Attribute Reduction Based on MapReduce Model and Discernibility Measure

Michal Czolombitko, Jaroslaw Stepaniuk

Attribute reduction based on MapReduce model and discernibility measure

Michal Czolombitko, Jaroslaw Stepaniuk

Faculty of Computer Science
Bialystok University of Technology
Wiejska 45A, 15-351 Bialystok, Poland
{m.czolombitko,j.stepaniuk}@pb.edu.pl
http://www.wi.pb.edu.pl

Abstract. This paper discusses two important problems of data reduction. The problems are related to computing reducts and core in rough sets. The authors use the fact that the necessary information about discernibility matrices can be computed directly from data tables, in the case of this paper so called counting tables are used. The discussed problems are of high computational complexity. Hence the authors propose to use the relevant heuristics, MRCR (**M**ap**R**educe **C**ore and **R**educt Generation) implemented using the MapReduce model.

Keywords: Rough Sets, MapReduce, reducts, attribute reduction, core.

1 Introduction

Since the massive data could be stored in cloud platforms, data mining for the large datasets is hot topic. Parallel methods of computing are alternative for large datasets processing and knowledge discovery for large data. MapReduce is a distributed programming model, proposed by Google for processing large datasets, so called Big Data. Users specify the required functions Map and Reduce and optional function Combine. Every step of computation takes as input pairs $< key, values >$ and produces another output pairs $< key', values' >$. In the first step, the Map function reads the input as a set $< key, values >$ pairs and applies user defined function to each pair. The result is a second set of the intermediate pairs $< key', values' >$, sent to Combine or Reduce function. Combine function is a local Reduce, which can help to reduce final computation. It applies second user defined function to each intermediate key with all its associated values to merge and group data. Results are sorted, shuffled and sent to the Reduce function. Reduce function merges and groups all values to each key and produces zero or more outputs.

Rough set theory is mathematical tool for dealing with incomplete and uncertain information [6]. In the decision systems, not all of the attributes are needed in decision making process. Some of them can be removed without affecting the classification quality, in this sense they are superfluous. One of the advantage of

rough set theory is an ability to compute the reductions of the set of conditional attributes, so called reducts.

In recent years, there has been some research works combining MapReduce and rough set theory. In [12] parallel method for computing rough set approximations was proposed. The authors continued their work and proposed in [13] three strategies based on MapReduce to compute approximations in incomplete information systems. In [11] method for computing core based on finding positive region was proposed. They also presented parallel algorithm of attribute reduction in [10]. However authors used model MapReduce only for splitting data set and parallelization computation using one of traditional reduction algorithm. In [4] is proposed a design of a Patient-customized Healthcare System based on the Hadoop with Text Mining for an efficient Disease Management and Prediction.

In this paper we propose a parallel method MRCR (MapReduce Core and Reduct Generation) for generating core and one reduct or superreduct based on distributed programming model MapReduce and rough set theory. In order to reduce the memory complexity, instead of discernibility matrix were used counting tables to compute discernibility measure of the datasets. The results of the experiments on real datasets show that the proposed method is effective for big data.

This paper is organized as follows. Section 1 includes background introduction to rough sets. Problem of attribute reduction and algorithm of generating core using counting tables are presented in Section 2. Parallel algorithm MRCR based on discernibility measure and MapReduce is proposed in Section 3. Results of experiments and analysis is presented in Section 4. Conclusions and future work are drawn in the last Section.

2 Reducing number of attributes

2.1 Reduct and Core Computation

Let $DT = (U, A \cup \{d\})$ be a decision table, where U is a set of objects, A is a set of conditional attributes and d is a decision attribute. Reduct is subset of conditional attributes $R \subseteq A$, sufficient to get information which we need from the decision table. Superreduct is not necessarily minimal set of those attributes. The core is a set of conditional attributes, which are cannot be removed from original data set without affecting the quality of the classification.

In order to compute the core and reduct we can use discernibility matrix introduced by Prof. Andrzej Skowron (see e.g. [6,8]). Discernibility Matrix is two dimensional table indexed by the objects. Cells of the discernibility matrix contains set with all of conditional attributes on which corresponding objects have different values. Number of not empty cells is discernibility measure of set A. The core is the set of all single element entries of the discernibility matrix. In some cases core can be an empty set.

However, memory complexity of creating discernibility matrix is equal to $(cardinality(U))^2 * cardinality(A)$. This makes impractical using this structure

for large data sets, so called Big Data. Solution could be using counting tables to compute discernibility measure. In next subsection we present method for computing the core, which is base to finding one of the reducts.

