Information Sciences 194 (2012) 209–223

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A parallel method for computing rough set approximations

Junbo Zhang [a], Tianrui Li [a,*], Da Ruan [b,c], Zizhe Gao [a], Chengbing Zhao [a]

[a] School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China
[b] Belgian Nuclear Research Centre (SCK•CEN), Boeretang 200, 2400 Mol, Belgium
[c] Department of Applied Mathematics and Computer Science, Ghent University, 9000 Gent, Belgium

ARTICLE INFO

Article history:
Received 11 February 2011
Received in revised form 2 December 2011
Accepted 31 December 2011
Available online 11 January 2012

Keywords:
Rough sets
Data mining
Approximations
Hadoop
MapReduce

ABSTRACT

Massive data mining and knowledge discovery present a tremendous challenge with the data volume growing at an unprecedented rate. Rough set theory has been successfully applied in data mining. The lower and upper approximations are basic concepts in rough set theory. The effective computation of approximations is vital for improving the performance of data mining or other related tasks. The recently introduced MapReduce technique has gained a lot of attention from the scientific community for its applicability in massive data analysis. This paper proposes a parallel method for computing rough set approximations. Consequently, algorithms corresponding to the parallel method based on the MapReduce technique are put forward to deal with the massive data. An extensive experimental evaluation on different large data sets shows that the proposed parallel method is effective for data mining.

1. Introduction

With the development of information technology, data volumes processed by many applications will routinely cross the peta-scale threshold, which will in turn increase the computational requirements. Data processing and knowledge discovery for massive data is always a hot topic in data mining [10]. There are many applications associated with massive data, such as association rule mining [11], sequential pattern mining [16], text mining [18] and temporal data mining [20].

Rough set theory, introduced by Pawlak in 1982, is a powerful mathematical tool for dealing with inconsistent information in decision situations [25,27,28]. It plays an important role in the fields of machine learning, pattern recognition and data mining during last decades [2–5,12–15,19,21,22,29–31]. Among many algorithms based rough sets, Hu et al. introduced a simple and efficient hybrid attribute reduction algorithm based on a generalized fuzzy-rough model [13], and constructed neighborhood-based attribute reduction techniques and classifiers [15]. Qian et al. proposed a theoretic framework based on rough set theory, called the positive approximation, to accelerate algorithms of heuristic attribute reduction [30]. Li et al. presented a method for updating rough set approximations of a concept in an incomplete information system through characteristic relations when an attribute set varies over time [19]. Chen et al. proposed another incremental algorithm for updating the rough set approximations of a concept under coarsening or refining of attributes' values [3]. Liu et al. further presented an optimization incremental approach as well as its algorithm for inducing interesting knowledge in business intelligent information systems [22].

Along with the coming cloud computing era, there are urgent requirements, *e.g.*, business intelligence, for data mining on the massive data by cloud computing platforms. For the purpose of data mining for massive data, parallel computing modes

[*] Corresponding author.
E-mail addresses: JunboZhang86@163.com, jbzhang86@gmail.com (J. Zhang), trli@swjtu.edu.cn (T. Li), druan@sckcen.be, da.ruan@ugent.be (D. Ruan), gaozizhe@126.com (Z. Gao), feiniao198751@foxmail.com (C. Zhao).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.12.036

and algorithms are typical methods in these research fields. Dean and Ghemawat from Google firstly presented a parallel programming model, MapReduce, which was a framework for processing huge data sets on certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster [6,7]. MapReduce is a popular computing model for cloud computing platforms. There are many research papers related to MapReduce combined with the traditional methods [1,8,23,33,36–39]. Followed by Google's work, Ekanayake et al. presented CGL-MapReduce, a streaming-based MapReduce implementation [8]. In [33], Wan et al. described how document clustering for large collection can be efficiently implemented with MapReduce and presented the design and implementation of tfidf and K-Means algorithm based on MapReduce. Zinn et al. presented approaches for exploiting data parallelism in XML processing pipelines through novel compilation strategies to the MapReduce framework [39]. Lv et al. proposed a parallel K-Means clustering of remote sensing images based on MapReduce in [23]. In [1], Berlinska et al. analyzed MapReduce computations as a divisible load scheduling problem and proposed a mathematical model of the application. Yuan et al. presented a new distributed data integration system, called VDB-MR, which was based on the MapReduce technology, to efficiently integrate data from heterogeneous data sources [37]. Zhao et al. proposed a parallel K-Means clustering algorithm based on MapReduce in [38]. Yang et al. presented an attribute reduction for massive data based on MapReduce in the field of rough sets [36].

