Knowledge Extraction from Big Data using MapReduce-based Parallel-Reduct Algorithm

Tapan Chowdhury
Dept. of CSE
Techno India, Salt Lake
Kolkata, India
tapan2005cse@gmail.com

Susanta Chakraborty
Dept. of CST
IIEST, Shibpur
Howrah, India
susanta.chak@gmail.com

S. K. Setua
Dept. of CSE
University of Calcutta
Kolkata, India
sksetua@gmail.com

***Abstract*— Extraction of knowledge and predictive analysis are the new challenges for the rapidly growing large volume of data to make the right decision at right time. It is difficult to store, analyze and visualize such large data volume with its diversities with standard data mining tools. Hence, in this paper, we develop a MapReduce approach of a Parallel-Reduct algorithm based on the rough set theory for knowledge extraction. It performs data and task parallelism with the help of Map and Reduce functions to find minimum reduct. An extensive experimental evaluation shows that the proposed MapReduce-based parallel approach effectively processes big data on the Hadoop platform and it is more efficient than the sequential approach to extract knowledge from large data sets under different coarseness.**

Keywords—MapReduce; Reduct; Degree of Dependency; Positive region; Rough Set; Big Data.

I. INTRODUCTION

Knowledge extraction is used to acquire important information from large datasets. According latest surveys, 90% of the collected data in the universe have been generated in last two years [1] and the rate of data generation is of the order of 2.0 quintillion bytes per day. But traditional data mining techniques are not capable of handling this large volume of fast growing scientific and industrial data because of data storage and computational complexities. It is difficult to manage such dataset in a large space with a single machine [2]. In case of knowledge extraction, all attributes from a dataset are not actively participated in decision analysis so it is required to discard irrelevant and redundant attributes that increase the size of search space and make the computation more complex [3].

For decision-making situations with inconsistent, uncertain and imprecise information, a mathematical tool, Rough set, proposed by Pawlak in 1982[4] is becoming state-of-the-art. This theory has been successfully employed in several real-life domains like text and message processing, pharmacology, clinical diagnosis, prediction of financial market analysis, etc. [5, 6].

Rough set based MapReduce framework has been applied to various domains e.g. rough set approximation computation [7], knowledge acquisition [8], rough entropy computation [9] and in knowledge reduction [10] method. Different reduct generation algorithm [11] uses various concepts of rough set theory like dependency, positive region, and equivalence classes. Some of them require more processing times and attempt to determine a reduct set without producing all possible combinations of reduct sets.

Knowledge extraction from big data, based on positive region, discernibility matrix etc., cannot be done in a single memory machine. To overcome this limitation, in this paper a MapReduce-based parallel method is proposed. The experiment demonstrates that the proposed parallel method can efficiently process massive data sets using MapReduce, with significant less runtime and storage space.

The rest part of the paper is structured as follows: in Section II, we present the preliminary background of rough set theory, reduct and MapReduce framework. The proposed method is illustrated in Section III. Section IV shows experimental results and analysis with four data sets from GEO Dataset Repository and UCI Machine Learning Repository and it illustrates the effectiveness of proposed method. Finally, conclusion is drawn in Section V.

II. PRELIMINARIES

In this section, we introduce the elementary background of rough set theory, reduct and MapReduce framework.

A. Rough Set

Rough set theory is used as a mathematical tool for decision making using uncertain and imprecise data. This theory allows us to reduce the dimensions of datasets by removing the superfluous data. It extracts the knowledge in terms of hidden relationships between objects and their classes.

Definition 1. An Information System [12] can be represented as a pair $I = (U, A)$, where, U is the finite non-empty set of objects, called universe of discourse and A is the finite non-empty set of attributes. Each attribute $a \in A$ represents a function $f_a: U \rightarrow V_a$, where V_a is the domain of attribute a. Set of attributes is divided into two subsets $C \subset A$ and $D = A - C$, where C is the set of conditional attributes and D is the decision attribute.