2.2 Pseudocode for Generating Core Based on Discernibility Measure of a Set of Attributes

Generating core based on discernibility measure was discussed in [5]. Counting table CT is a two-dimensional array indexed by values of information vectors (vector of all values of an attribute set $B \subseteq A$) and decision values, where

$$CT(i,j) = cardinality(\{x \in U : \vec{x}_B = i \text{ and } d(x) = j\})$$

Pessimistic memory complexity of creating this type of matrix is equal to $cardinality(U) * cardinality(V_d)$, where V_d is a set of all decisions.
The discernibility measure $disc(B)$ of set of attributes $B \subseteq A$ can be calculated from the counting table as follows:

$$disc(B) = \frac{1}{2} \sum_{i,j} \sum_{k,l} CT(i,j) \cdot CT(k,l), \text{if } i \neq k \text{ and } j \neq l$$

The discernibility measure is the number of non empty cells in discernibility matrix.

Below is the pseudocode for this algorithm:

Input: decision table $DT = (U, A \cup \{d\})$
Output: $Core \subseteq A$

$Core \leftarrow \emptyset$ {Core is equal to empty set}
$CT \leftarrow 0$ {All values in counting table CT are equal to 0}
for $x \in U$ **do**
  $CT(\vec{x}_A, d(x)) \leftarrow CT(\vec{x}_A, d(x)) + 1$
end for
$disc(A) \leftarrow 0$
for $[x]_A \in U/IND(A)$ {$U/IND(A)$ is partition of U defined by A} **do**
  for $[y]_A \in U/IND(A)$ **do**
    if $\vec{x}_A \neq \vec{y}_A$ **and** $d(x) \neq d(y)$ **then**
      $disc(A) \leftarrow disc(A) + CT(\vec{x}_A, d(x)) \cdot CT(\vec{y}_A, d(y))$
    end if
  end for
end for
$CT \leftarrow 0$
for $a \in A$ **do**
  $B \leftarrow A - \{a\}$
  $disc(B) \leftarrow 0$
  for $x \in U$ **do**
    $CT(\vec{x}_B, d(x)) \leftarrow CT(\vec{x}_B, d(x)) + 1$
  end for

21: **for** $[x]_B \in U/IND(A)$ **do**
22: **for** $[y]_B \in U/IND(A)$ **do**
23: **if** $\vec{x}_B \neq \vec{y}_B$ **and** $d(x) \neq d(y)$ **then**
24: $disc(B) \leftarrow disc(B) + CT(\vec{x}_B, d(x)) \cdot CT(\vec{y}_B, d(y))$
25: **end if**
26: **end for**
27: **end for**
28: **if** $disc(B) < disc(A)$ **then**
29: $C \leftarrow Core \cup \{a\}$
30: **end if**
31: **end for**

Input to the algorithm is a decision table DT, and output is the core C of DT. In the beginning core C is initialized as empty set and all values in the counting table are set to zero. First loop in line 3 generate counting table for set of all conditional attributes. For each object in decision table, value in array is increased by one. Indexes of value are information vector of object and its value of decision. In the line 6 value of discernibility measure of set of all conditional attributes, $disc(A)$, is initialized as zero. Next two loops in lines 7 and 8 take subsequent equivalence classes to comparison. If information vectors of these classes are not equal, $disc(A)$ is increased by product of two values from the counting tables where indexes are values information vectors and different decisions. Next step is computing the discernibility measure of set of attributes after removing one of them. In line 14 all values in the counting table are set to zero. Loop in line 15 takes attribute a from set of all conditional attributes A. Set B is initialized as set of all conditional attributes after removing this attribute. Similarly as above, in lines 15-27 is calculated $disc(B)$. Finally, values of the discernibility measure of set of all attributes and after removing attribute a are compared in the line 28. In case of difference, this attribute is added to the core.

3 MapReduce Implementation - proposed method

The main concept of the proposed algorithm MRCR is parallel computation of counting tables and discernibility measure for each of conditional attributes and potential reducts. We extended the method proposed by us in [3] on parallel computing discernibility measure and adopt to computing the reduct.