To our knowledge, most of the traditional algorithms based on rough sets are the serial algorithms and existing rough set tools only run on a single computer to deal with small data sets, *e.g.*, ROSETTA, a rough set toolkit for analysis of data, can be downloaded from http://www.lcb.uu.se/tools/rosetta/. It greatly restricts the applications of rough sets. Generally, the computation of approximations is a necessary step in knowledge representation and reduction based on rough sets. To expand the application of rough sets in the field of data mining and deal with huge data sets, we discuss the parallel computation of the rough set approximations in this paper. To design algorithms (*i.e.*, attribute reduction, rule induction algorithms and so on) for dealing with massive data effectively, we propose a parallel method for computing rough set equivalence classes and decision classes. The association is defined to describe the relation between equivalence classes and decision classes. We design the corresponding parallel algorithms for computing equivalence classes, decision classes and associations between equivalence classes and decision classes. We present a parallel method for computing rough set approximations. Our parallel algorithms have all been implemented on Hadoop MapReduce platform [40]. The experimental results on the large data set KDDCup-99 and three synthetic data sets show that the proposed parallel method scales up very well and has excellent speedup and sizeup behavior. Specially, as the size of the data set increases, the speedup performs better.

The remaining of this paper is organized as follows. Section 2 includes the elementary background introduction to MapReduce and rough sets. The methods and algorithms for computing rough set approximations based on MapReduce are presented in Section 3. Experimental analysis is given in Section 4. The paper ends with conclusions and future work in Section 5.

2. Preliminaries

In this section, we briefly review the MapReduce technique [6,7,17,32], some basic concepts, notations and results of rough sets [25–28].

2.1. The MapReduce model

MapReduce is a programming model and an associated implementation proposed by Google for processing and generating large data sets in a distributed computing environment that is amenable to a broad variety of real-world tasks. Dean and Ghemawat [7] described the MapReduce programming model as follows:

The computation takes a set of input *key*/*value* pairs, and produces a set of output *key*/*value* pairs. The user of the MapReduce library expresses the computation as two functions: Map and Reduce.

- **Map-function** takes an input pair and produces a set of intermediate *key*/*value* pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key *I* and transforms them to the Reduce function.
- **Reduce-function** accepts an intermediate key *I* and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically, just zero or one output value is produced per Reduce invocation.

Fig. 1 shows the data flow and different phases of the MapReduce framework, where HDFS means Hadoop distributed file system [34].

2.2. Rough sets

Given a pair $K = (U,R)$, where U is a finite and non-empty set called the universe, and $R \subseteq U \times U$ is an equivalence on U. The pair $K = (U,R)$ is called an approximation space. The equivalence relation R partitions the set U into several disjoint subsets. This partition of the universe forms a quotient set induced by R, denoted by U/R. If two elements $x, y \in U (x \neq y)$, are indistinguishable under R, we say x and y belong to the same equivalence class. The equivalence class including x is denoted by $[x]_R$.

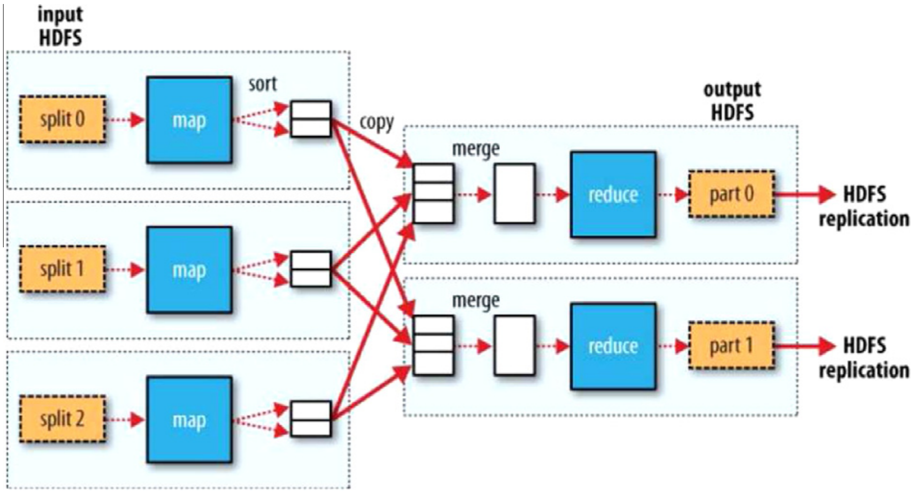

Fig. 1. The MapReduce programming model [34].

An approximation space $K = (U,R)$ is characterized by an information system $S = (U,A,V,f)$, where $U = \{x_1,x_2,\ldots,x_n\}$ is a non-empty finite set of objects, called a universe. $A = \{a_1,a_2,\ldots,a_m\}$ is a non-empty finite set of attributes (features). Specifically, $S = (U,A,V,f)$ is called a decision table if $A = C \cup D$, where C is a set of condition attributes and D is a set of output or decision results, $C \cap D = \emptyset$. $V = \bigcup_{a\in A} V_a$, V_a is a domain of the attribute a. f: $U \times A \rightarrow V$ is an information function such that $f(x,a) \in V_a$ for every $x \in U$, $a \in A$. $f(x_i,a_j)$ denotes the value of object x_i on attribute a_j.