Definition 2. For $P \subseteq C$ and $X \subseteq U$, the P-lower and P-upper approximations of X with respect to P are respectively defined as follows:

$$\underline{PX} = \{ x \in U \mid [x]_p \subseteq X \} \quad (1)$$

$$\overline{PX} = \{ x \in U \mid [x]_p \cap X \neq \varphi \} \quad (2)$$

where, $[x]_p$ is the equivalence class of P-indiscernibility relation [12].

Definition 3. Let $P, Q \subseteq A$, an equivalence relation over U, then a P-positive region [12] of Q can be defined as

$$POS_p(Q) = \cup_{X \subseteq U/Q} \underline{PX} \quad (3)$$

The P-positive region of X contains those objects that can be certainly classified in the set X.

The degree of dependency (k) of Q on P can be defined as

$$k = \gamma_P(Q) = \frac{|pos_P(Q)|}{|U|} \quad (4)$$

Q is totally and partially dependent on P, if $k=1$ and if $0 < k < 1$ respectively. For $k=0$, Q is independent of P.

Definition 4. A reduct [12] R is a minimal subset of conditional attributes C ($R \subseteq C$) with respect to decision attribute D, if $POS_R(D) = POS_C(D)$ and R is independent. It is a set of attributes that can fully characterize and classify the knowledge of universe which signifies that it can be possible to remove some redundant conditional attributes from the information system while retaining its basic properties. However finding an optimal reduct with a minimal number of attributes is NP-Hard [13]. For an Information System $I=(U, C, D)$, a reduct can be defined as follows:

$$RED(C) = R \subseteq C \mid \gamma_R(D) = \gamma_C(D), \forall B \subset R, \gamma_B \neq \gamma_C(D) \quad (5)$$

where, C is the set of conditional attributes and D is the set of decision attributes.

Definition 5. Let an information system $I = (U, C, D)$. Every $x \in U$, determines the sequence of decision rule induced by x such that $c_1(x),....,c_n(x) \rightarrow d_1(x),....,d_m(x)$ or in short $C \rightarrow_x D$.

Support [4] of the decision rule $C \rightarrow_x D$, denoted by $supp_x(C, D)$, can be defined as

$$supp_x(C, D) = | C(x) \cap D(x) | \quad (6)$$

Strength [4] of the decision rule $C \rightarrow_x D$, denoted by $\sigma_x(C, D)$, can be defined as

$$\sigma_x(C, D) = \frac{supp_x(C, D)}{|U|} \quad (7)$$

Certainty factor [4] of the decision rule $C \rightarrow_x D$, denoted by $cer_x(C, D)$ can be defined as

$$cer_x(C, D) = \frac{| C(x) \cap D(x) |}{| C(x) |} = \frac{\text{supp}_x(C, D)}{| C(x) |} \quad (8)$$

Coverage factor [4] of the decision rule $C \rightarrow_x D$, denoted by $cov_x(C, D)$, can be defined as

$$\text{cov}_x(C, D) = \frac{| C(x) \cap D(x) |}{| D(x) |} = \frac{\text{supp}_x(C, D)}{| D(x) |} \quad (9)$$

B. Rules Generation

LEM2 [13] (Learnable Evolution Model, Version 2) is a rule induction algorithm based on rough set theory. It generates certain and possible rule sets from decision table by computing the local covering from search space of attribute-value pairs. The main task of the LEM2 algorithm is to learn minimal set of categorized rule set.

C. MapReduce Framework

MapReduce [14] is a standard programming model that can handle large amount of structured, semi-structured and unstructured data by exploiting the ability of high parallel processing. It has been introduced by Google as a programming framework to analyze a large amount of data in parallel. This programming framework has three phases:

Map phase: In this phase, *mapper* function splits the data set into M map functions. Each *mapper* runs in parallel, accepts an input pair and generates a set of intermediary <*key,value*> pairs.