Algorithm MRCR
Input: decision table $DT = (U, A \cup \{d\})$
Output: reduct R
1: $Core \leftarrow ComputeCoreMR(DT)$ {See Step 1a & Step 1b }
2: $B \leftarrow A - Core$
3: $R \leftarrow Core$
4: **while** $disc(A) > disc(R)$ **do**
5: **for** $a \in B$ **do**
6: $PR_a \leftarrow R \cup \{a\}$

7: **end for**
8: $ComputeDiscMeasureMR(PR, DT)$ {See Step 2}
9: $R \leftarrow max\_disc(PR_a)$) {function returns the set with the largest value of discernibility measure}
10: $B \leftarrow B - R$
11: **end while**

The proposed algorithm consists of two main steps: compute core and one reduct. Both stages are composed of two operations Map and Reduce. After computing core, set B is initialized as a set of attributes that not in the core and reduct as a set attributes that are in core.

Table PR (potential reducts) is initialized in the line 6 as a collection containing sets that are unions of the reduct and the every attribute from set B. For each of these sets is parallel calculated the discernibility measure in the line 9. As a new reduct is chosen a set with the maximal discernibility measure and from set B is removed the attribute added to the reduct. Operations in lines 5-10 are repeated until the discernibility measures of the reduct and the set containing all conditional attributes aren't equal.

Step 1a. ComputeCoreMR - Counting Tables
Map
Input: key : subtable id, $value$: decision subtable $DT_i = (U_i, A \cup \{d\})$
Output: $< key', value' >$ pair where $key' : (\vec{x}_B, d(x))$ and $value'$ is id of the object x
1: **for** $x \in U_i$ **do**
2: $key' \leftarrow (\vec{x}_A, d(x))$
3: $value' \leftarrow$ id of the object x
4: $emit(< key', value' >)$
5: **for** $a \in A$ **do**
6: $B \leftarrow A - \{a\}$
7: $key' \leftarrow (\vec{x}_B, d(x))$
8: $value' \leftarrow$ id of the object x
9: $emit(< key', value' >)$
10: **end for**
11: **end for**

Input to function Map are: key is a subtable id stored in HDFS, and $value$ is decision subtable $DT_i = (U_i, A \cup \{d\})$. First loop in line 1 takes an object from decision subtable DT_i and emits pair $< key', value' >$, where key' is information vector with respect to set of all conditional attributes and decision of the object x, and $value$ is id of the this object. Loop in line 5 takes attribute a from set of all conditional attributes A. Set B is initialized as set A after removing this attribute. Next step is emitting pair $< key', value' >$, where key' is information vector with respect to set B and decision of the object x, and $value'$ is id of this object.

Reduce

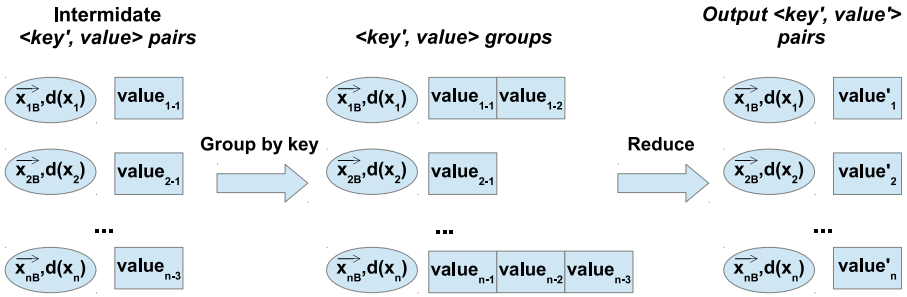
Input: $< key, value >$ where $key : (\vec{x}_B, d(x))$ and $value$ is the number of objects belonging to equivalence class $[x]_B$ with equal decision values from decision subtable DT_i.

Output: the files containing pairs $< key', value' >$ where $key : (\vec{x}_B, d(x))$ and $value'$ is the number of objects belonging to equivalence class $[x]_B$ with equal decision values from decision table DT.

1: $key' \leftarrow key$
2: $value' \leftarrow 0$
3: **for all** $value$ **do**
4: $\quad value' \leftarrow value' + value$
5: **end for**
6: $emit(< key', value' >)$

Input to function Reduce are $< key, value >$ pairs where $key : (\vec{x}_B, d(x))$ and $value$ is the number of objects belonging to equivalence class $[x]_B$ with the same decision from decision subtable DT_i. Function emits pairs $< key', value' >$, where key' is $(\vec{x}_B, d(x))$ and $value'$ is a number of the objects associated with this key from decision table DT. Each of these pair is emitting (See fig. 1).