Definition 1. Let $B = \{b_1,b_2,\ldots,b_l\} \subseteq C$ be a subset of condition attributes. The information set with respect to B for any object $x \in U$ can be denoted by

$$\vec{x}_B = \langle f(x,b_1), f(x,b_2), \ldots, f(x,b_l)\rangle \tag{1}$$

An equivalence relation with respect to B called the indiscernibility relation, denoted by $IND(B)$, is defined as

$$IND(B) = \left\{(x,y)|(x,y) \in U \times U, \vec{x}_B = \vec{y}_B\right\} \tag{2}$$

Two objects x, y satisfying the relation $IND(B)$ are indiscernible by attributes from B.

The equivalence relation $IND(B)$ partitions U into some equivalence classes given by

$$U/IND(B) = \{[x]_B | x \in U\} \tag{3}$$

where $[x]_B$ denotes the equivalence class determined by x with respect to B, $[x]_B = \{y \in U|(x,y) \in IND(B)\}$. For simplicity, $U/IND(B)$ will be replaced by U/B.

Definition 2. Let $B \subseteq C$ be a subset of condition attributes. The information set with respect to B for any $E \in U/B$ is denoted by

$$\vec{E}_B = \vec{x}_B, \quad x \in E \tag{4}$$

Example 1. A decision table $S = (U,A,V,f)$ is given in Table 1, where a_1, a_2 are the condition attributes and D is the decision.

Table 1
A decision table S.

Object	a_1	a_2	D	Object	a_1	a_2	D
x_1	0	0	0	x_7	0	0	0
x_2	0	0	1	x_8	1	1	1
x_3	1	0	1	x_9	1	1	2
x_4	1	1	1	x_{10}	0	1	1
x_5	0	0	0	x_{11}	1	0	2
x_6	1	1	2	x_{12}	1	1	1

Let $B = \{a_1, a_2\}$. We compute the equivalence classes. According to Definition 1, we have $U/B = \{E_1, E_2, E_3, E_4\}$, where

$$\begin{cases} E_1 = [x_1]_B = [x_2]_B = [x_5]_B = [x_7]_B = \{x_1, x_2, x_5, x_7\} \\ E_2 = [x_3]_B = [x_{11}]_B = \{x_3, x_{11}\} \\ E_3 = [x_4]_B = [x_6]_B = [x_8]_B = [x_9]_B = [x_{12}]_B = \{x_4, x_6, x_8, x_9, x_{12}\} \\ E_4 = [x_{10}]_B = \{x_{10}\} \end{cases}$$

Obviously, the information set for E_1 and information set for the object of E_1 with respect to B are shown as follows:

$$\begin{cases} \vec{E}_{1B} = \langle 0, 0\rangle \\ \vec{x}_{1B} = \vec{x}_{2B} = \vec{x}_{5B} = \vec{x}_{7B} = \langle 0, 0\rangle \end{cases}$$

Theorem 1. *Let $B \subseteq C$ be a subset of condition attributes, E, F be two equivalence classes with respect to B. One of the following results holds:*

(1) *If $\vec{E}_B = \vec{F}_B$, then these two equivalence classes E and F can be combined as one equivalence class, denoted as G, with respect to B, where $G = E \cup F$ and $\vec{G}_B = \vec{E}_B = \vec{F}_B$;*
(2) *If $\vec{E}_B \neq \vec{F}_B$, then these two equivalence classes E and F cannot be combined as one equivalence class with respect to B.*

Proof.

(1) $\vec{E}_B = \vec{F}_B$ means E, F have the same information set with respect to B. By Definition 1, E, F are indistinguishable with respect to B. Therefore, these two equivalence classes E and F can be combined as one equivalence class.
(2) $\vec{E}_B \neq \vec{F}_B$ means E, F have different information sets with respect to B. By Definition 1, E, F are distinguishable with respect to B. Therefore, these two equivalence classes E and F cannot be combined as one equivalence class. □

Definition 3. Let $X \subseteq U$, and R be an equivalence relation. $U/R = \{E_1, E_2, \ldots, E_t\}$. The lower and upper approximations of X are defined as

$$\begin{cases} \underline{R}(X) = \{x \in U | [x]_R \subseteq X\} \\ \overline{R}(X) = \{x \in U | [x]_R \cap X \neq \emptyset\} \end{cases} \tag{5}$$

or, equivalently,

$$\begin{cases} \underline{R}(X) = \cup\{E \in U/R | E \subseteq X\} \\ \overline{R}(X) = \cup\{E \in U/R | E \cap X \neq \emptyset\} \end{cases} \tag{6}$$

Obviously, $\underline{R}(X) \subseteq X \subseteq \overline{R}(X)$. The boundary region of X in the approximation space is defined as

$$BND_B(X) = \overline{R}(X) - \underline{R}(X) \tag{7}$$

Example 2 (*Example 1 continued*). Assume $X = \{x_2, x_3, x_4, x_8, x_{10}, x_{12}\}$. We compute the approximations and boundary region of X with respect to B.

$$\begin{cases} \underline{R_B}(X) = \cup\{E \in U/B | E \subseteq X\} = E_4 = \{x_{10}\} \\ \overline{R}_B(X) = \cup\{E \in U/B | E \cap X \neq \emptyset\} = E_1 \cup E_2 \cup E_3 \cup E_4 = \{x_1, x_2, \ldots, x_{12}\} \\ BND_B(X) = \overline{R}_B(X) - \underline{R_B}(X) = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{11}, x_{12}\} \end{cases}$$