Combine and sort phase: In this phase, *combiner* function sorts and combines all the intermediary <*key, values*> related to the same intermediary key. It passes the results to the next phase.

Reduce phase: In this phase, *reducer* function takes intermediary <*key, value*> pair from the previous phase, and a computation algorithm has been applied to generate a list of values as the final result.

Fig.1 shows the flow of data and different phases of the MapReduce framework.

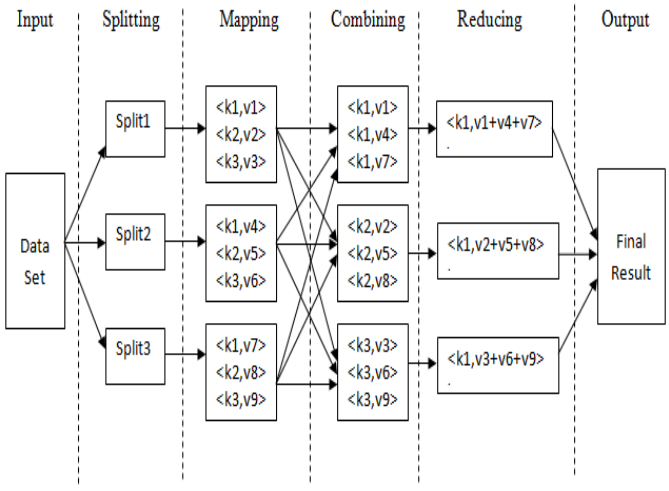

Fig. 1. Dataflow in MapReduce Framework

III. MAPREDUCE-BASED PARALLEL REDUCT ALGORITHM

A. Proposed Method

Parallel-Reduct Algorithm generates reduct in parallel that provides optimum solutions as well as minimizes the running time. In order to reduce the run time taken by sequential reduct generation algorithm to generate reduct set, we propose a parallel-reduct algorithm that follows MapReduce paradigm. In our approach, MapReduce programming model is used to extract knowledge from large data sets by generating minimal reduct.

Our MapReduce-based proposed approach is divided into following tasks:

- **Mapper:**
  - Read the input file from local file system and splits it into different blocks.
  - Upload the data blocks into HDFS.
  - Distribute the data with *{key,value}* pair.
  - Compute the cardinality of *{key,value}* pair
- **Combiner**
  - Determine the equivalent classes for each *{key, value}* pair.
  - Combine the *{key, value}*
- **Reducer**
  - Reduce the number of *{key,value}* pair with respect to same decisions.
- **Compute Reduct**
  - Calculate the degree of dependency of each reduced *{key,value}* pair based on positive region value of each attribute.
  - Find the attribute with a minimum degree of dependency value and include that attribute in Reduct set.
  - Compute the degree of dependency for all the combinations of reduct set element with other attributes until the degree of dependency is equal to 1 and include this combination in Reduct set.

Consider the decision table Table I, where *{a,b,c,d}* is the set of conditional attributes and *e* is the decision attribute.

TABLE I. DECISION TABLE

x∈U	*a*	*b*	*c*	*d*	*e*
1	1	5	10	6	1
2	1	5	11	6	2
3	2	5	10	6	3
4	2	5	11	6	4
5	3	5	10	6	5

Initially Reduct set is empty. So the decision table Table I can be mapped with respect to conditional attribute *{a,b,c,d}* shown in Fig. 2.

Consider Initially $R = \{\ \}$.

Partitioning the decision table U with respect to R,

$U \setminus R = \{\{1,2\},\{3,4\},\{5\}\}$

Partitioning the decision table U with respect to decision attribute e, $U \setminus \{e\} = \{\{1\},\{2\},\{3\},\{4\},\{5\}\}$

Mapper function maps the data set in different nodes and computes the *{Key, Value}* pair with respect to equation $U \setminus R$ and $U \setminus \{e\}$ in parallel on different nodes. The output is shown in Table II.