Fig. 1. The Reduce operation in the Step 1a



Step 1b. ComputeCoreMR - Discernibility measure

Map

Input: the files containing the pairs $< key, value >$ where $key : (\vec{x}_B, d(x))$ and $value$ is number of the objects belong to $[x]_B$ with the same decision value - $count(\vec{x}_B, d(x)))$

Output: $< key', value' >$ pair where key' is the removed attribute from the set of the conditional attributes and $value'$ contains $\vec{x}_B, d(x)$ and the number of objects belonging to equivalence class $[x]_B$ with equal decision values - $count(\vec{x}_B, d(x))$

1: **for all** $(\vec{x}_B, d(x))$ **do**
2: $key' \leftarrow$ removed attribute
3: $value' \leftarrow \vec{x}_B, d(x), count(\vec{x}_B, d(x))$
4: $emit(< key', value' >)$
5: **end for**

Input to the Map operation is directory contains files with pairs $< key, value >$ where $key : (\vec{x}_B, d(x))$ and $value$ is number of the objects belong to $[x]_B$ with the same decision value. Loop in the first line takes every pair $< key, value >$ and emits $< key', value' >$ pair where key' is the removed attribute from the set of the conditional attributes and $value'$ contains $\vec{x}_B, d(x)$ and the number of objects belonging to equivalence class $[x]_B$ with equal decision values - $count(\vec{x}_B, d(x))$.

Reduce

Input: the pairs $< key, value >$ where key is the removed attribute a from the set of the conditional attributes A and $value : (\vec{x}_B, d(x), count(\vec{x}_B, d(x)))$

Output: $< key', value' >$ pair where key' is the removed attribute a from the set of the conditional attributes and $value'$ is the discernibility measure of set A after removing attribute a

1: $disc(A - \{a\}) \leftarrow 0$
2: **for all** $(\vec{x}_B, d(x), count(\vec{x}_B, d(x)))$ **do**
3: **for all** $(\vec{y}_B, d(y), count(\vec{y}_B, d(y)))$ **do**
4: **if** $\vec{x}_B \neq \vec{y}_B$ **and** $d(x) \neq d(y)$ **then**
5: $disc(A - \{a\}) \leftarrow disc(A - \{a\}) + count(\vec{x}_B, d(x))) \cdot count(\vec{y}_B, d(y))$
6: **end if**
7: **end for**
8: **end for**
9: $key' \leftarrow a$
10: $value' \leftarrow disc(A - \{a\})$
11: $emit(< key', value' >)$

Input to the Reduce operation are pairs $< key, value >$ where key is the removed attribute a from the set A of conditional attributes and $value : (\vec{x}_B, d(x), count(\vec{x}_B, d(x)))$. Two loops in lines 2 and 3 take subsequent lines from input to comparison. If these two pairs contains information about two different information vectors and decisions, $disc(A - \{a\})$ is increased by product of $count(\vec{x}_B, d(x))$ and $count(\vec{y}_B, d(y))$.

Computing the core consists of finding in the output files all attributes that cannot be removed from set A without decrease of the discernibility measure.

Step 2. ComputeDiscMeasureMR
Counting Tables - Map

Input: key : subtable id, $value$: decision subtable $DT_i = (U_i, A \cup \{d\})$, table PR - potential reducts

Output: $< key', value' >$ pair where $key' : (\vec{x}_B, d(x))$ and $value'$ is id of the object x

1: **for** $x \in U_i$ **do**
2: **for** $PR_i \in PR$ **do**
3: $key' \leftarrow (x_{\vec{PR_i}}, d(x))$
4: $value' \leftarrow$ id of the object x
5: $emit(< key', value' >)$
6: **end for**
7: **end for**

Input to function Map are: *key* is a subtable id stored in HDFS and *value* is decision subtable $DT_i = (U_i, A \cup \{d\})$ and potential reducts.

Reduce operation is the same like in operation Reduce in the Step 1a. The discernibility measure is computing similarly like in the Step 1b so its description will be skipped.

The output of the second step are files containing subsets of the conditional attributes with their discernibility measure.

4 Experimental Results

The algorithm MRCR was running on the Hadoop MapReduce platform [1], where Hadoop MapReduce is a YARN-based system for parallel processing of big datasets. In the experiments, Hadoop 2.5.1 version was used. All the computation nodes have four 3.4 GHz cores processors and 16 GB of memory. All files were stored in HDFS. Each file was merged on blocks and each of those blocks was replicated tree times. A bound of the proposed method is available free disc space in distributed file system for files and temporary data.

In this paper, we present the results of the conducted experiments using data about children with insulin-dependent diabetes mellitus (type 1) and dataset KDDCup-99 from the machine learning data repository, University of California at Irvine[2].