Definition 4. Given a decision table $S = (U, C \cup D, V, f)$. $U/D = \{D_1, D_2, \ldots, D_r\}$ called the set of decision classes, means that the object set U is partitioned into r mutually exclusive crisp subsets by the decision attributes D. Given any subset $B \subseteq C$ and $IND(B)$ is the equivalence relation induced by B, then one can define the lower and upper approximations of the decision D as

$$\begin{cases} \underline{R_B}(D) = \{\underline{R_B}(D_1), \underline{R_B}(D_2), \ldots, \underline{R_B}(D_r)\} \\ \overline{R}_B(D) = \{\overline{R}_B(D_1), \overline{R}_B(D_2), \ldots, \overline{R}_B(D_r)\} \end{cases} \tag{8}$$

We denote $POS_B(D) = \bigcup_{D_i \in U/D} \underline{R_B}(D_i)$ which is called as a positive region of D with respect to the condition attribute set B. It is the set of all elements of U that can be uniquely classified to blocks of the partition U/D by means of B.

Example 3 (*Examples 1 and 2 continued*). D is the decision. Here, we compute the lower and upper approximations of the decision D and the positive region of D. By Definition 1, $U/D = \{D_1,D_2,D_3\}$, where $D_1 = \{x_1,x_5,x_7\}$, $D_2 = \{x_2,x_3,x_4,x_8,x_{10},x_{12}\}$, $D_1 = \{x_6,x_9,x_{11}\}$.

$$\begin{cases} \underline{R}_B(D_1) = \emptyset \\ \overline{R}_B(D_1) = E_1 = \{x_1, x_2, x_5, x_7\} \\ \underline{R}_B(D_2) = E_4 = \{x_{10}\} \\ \overline{R}_B(D_2) = E_1 \cup E_2 \cup E_3 \cup E_4 = \{x_1, x_2, \ldots, x_{12}\} \\ \underline{R}_B(D_3) = \emptyset \\ \overline{R}_B(D_3) = E_2 \cup E_3 = \{x_3, x_4, x_5, x_8, x_9, x_{11}, x_{12}\} \end{cases}$$

$$POS_B(D) = \bigcup_{D_i \in U/D} \underline{R}_B(D_i) = \underline{R}_B(D_1) \cup \underline{R}_B(D_2) \cup \underline{R}_B(D_3) = \{x_{10}\}$$

3. Computation of rough set approximations based on MapReduce

We now present the idea of our parallel computation of rough set approximations based on MapReduce. We briefly summarize the algorithm for computation of rough set approximations and analyze the parallel parts and serial parts in the algorithm.

According to Definitions 1–4, we present a serial method for computing rough set approximations and a Naive Algorithm for Computation of Rough Set Approximations (NACRSA) corresponding to the serial method (Fig. 2 and Algorithm 1).

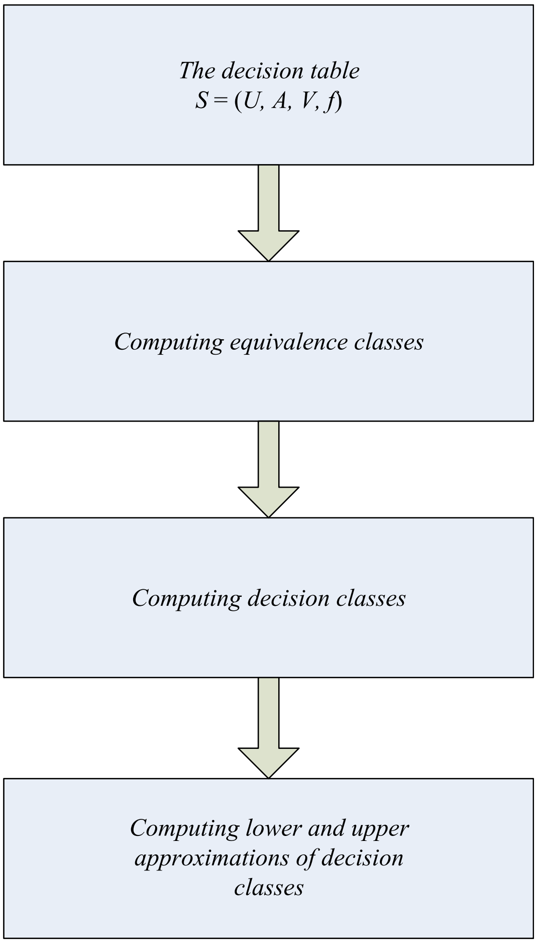
The decision table
S = (U, A, V, f)
Computing equivalence classes
Computing decision classes
Computing lower and upper approximations of decision classes

Fig. 2. The serial method for computing rough set approximations.