TABLE II. {KEY, VALUE} DISTRIBUTION

x∈U	*Key*	*Value1*	*Value2*
1	{1,2}	{1}	1
2	{1,2}	{2}	1
3	{3,4}	{3}	1
4	{3,4}	{4}	1
5	5	{5}	1

Reducer function reduces the Table II with respect to same key value as shown in Table III and in parallel it computes the positive region for all conditional attributes with respect to decision attribute *e*. The result is shown in Fig. 3.

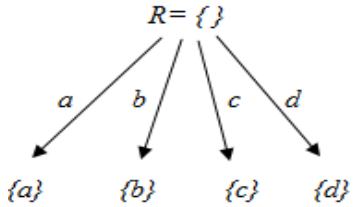

Fig. 2. Initial mapping

TABLE III. REDUCED {KEY, VALUE} PAIR

x∈U	*Key*	*Value1*	*Value2*
1	{1,2}	-1	0
2	{3,4}	-1	0
3	{5}	{5}	1

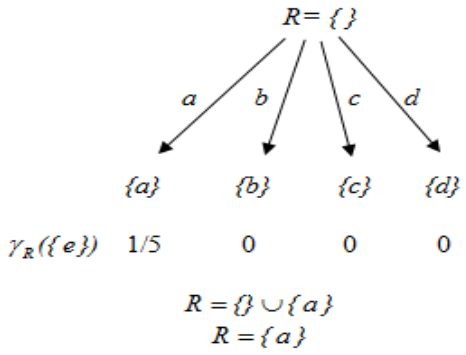

Fig. 3. Degree of dependency of conditional attributes

After calculation of the degree of dependency of all conditional attributes, it is seen that the value of attribute *{a}* is highest. Hence, the first element in reduct set R is {a}. Now, consider all the combination of attribute *{a}* with other attributes *{b,c,d}* and recalculate the degree of dependency. After recalculation, combination {a,c} has the highest degree of dependency value and it is equal to 1, so the final reduct set is R=*{a,c},* shown in Fig. 4.

The proposed method introduces a novel Parallel-Reduct algorithm using MapReduce framework. It includes three phases shown in Fig. 5.The first phase HEURISTICMAPRED is responsible for computing the cardinality of the sets, in the next level SUMMERMAPRED computes the values of the degree of dependency of *X*. Finally it selects the best attribute *x*. This attribute *x* is the element of reduct set.