Diabetes mellitus is a chronic disease of the body's metabolism characterized by an inability to produce enough insulin to process carbohydrates, fat, and protein efficiently. Treatment requires injections of insulin. Twelve conditional attributes, which include the results of physical and laboratory examinations and one decision attribute (microalbuminuria) describe the database used in our experiments. The data collection so far consists of 107 cases. The database is shown at the end of the paper [7]. A detailed analysis of the above data is in chapter 6 of the book [8]. This database was used for generating bigger datasets consisting of $1 \cdot 10^6$ to $20 \cdot 10^6$ of objects. New datasets were created by randomly multiplying the rows of original dataset.

The data set KDDCup-99 consists of almost five million objects. Each object is described by forty one conditional attributes and one decision attribute. Informations gather in this dataset can be used to build a network intrusion detector. This database was used for generating smaller datasets consisting approximately of $0.25 \cdot 10^6$ to $5 \cdot 10^6$ of objects. New datasets were created by randomly splitting the rows of original dataset.

Numerical values in both datasets were discretized.

4.1 Speedup

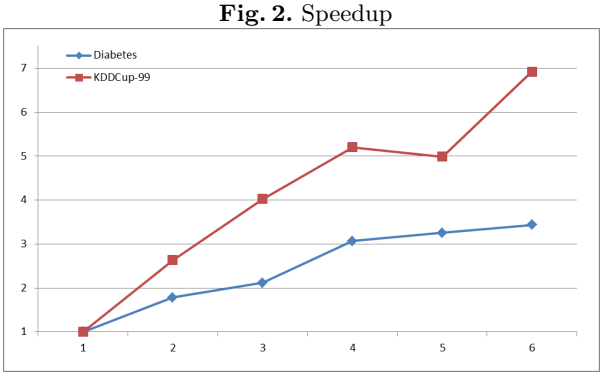
In speedup tests, the dataset size is constant and the number of nodes grows in each step of experiment. To measure speedup, we used dataset diabetes contains $20{\cdot}10^6$ objects and whole KDDCup-99 dataset. The speedup given by the n-times larger system is measured as [9]:

$$Speedup(n) = \frac{t_n}{t_1}$$

where n is number of the nodes in cluster, t_1 is the computation time on one node, and t_n is the computation time on n nodes.

The ideal parallel system with n nodes provides n times speedup. The linear speedup is difficult to achieve because of the I/O operations data from HDFS and the communication cost between nodes of Hadoop cluster. Fig. 3 shows that

Fig. 2. Speedup



with the growth of the number of nodes, the speed performs better.

4.2 Scaleup

Scaleup analysis measures stability system when system and dataset size grow in each step of experiment. The scaleup coefficient is defined as follows [9]:

$$Scaleup(DT, n) = \frac{t_{DT_{1,1}}}{t_{DT_n,n}}$$

where $t_{DT_{1,1}}$ is the computational time for dataset DT on one node, and $t_{DT_{n,n}}$ is the computational time for n times larger dataset DT on n nodes.

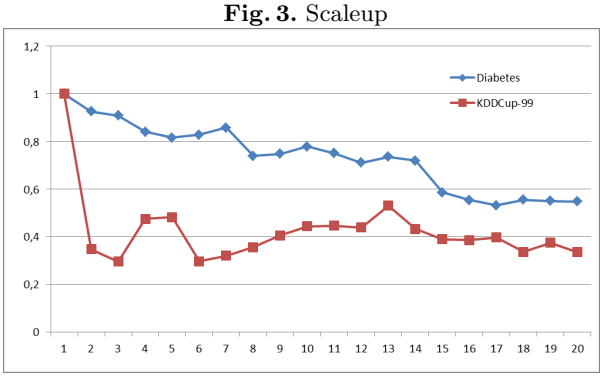
Table shows number of attributes in computed the core and the reduct for KDDCup-99 dataset.

Fig. 3 shows the scalability of algorithm for KDDCup-99 dataset strictly depends on the size of the core and the reduct. For the diabetes dataset scalability of the proposed parallel algorithm stabilize when the number of the nodes is equal 15.