Algorithm 1. A Naive Algorithm for Computation of Rough Set Approximations (NACRSA)

```
Input:
$S = (U, C \cup D, V, f)$.
Output:
The approximations of decision classes: $\underline{R}_C(D_i)$ and $\overline{R}_C(D_i), i = 1, 2, \ldots, r$.
begin
    compute $U/C = \{E_1, E_2, \ldots, E_t\}$;
    compute $U/D = \{D_1, D_2, \ldots, D_n\}$;
    for each $D_i \in U/D$ do
        compute $\underline{R}(D_i) = \cup\{E \in U/C | E \subseteq D_r\}$;
        compute $\overline{R}(D_i) = \cup\{E \in U/C | E \cap D_i \neq \emptyset\}$;
    end
    return $\underline{R}_C(D_i)$ and $\overline{R}_C(D_i), i = 1, 2, \ldots, r$.
end
```

From Fig. 2 and Algorithm 1, the most intensive computation to occur is the computation of rough set equivalence classes, decision classes and approximations. We discuss how to execute these three parts in parallel as follows:

Definition 5. Given a decision table $S = (U, C \cup D, V, f)$. Let $S = \bigcup_{i=1}^m S_i$, where $S_i = (U_i, C \cup D, V, f)$. It satisfies (1) $U = \bigcup_{i=1}^m U_i$; (2) $U_j \cap U_k = \emptyset$, $\forall j, k \in \{1, 2, \ldots, m\}$ and $j \neq k$. It means the decision table S is divided into m sub-decision tables. Then we call S_i is a sub-decision table of S.

3.1. The computation of rough equivalence classes based on MapReduce

Theorem 2. *Given a decision table $S = (U, C \cup D, V, f)$. Let $S = \bigcup_{i=1}^m S_i$, where $S_i = (U_i, C \cup D, V, f)$. For any subset $B \subseteq C$, $U/B = \{E_1, E_2, \ldots, E_t\}$ and $\forall i \in \{1, 2, \ldots, m\}$, $U_i/B = \{E_{i1}, E_{i2}, \ldots, E_{ip_i}\}$. Let $E_{all} = \bigcup_{i=1}^m U_i/B = \{E_{11}, E_{12}, \ldots, E_{1p_1}, \ldots, E_{m1}, E_{m2}, \ldots, E_{mp_m}\}$. Therefore, for any $E_j \in U/B$, $j \in \{1, 2, \ldots, t\}$, we have $E_j = \bigcup\{F \in E_{all} | \vec{F}_B = \vec{E}_{jB}\}$.*

Proof. By Theorem 1, two equivalence classes can be combined as one equivalence class if their information set with respect to B is the same. It means multiple equivalence classes can be combined as one equivalence class if their information set with respect to B is the same. Therefore, for any $E_j \in U/B$, $j \in \{1, 2, \ldots, t\}$, we have $E_j = \bigcup\{F \in E_{all} | \vec{F}_B = \vec{E}_{jB}\}$.

According to Theorem 2, each sub-decision table can compute equivalence classes independently. At the same time, the equivalence classes of different sub-decision tables can combine together if their information set is the same. Therefore, rough set equivalence classes of the decision table S can be executed in parallel. Here, we design a Parallel Algorithm for Computation of Rough Set Equivalence Classes (PACRSEC) based on MapReduce. The **Algorithm PACRSEC** is divided into **Algorithm PACRSEC-Map** and **Algorithm PACRSEC-Reduce**, as outlined in Algorithms 2 and 3, respectively. □

Algorithm 2. PACRSEC-Map (*key*, *value*)

```
Input:
//key: document name
//value: $S_i = \{U_i, C \cup D, V, f\}$
Output:
⟨key′, value′⟩ pair, where key′ is the information set of the object with respect to the condition attribute set C and value′ is the ID of the object.
begin
    for each $x \in U_i$ do
        let $key' = \vec{x}_C$;
        let $value'$ = the ID of $x$;
        output.collect (key′, value′);
    end
end
```

Algorithm 3. PACRSEC-Reduce (key,V)

```
Input:
//key: the information set of the object with respect to the condition attribute set $C$
//value: the list of the ID from a different host
Output:
$\langle key', value' \rangle$ pair, where $key'$ is the information set with respect to the condition attribute set $C$ and $value'$ is the set of the ID.
begin
    let $value' \leftarrow \emptyset$;
    for each $ID \in V$ do
        $value' \leftarrow value' \cup ID$;
    end
    output.collect ($key', value'$);
end
```

Example 4. Two sub-decision tables of S, $S_1 = (U_1, C \cup D, V, f)$ and $S_2 = (U_2, C \cup D, V, f)$, $S = S_1 \cup S_2$, are given in Tables 2 and 3, where $C = \{a_1, a_2\}$ is the condition attribute set, D is the decision, $U_1 = \{x_1, x_2, \ldots, x_6\}$ and $U_2 = \{x_7, x_8, \ldots, x_{12}\}$. Here, according to **Algorithm PACRSEC-Map** and **Algorithm PACRSEC-Reduce**, we compute rough set equivalence classes in parallel. From Fig. 3, there are two steps in this process: Map and Reduce. In Map step, we partition U_1 and U_2 into the some equivalence classes, respectively. In Reduce step, the equivalence classes are combined if their information set is the same. We present the detailed computational process as follows:

1. PACRSEC-Map step
   - Map1: $U_1/C = \{E_{11}, E_{12}, E_{13}\}$, where $E_{11} = \{x_1, x_2, x_5\}$, $E_{12} = \{x_3\}$, $E_{13} = \{x_4, x_6\}$.
   - Map2: $U_2/C = \{E_{21}, E_{22}, E_{23}, E_{24}\}$, where $E_{21} = \{x_7\}$, $E_{22} = \{x_8, x_9, x_{12}\}$, $E_{23} = \{x_{10}\}$, $E_{24} = \{x_{11}\}$.
2. PACRSEC-Reduce step
   - Reduce1: $E_1 = E_{11} \cup E_{21} = \{x_1, x_2, x_5, x_7\}$.
   - Reduce2: $E_2 = E_{12} \cup E_{24} = \{x_3, x_{11}\}$.
   - Reduce3: $E_3 = E_{13} \cup E_{22} = \{x_4, x_6, x_8, x_9, x_{12}\}$.
   - Reduce4: $E_4 = \emptyset \cup E_{23} = \{x_{10}\}$,

   where $\overrightarrow{E_1} = \langle 0,0 \rangle, \overrightarrow{E_2} = \langle 1,0 \rangle, \overrightarrow{E_3} = \langle 1,1 \rangle, \overrightarrow{E_4} = \langle 0,1 \rangle$.

3.2. The computation of rough decision classes based on MapReduce

Theorem 3. *Given a decision table $S = (U, C \cup D, V, f)$. Let $S = \bigcup_{i=1}^{m} S_i$, where $S_i = (U_i, C \cup D, V, f)$. For the decision D, $U/D = \{D_1, D_2, \ldots, D_r\}$ and $\forall i \in \{1, 2, \ldots, m\}$, $U_i/D = \{D_{i1}, D_{i2}, \ldots, D_{iq_i}\}$. Let $D_{all} = \bigcup_{i=1}^{m} U_i/D = \{D_{11}, D_{12}, \ldots, D_{1q_1}, \ldots, D_{m1}, D_{m2}, \ldots, D_{mq_m}\}$. Therefore, for any $D_k \in U/D$, $k \in \{1, 2, \ldots, r\}$, we have $D_k = \bigcup \{F \in D_{all} | F_D = D_{kD}\}$.*

Table 2
A decision table S_1.

Object	a_1	a_2	D
x_1	0	0	0
x_2	0	0	1
x_3	1	0	1
x_4	1	1	1
x_5	0	0	0
x_6	1	1	2

Table 3
A decision table S_2.

Object	a_1	a_2	D
x_7	0	0	0
x_8	1	1	1
x_9	1	1	2
x_{10}	0	1	1
x_{11}	1	0	2
x_{12}	1	1	1

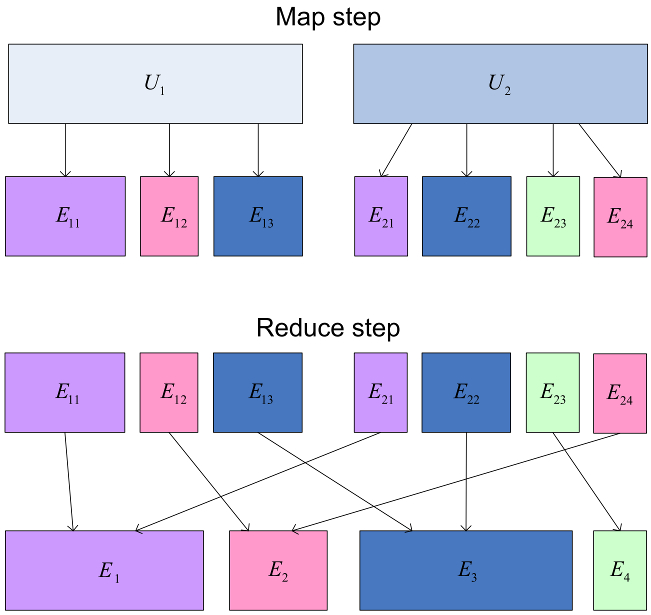

Fig. 3. The example of computing rough set equivalence classes based on MapReduce.

Proof. The proof is similar to that of Theorem 2. □

According to Theorem 3, similar to Theorem 2, rough set decision classes of the decision table S can also be executed in parallel. In the same way, we design a Parallel Algorithm for Computation of Rough Set Decision Classes (PACRSDC) based on MapReduce. The **Algorithm PACRSDC** is divided into **Algorithm PACRSDC-Map** and **Algorithm PACRSDC-Reduce**, as shown in Algorithms 4 and 5, respectively.