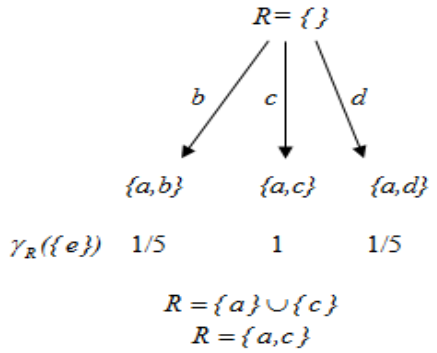

Fig. 4. Reduct set R

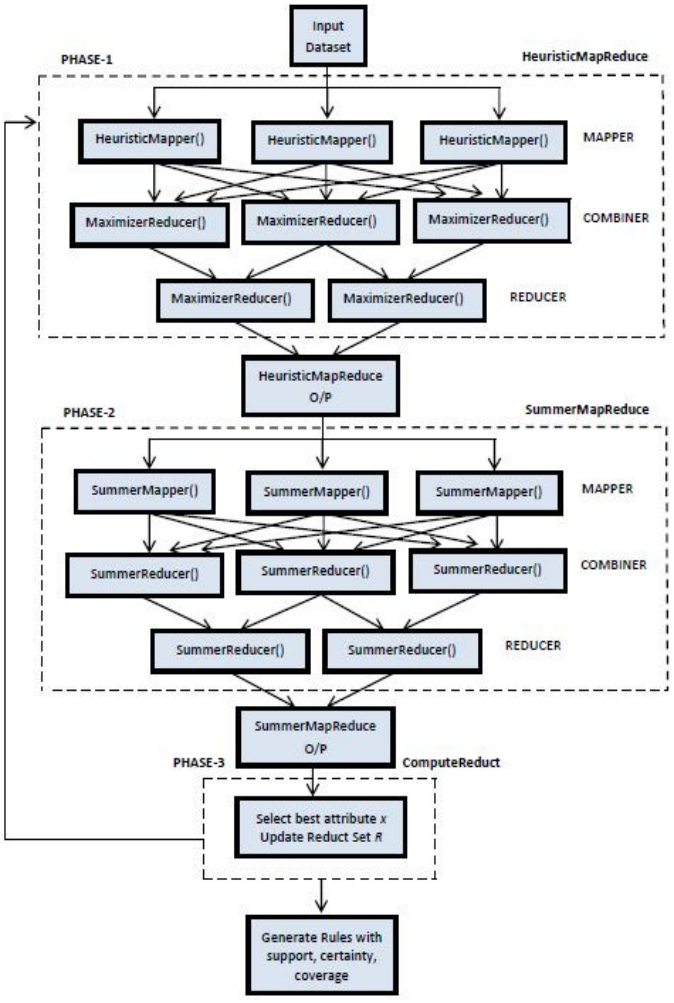

Fig. 5. Dataflow in proposed MapReduce framework

B. *Proposed Algorithm*

Input: An Information System $I = <U, C, D>$, where, U is the information table, C is the set of conditional attribute and D is the decision attribute.
Output: Reduct set R.
$R \leftarrow \{\}$; $i \leftarrow 0$
repeat
  StartMapReduceTask HEURISTICMAPRED
    Mapper ← HeuristicMapper
    Combiner ← MaximizerReducer
    Reducer ← MaximizerReducer
    Input ← DataSet
  EndMapReduceTask

  StartMapReduceTask SUMMERMAPRED
    Mapper ← SummerMapper
    Combiner ← SummerReducer
    Reducer ← SummerReducer
    Input ← HEURISTICMAPRED.output
  EndMapReduceTask

  $x \leftarrow \varepsilon$
  $\gamma_{R\cup\{x\}}(D) \leftarrow -\infty$
  for all $< j, \gamma_{R\cup\{j\}}(D) > \in$ SUMMERMAPRED.output
    do
      if $\gamma_{R\cup\{j\}}(D) > \gamma_{R\cup\{x\}}(D)$ **then**
        $x \leftarrow j$
        $\gamma_{R\cup\{x\}}(D) \leftarrow \gamma_{R\cup\{j\}}(D)$
      endif
  endfor
  $R \leftarrow R \cup \{x\}$
until $\gamma_R(D) = \gamma_C(D)$
return R

PHASE-1: HEURISTICMAPRED

In this phase, MapReduce is launched in order to compute the cardinality of the datasets.

function HeuristicMapper *(Object O)*
  Compute $O[R]$ // Values of attributes contained in the set R for Object O
  for all $x \in C-R$ **do**
    PROJECT($key=x||O[x]||O[R]$, $value = \langle O[D],1\rangle$)
  endfor

The *function HeuristicMapper* computes the cardinality of *{Key, Value}* pair where *Key* and *Value* are the value set and ID of the object respectively. First, it determines the values of attributes contained in the set *R* for Object *O* and then projects the *{Key, Value}* pair with respect to decision attributes *D* for all attribute $x \in C$-*R*.

