple minimum supports", *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge discovery and data mining*, ACM, 1999.

[3] Yang, Xin Yue, Zhen Liu, and Yan Fu, "MapReduce as a programming model for association rules algorithm on Hadoop", *Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on* IEEE, 2010.

[4] Singh, Jaishree, Hari Ram, and J. S. Sodhi, "Improving efficiency of apriori algorithm using transaction reduction", *International Journal of Scientific and Research Publications*, Vol. 3, No. 1, pp. 1-4, 2013.

[5] Park, Jong Soo, Ming-Syan Chen, and Philip S. Yu. *An effective hash-based algorithm for mining association rules, ACM,* Vol. 24, No. 2, 1995.

[6] Yoon, Yongwook, and Gary G. Lee, "Text categorization based on boosting association rules", *Semantic Computing, 2008 IEEE International Conference on IEEE,* 2008.

[7] J. Mohammed Zaki, "Mining non-redundant association rules", *Data mining and knowledge discovery*, Vol. 9, No. 3, pp. 223-248.

[8] Lucchese, Claudio, Salvatore Orlando, and Raffaele Perego, "Fast and memory efficient mining of frequent closed itemsets", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 1, pp. 21-36, 2006.

[9] Lim, Tjen-Sien, Wei-Yin Loh, and Yu-Shan Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms", *Machine learning*, Vol. 40, No. 3, pp. 203-228, 2000.

[10] Ashrafi, Mafruz Zaman, David Taniar, and Kate Smith, "ODAM: An optimized distributed association rule mining algorithm", *IEEE distributed systems online*, Vol. 5, No. 3, 2004.

[11] Dhok, Jaideep, and Vasudeva Varma, "Using pattern classification for task assignment in mapreduce", *Proc. ISEC*. 2010.

[12] Dean, Jeffrey, and Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters", *Communications of the ACM*, Vol. 51, No. 1, pp. 107-113, 2008.

[13] [Online] Available at https://en.wikipedia.org/wiki/Apache_Software_Foundation

[14] Apache hadoop. http://hadoop.apache.org/2013.

[15] [Online] Available at https://developer.yahoo.com/hadoop/tutorial/

[16] K. Gordon, What is Big Data?. IT NOW, Vol. 55, No. 3, pp. 12- 13, 2013.

[17] Zaharia, Matei, *et al.,* "Improving MapReduce performance in heterogeneous environments", *Osdi*., Vol. 8. No. 4. 2008.

[18] P.Umasankar, V. Thiagarasu, "A Novel Thrice Filtered Information Energy based Particle Swarm Feature Selection for the Heart Disease Diagnosis", *International Journal of Pure and Applied Mathematics*, Vol. 119, No. 15, pp. 3485-3499, 2018.