Algorithm 4. PACRSDC-Map (*key*, *value*)

```
Input:
//key: document name
//value: $S_i = \{U_i, C \cup D, V, f\}$
Output:
$\langle key', value' \rangle$ pair, where key' is the decision value of the object and value' is the ID of the object.
begin
    for each $x \in U_i$ do
        let $key' = f(x, D)$;
        let $value'$ = the ID of $x$;
        output.collect ($key', value'$);
    end
end
```

Algorithm 5. PACRSDC-Reduce (*key*, *V*)

```
Input:
//key: the decision value of the object
//value: the list of the ID from different host
Output:
$\langle key', value' \rangle$ pair, where key' is the decision value of the object and value' is the set of the ID.
begin
    let $value' \leftarrow \emptyset$;
   for each $ID \in V$ do
        $value' \leftarrow value' \cup ID$;
    end
    output.collect ($key', value'$);
end
```

Example 5 (*Example 4 continued*). According to the **Algorithm PACRSDC-Map** and **Algorithm PACRSDC-Reduce**, we compute rough set decision classes in parallel as follows:

1. PACRSDC-Map step
   - Map1: $U_1/D = \{D_{11}, D_{12}, D_{13}\}$, where $D_{11} = \{x_1, x_5\}$, $D_{12} = \{x_2, x_3, x_4\}$, $D_{13} = \{x_6\}$.
   - Map2: $U_2/D = \{D_{21}, D_{22}, D_{23}\}$, where $D_{21} = \{x_7\}$, $D_{22} = \{x_8, x_{10}, x_{12}\}$, $D_{23} = \{x_9, x_{11}\}$.
2. PACRSDC-Reduce step
   - Reduce1: $D_1 = D_{11} \cup D_{21} = \{x_1, x_5, x_7\}$.
   - Reduce2: $D_2 = D_{12} \cup D_{22} = \{x_2, x_3, x_4, x_8, x_{10}, x_{12}\}$.
   - Reduce3: $D_3 = D_{13} \cup D_{23} = \{x_6, x_9, x_{11}\}$,

where $\vec{D_1} = \langle 0 \rangle,\ \vec{D_2} = \langle 1 \rangle,\ \vec{D_3} = \langle 2 \rangle$.

3.3. The parallel method for computing rough set approximations

By Theorems 2 and 3, both rough set equivalence classes and decision classes can be computed in parallel based on MapReduce. In this subsection, we give the detailed process for computing rough set approximations in parallel.

Definition 6. Given a decision table $S = (U, C \cup D, V, f)$, $B \subseteq C$. $U/B = \{E_1, E_2, \ldots, E_t\}$ and $U/D = \{D_1, D_2, \ldots, D_r\}$. If $E_k \cap D_j \neq \emptyset$, $E_k \in U/B$, $D_j \in U/D$, we call E_k and D_j association, denoted by $Ass(E_k, D_j) = True$; otherwise, $Ass(E_k, D_j) = False$.

Theorem 4. *Given a decision table $S = (U, C \cup D, V, f)$, $B \subseteq C$. $U/B = \{E_1, E_2, \ldots, E_t\}$ and $U/D = \{D_1, D_2, \ldots, D_r\}$. Let $S = \bigcup_{i=1}^m S_i$, where $S_i = (U_i, C \cup D, V, f)$. $\forall i \in \{1, 2, \ldots, m\}$, $U_i/B = \{E_{i1}, E_{i2}, \ldots, E_{ip_i}\}$ and $U_i/D = \{D_{i1}, D_{i2}, \ldots, D_{iq_i}\}$. If $Ass(E_{ih}, D_{ig}) = True$ ($E_{ih} \in U_i/B$, $D_{ig} \in U_i/D$, $i \in \{1, 2, \ldots, m\}$) and $\vec{E_{ih}} = \vec{E_k}, \vec{D_{ig}} = \vec{D_j}(E_k \in U/B, D_j \in U/D)$, then $Ass(E_k, D_j) = True$.*

Proof. Since $Ass(E_{ih}, D_{ig}) = True$, then $E_{ih} \cap D_{ig} \neq \emptyset$. Suppose $x \in E_{ih} \cap D_{ig}$, then $x \in E_{ih}$ and $x \in D_{ig}$. As $\vec{E_{ih}} = \vec{E_k}$ and $\vec{D_{ig}} = \vec{D_j}$, according to Theorems 2 and 3, $E_{ih} \subseteq E_k$ and $D_{ig} \subseteq D_j$. Thus, $x \in E_k$, $x \in D_j$, and $E_k \cap D_j \neq \emptyset$. So $Ass(E_k, D_j) = True$.

According to Theorem 4, the associations between equivalence classes and decision classes of the decision table S can also be executed in parallel.