The *function MaximizerReducer* computes the reduced *{Key K, Value V}* pair. First, it initializes the total number of *Key K* with respect to *Value V* based on decision attribute and then determines the equivalence class with respect to *{Key,Value}* pair. Finally, it combines and reduces the number of *Key K* based on same decision class.

```
function MaximizerReducer (Key K, Values V )
sum ← 0
class ← V[0][0]
class_shared ← TRUE
for all (d, x) ∈ V do
    if d ≠ class then
        class_shared ← FALSE
        Break
    else
        sum ← sum + x
  endif
endfor
if class_shared then
    PROJECT(key = K, value = ‹class, sum›)
else
    PROJECT(key = K, value = ‹ε, 0›)
 endif
```

PHASE-2: SUMMERMAPRED

In this phase, addition MapReduce task has been introduced. The *function SummerMapper* computes the cardinalities of all the elements of the partition $U \setminus P \cup \{x\}$ which belongs to definite decision class. It reduces the number of values passed to *SummerReducer function*.

```
function SummerMapper (Object ‹ x||O[x]||O[R], class,
                                                sum ›)
    if class ≠ ε then
        PROJECT(key = x, value = ‹ sum ›)
    endif
```

The *function SummerReducer* is a simple aggregate function. This function calculates the summation of all input values.

```
function SummerReducer (Key K, Values V )
sum ← 0

for all x ∈ V do
   sum ← sum + x
endfor

PROJECT (key = K, value = sum)
```

This phase uses various concepts of Rough Set theory like positive region, equivalence class, and degree of dependency.

The evaluation of this phase can be done using following steps:

Step1. Scan the output of level-2 SUMMERMAPRED

Step2. Find the equivalence class of conditional and decision attributes.

Step3. Find the subset of each conditional with the decision class.

Step4. If the dependency is 1 then eliminate the conditional Attribute Else goto Step3

Step5. Compare all conditional attributes and get the reduced reduct set.

```
Repeat
x ← ε
γ_{R∪{x}}(D) ← −∞
for all < j, γ_{R∪{j}}(D) > ∈  SUMMERMAPRED.output
do
   if γ_{R∪{j}}(D) > γ_{R∪{x}}(D) then
      x ← j
      γ_{R∪{x}}(D) ← γ_{R∪{j}}(D)
   endif
endfor

R ← R ∪ {x}
until γ_R(D) = γ_C (D)
```

C. Time Complexity of the Proposed Algorithm

The total complexity of the algorithm is:

$$O(|U|*((|D|*\log|D|)+(|C|*\log(|C|+|U|)))*K)$$

where, $K=|R|$ is the size of the reduct set which is usually a smaller number. The complexities of each of the functions are shown below Table IV. Since we assume that $\gamma_C(D)=\gamma_D(D)=1$ (that is the dataset is large enough, with $|C|>>|U|$), we do not need to calculate $\gamma_D(D)$. Therefore the final time complexity is:

$$O(|U|*(|C|*\log(|C|+|U|))*K)$$

Parallel-Reduct algorithm decreases running time by harnessing the power of multiple nodes and multiple cores, and making the processing of large datasets possible, especially those that exceed the capacity of a single node (>>2TB).

TABLE IV. TIME COMPLEXITY

Function	***Time complexity***
HeuristicMapper	$O(\|C\|)$
Sorting keys (HeuristicMapRed)	$O((\|U\|*\|C\|)*\log(\|U\|*\|C\|))$
MaximizerReducer	$O(\|U\|*\|C\|)$
SummerMapper	$O(1)$
Sorting keys (SummerMapRed)	$O((\|U\|*\|C\|)*\log(\|U\|*\|C\|))$
SummerReducer	$O(\|U\|)$
Calculating the best x	$O(\|C\|)$
Total per iteration	$O(\|U\|*(\|C\|*\log(\|C\|+\|U\|)))$

IV. EXPERIMENTAL RESULTS

The performance of the proposed parallel computing approach is evaluated with two gene expression datasets and two machine learning datasets that are taken from GEO Datasets repository [15] and UCI Machine Learning repository [16] respectively. Table V shows the number of instances and number of features present in the datasets.

The experiments are performed on Hadoop cluster in pseudo-distributed mode. The implementation was done in Ubuntu 14.0.4 platform with PCs of Intel(R) Core(TM) i5-4690 CPU @3.90GHZ and 8GB RAM using Java version7.