Table 1. Size of the core and the reduct in scaleup experiment for KDDCup-99 dataset

Number of the nodes	1	2–3	4–5	6–17	18	19–20
Size of the core	4	18	26	30	31	32
Size of the reduct	8	20	26	30	31	32

Fig. 3. Scaleup



4.3 Sizeup

In sizeup tests, the number of nodes is constant, and the dataset size grows in each step of experiment. Sizeup measures how much time is needed for calculations when the size of dataset is n times larger than the original dataset. Sizeup is defined as follows [9]:

$$Sizeup(DT) = \frac{t_{DT_n}}{t_{DT_1}}$$

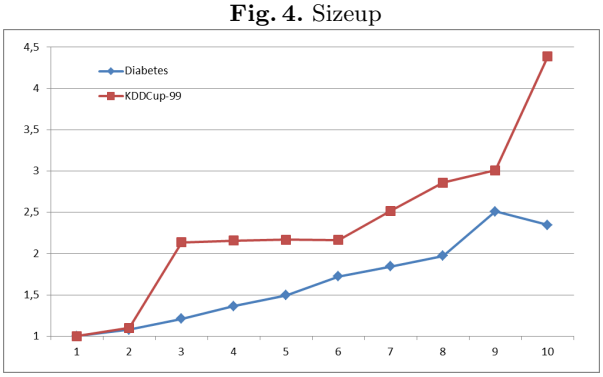
where t_{DT_1} is execution time for a given dataset DT, and t_{DT_n} is execution time n times larger dataset than DT.

Fig. 4 shows the sizeup experiments results on twenty nodes. Results shows that the proposed algorithm copes well with the growing amount of data.

Conclusions and Future Research

In this paper, attribute reduction for large datasets based on rough set theory is studied. A parallel attribute reduction algorithm MRCR is proposed, which is based on the distributed programming model of MapReduce and the reduct computation algorithm using discernibility measure of a set of attributes. It is worth noting that a very interesting element of the paper is an usage of a counting table instead of a discernibility matrix. The results of the experiments show that the proposed method is efficient for large data, and it is a useful

Fig. 4. Sizeup



method for data mining and knowledge discovery for big datasets. Our future research work will focus on applications of distributed in-memory computing for attribute discretization.

Acknowledgements

This research was partially supported by the grant S/WI/3/2013 of the Polish Ministry of Science and Higher Education. The experiments were performed using resources co-financed with the European Union funds as a part of the "Centre for Modern Education of the Bialystok University of Technology" project (Operational Programme Development of Eastern Poland).

References

1. Hadoop MapReduce, `http://hadoop.apache.org`
2. UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science, Irvine, CA, `http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data`
3. M. Czołombitko, J. Stepaniuk, Generating core based on discernibility measure and MapReduce, Pattern recognition and machine intelligence: 6th International conference, PReMI 2015, Lecture Notes in Computer Science, vol. 9124, 2015, 367–376.
4. B.K. Lee, E.H. Jeong, A Design of a Patient-customized Healthcare System based on the Hadoop with Text Mining (PHSHT) for an efficient Disease Management and Prediction, International Journal of Software Engineering and Its Applications, vol. 8, no. 8, 2014, 131–150.
5. H. S. Nguyen, Approximate Boolean Reasoning: Foundations and Applications in Data Mining, Transactions on Rough Sets V, Lecture Notes in Computer Science, vol. 4100, 2006, 334–506.
6. Z. Pawlak, A. Skowron, Rudiments of rough sets. Information Sciences, 177(1) 2007, 3–27.

7. J. Stepaniuk, Knowledge discovery by application of rough set models, Rough Set Methods and Applications. New Developments in Knowledge Discovery in Information Systems, Physica–Verlag, Heidelberg, 2000, 137–233.
8. J. Stepaniuk, Rough–Granular Computing in Knowledge Discovery and Data Mining, Springer, 2008.
9. X. Xu, J. Jäger, H.P. Kriegel, A fast parallel clustering algorithm for large spatial databases, Data Mining and Knowledge Discovery 3, 1999, 263–290.
10. Y. Yang, Z. Chen, Z. Liang, G. Wang, Attribute Reduction for Massive Data Based on Rough Set Theory and MapReduce, Lecture Notes in Computer Science, vol. 6401, 2010, 672–678.
11. Y. Yang, Z. Chen, Parallelized Computing of Attribute Core Based on Rough Set Theory and MapReduce, Lecture Notes in Computer Science, vol. 7414, 2012, 155–160.
12. J. Zhang, T. Li, D. Ruan, Z. Gao, Ch. Zhao, A parallel method for computing rough set approximations, Information Sciences, vol. 194, 2012, 209–223.
13. J. Zhang., J. Wong, Y. Pan, T. Li: A parallel matrix-based method for computing approximations in incomplete information systems, systems. IEEE Transactions on Knowledge Data Engineering, vol. 27, 2015, 326–339.