Table VI shows the outcome of experiments for reduct sets generation. These reduct sets contain a minimum number of features to extract important knowledge from datasets. It shows that our MapReduce-based parallel approach gives better results than the sequential approach in terms of reduct generation and runtime calculations. Table VII illustrates the results of experiments for extracting knowledge from reduct sets in terms of rules generation. It shows that important rules are extracted and categorized whenever the certainty, strength, coverage values are high. The sample rules for Heart Disease dataset are shown in Table VII. In these rules the decision is *presence* for a patient when the features blood_pressure, age and sex values are 130, 62, and 0 respectively. The certainty value of this rule is 100% which signifies that the Heart Disease dataset is consistent with respect to feature values in the reduct set. The strength value of this rule 0.47% indicates that it satisfies 0.47% instances in the dataset. The coverage value of the rule is 1.01%. It signifies that the decision is *presence* for a patient when the probability of the rule is 1.01%. Fig. 6 shows the running time of our proposed algorithm compared to the sequential algorithm. It proves that our MapReduce-based proposed parallel approach performs better than sequential approach.

V. CONCLUSION

In this paper, we proposed a parallel computing approach for extracting knowledge from big data using MapReduce methodology. MapReduce is an effective tool for analysis of large-scale fault-tolerant data. Earlier sequential reduct generation algorithm needs additional computing times whenever the input number and input spaces are high in the system but our MapReduce-based parallel algorithm handles the massive amount of data in distributed manner. Parallel computations in three phases of our proposed approach efficiently process large datasets. Experimental results demonstrate that our MapReduce-based proposed algorithm can scale up the massive datasets in distributed fashion that exploits the advantages of parallel computing approach for extraction of knowledge from big data.

TABLE V. DATASET INFORMATION

Index	*Dataset*	*Number of instances*	*Number of features*
1	Breast Cancer-(GDS3952)[15]	162	54675
2	Lung Cancer-(GDS2771)[15]	192	22283
3	Breast Cancer Wisconsin (Original) Dataset[16]	699	9
4	Heart Disease[16]	270	13

TABLE VI. REDUCT SET

Index	*Number of features in Reduct Set*	*Reduct Set*
1	12	{224980_at, 221999_at, 218587_s_at, 214741_at, 208827_at, 203536_s_at, 200849_s_at, 200781_s_at, 200072_s_at, 1559960_x_at, 1553130_at, 1552287_s_at}
2	8	{202435_s_at , 202461_at , 205830_at , 206628_at , 209774_x_at , 213796_at , 215001_s_at , 215690_x_at}
3	4	{thickness, size.unif, bare, bland}
4	3	{ blood_pressure, age, sex}

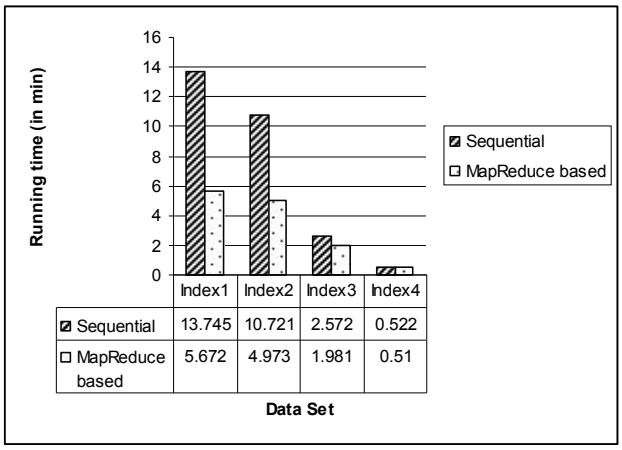

Fig. 6. Running time comparisons

TABLE VII. RULES SET

Index	*Sample Rules*	*Decision*	*Certainty*	*Strength*	*Coverage*
1	if(200849_s_at =(-inf-8.14), 203536_s_at =(-inf-6.46)), 224980_at =(-inf-6.04))	breast cancer(malignant)	100 %	4.32 %	13.73 %
	if(200072_s_at =(-inf-8.935), 218587_s_at =(6.405-7.195))	healthy	100 %	6.79 %.	35.48 %
2	if(202435_s_at =5910.0, 202461_at = 5990.0, 205830_at =16115.0, 209774_x_at =27825.0, 213796_at =39581.0, 215690_x_at = 45266.0)	cancer	100%	2.03 %	3.92 %
	if(209774_x_at = 27822.0, 213796_at = 39581.0, 215690_x_at = 45266.0)	no cancer	100%	3.05 %.	6.32 %
3	if (size.unif = 1.0, bare = 1.0)	Benign	100%	49.27%	75.68%
	if (bare = 10.0, bland = 7.0)	Malignant	100%	5.72%	16.39%
4	If(blood_pressure=120, age=44)	Absence (1)	100%	0.94%	1.77%
	If(blood_pressure=130, age=62, sex=0)	Presence (2)	100%	0.47%	1.01%

REFERENCES

[1] V. López, S. del Río, J. M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data," Fuzzy Sets and Systems, vol. 258, pp. 5-38, 2015.

[2] A. Srinivasan, T. A. Faruquie, and S. Joshi, "Data and task parallelism in ILP using MapReduce," Machine Learning, vol. 86, pp. 141-168, 2011.

[3] Guyon, Isabelle, and A. Elisseeff. "An Introduction to Variable and Feature Selection."Journal of Machine Learning Research, vol. 3, pp. 1157-1182, 2003.

[4] Z. Pawlak, "Rough sets," International Journal of Computer & Information Sciences, vol. 11, pp. 341–356, 1982.

[5] Shang, C., Shen, Q., "Rough Feature Selection for Neural Network Based Image classification," International Journal of Image and Graphics 2, pp. 541-555, 2002.

[6] Francis E.H. tTay, Shen, L., "Economic and Financial Prediction using Rough Sets Model," European Journal of Operational Research 141, pp. 641-659, 2002.

[7] J. Zhang, T. Li, D. Ruan, Z. Gao, and C. Zhao. A parallel method for computing rough set approximations. *Information Sciences*, 194(0): 209–223, July 2012.

[8] Junbo Zhang, Tianrui Li,Yi Pan,"Parallel rough set based knowledge acquisition using MapReduce from big data",Proceeding in, BigMine'12,Beijing, China August, 2012.

[9] Si-Yuan Jing, Jin Yang, Kun She," A Parallel Method for Rough Entropy Computation using MapReduce" 2014 10th International Conference on Computational Intelligence and Security, Kunming, China,15-16 November, 2014.

[10] Weiping Cui , Lei Huang, "Knowledge Reduction method based on Information Entropy for Port Big Data using Mapreduce" 2015 International Conference on Logistics, Informatics and Service Sciences (LISS 2015), Barcelona, Spain,pp. 27-29 July, 2015.

[11] R. Jensen and Q. Shen, "Semantics-preserving dimensionality reduction: rough and fuzzy-rough based approaches," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 12, pp. 1457–1471, 2004.

[12] Z. Pawlak, J. W. Grazymala-Busse, R. Slowinski and W. Ziarko, *Rough sets*, Communications of the ACM, vol. 38, pp. 88-95, 1995.

[13] J. W. Grzymala-Busse, *A new version of the rule induction system LERS*. Fundamenta Informaticae,vol. 31, pp.27-39, 1997.

[14] J. Dean, S. Ghemawat. MapReduce: simplified data processing on large clusters. Proceedings of the OSDI'04, pp. 137-150, 2004.

[15] GEO DataSets: <http://www.ncbi.nlm.nih.gov/gds>

[16] UCI Machine Learning repository: < http://archive.ics.uci.edu/ml/>