We design a Parallel Algorithm for Construction of Associations between Equivalence classes and Decision classes (PACAED) based on MapReduce. The **Algorithm PACAED** is divided into **Algorithm PACAED-Map** and **Algorithm PACAED-Reduce**, as shown in Algorithms 6 and 7, respectively. □

Algorithm 6. PACAED-Map (*key*, *value*)

```
Input:
//key: document name
//value: $S_i = \{U_i, C \cup D, V, f\}$
Output:
$\langle key', value' \rangle$ pair, where key' is the information set of the object with respect to the attribute set $C \cup D$ and value' is empty.
begin
    for each $x \in U_i$ do
        let $key' = \vec{x}_C + \vec{x}_D$;
        let $value' = \emptyset$;
        output.collect (key', value');
    end
end
```

Algorithm 7. PACAED-Reduce (*key*, *V*)

```
Input:
//key: the information set of the object with respect to the attribute set $C \cup D$
//value: empty
Output:
$\langle key', value' \rangle$ pair, where key' is the information set with respect to the attribute set $C \cup D$ and value' is empty.
```

```
begin
    $key' = key$;
    $value' = \emptyset$;
    output.collect ($key', value'$);
end
```

Example 6 (*Example 4 continued*). According to the **Algorithm PACAED-Map** and **Algorithm PACAED-Reduce**, we construct the associations between equivalence classes and decision classes in parallel as follows:

1. PACAED-Map step According to **Algorithm PACAED-Map**, the value of the output is empty. Hence, we just give the keys of the output.
   - Map1:

$$\vec{x}_{1C} + \vec{x}_{1D} = \langle 0,0 \rangle + \langle 0 \rangle = \vec{E}_1 + \vec{D}_1; \vec{x}_{2C} + \vec{x}_{2D} = \langle 0,0 \rangle + \langle 1 \rangle = \vec{E}_1 + \vec{D}_2;$$

$$\vec{x}_{3C} + \vec{x}_{3D} = \langle 1,0 \rangle + \langle 1 \rangle = \vec{E}_2 + \vec{D}_2; \vec{x}_{4C} + \vec{x}_{4D} = \langle 1,1 \rangle + \langle 1 \rangle = \vec{E}_3 + \vec{D}_2;$$

$$\vec{x}_{5C} + \vec{x}_{5D} = \langle 0,0 \rangle + \langle 0 \rangle = \vec{E}_1 + \vec{D}_1; \vec{x}_{6C} + \vec{x}_{6D} = \langle 1,1 \rangle + \langle 2 \rangle = \vec{E}_3 + \vec{D}_3 .$$

   - Map2:

$$\vec{x}_{7C} + \vec{x}_{7D} = \langle 0,0 \rangle + \langle 0 \rangle = \vec{E}_1 + \vec{D}_1; \vec{x}_{8C} + \vec{x}_{8D} = \langle 1,1 \rangle + \langle 1 \rangle = \vec{E}_3 + \vec{D}_2;$$

$$\vec{x}_{9C} + \vec{x}_{9D} = \langle 1,1 \rangle + \langle 2 \rangle = \vec{E}_3 + \vec{D}_3; \vec{x}_{10C} + \vec{x}_{10D} = \langle 0,1 \rangle + \langle 1 \rangle = \vec{E}_4 + \vec{D}_2;$$

$$\vec{x}_{11C} + \vec{x}_{11D} = \langle 1,0 \rangle + \langle 2 \rangle = \vec{E}_2 + \vec{D}_3; \vec{x}_{12C} + \vec{x}_{12D} = \langle 1,1 \rangle + \langle 1 \rangle = \vec{E}_3 + \vec{D}_2 .$$

2. PACAED-Reduce step According to **Algorithm PACAED-Reduce**, the values of the input are empty. Therefore, the values of the output are empty and we give the keys of the output, too.
   - Reduce1: $\vec{E}_1 + \vec{D}_1$.
   - Reduce2: $\vec{E}_1 + \vec{D}_2$.
   - Reduce3: $\vec{E}_2 + \vec{D}_2$.
   - Reduce4: $\vec{E}_3 + \vec{D}_2$.
   - Reduce5: $\vec{E}_3 + \vec{D}_3$.
   - Reduce6: $\vec{E}_2 + \vec{D}_3$.
   - Reduce7: $\vec{E}_4 + \vec{D}_2$,

   which means $Ass(E_1,D_1) = True$, $Ass(E_1,D_2) = True$, $Ass(E_2,D_2) = True$, $Ass(E_3,D_2) = True$, $Ass(E_3,D_3) = True$, $Ass(E_2,D_3) = True$ and $Ass(E_4,D_2) = True$.

Theorem 5. *Given a decision table $S = (U, C \cup D, V, f)$, $B \subseteq C$. $U/B = \{E_1, E_2, \ldots, E_t\}$ and $U/D = \{D_1, D_2, \ldots, D_r\}$. If $Ass(E_k, D_j) = True$, $E_k \in U/B$, $D_j \in U/D$, then $E_k \subseteq \overline{R}_B(D_j)$.*

Proof. The proof follows directly from the definition of upper approximation. □

Theorem 6. *Given a decision table $S = (U, C \cup D, V, f)$, $B \subseteq C$. $U/B = \{E_1, E_2, \ldots, E_t\}$ and $U/D = \{D_1, D_2, \ldots, D_r\}$.*

(1) *If $Ass(E_k, D_{j_1}) = True$ and $Ass(E_k, D_{j_2}) = True$ ($j_1 \neq j_2, j_1, j_2 \in \{1, 2, \ldots, r\}$, $k \in \{1, 2, \ldots, t\}$), then $\forall D_j \in U/D$, $E_k \nsubseteq \underline{R}_B(D_j)$, denoted by $Boolean(E_k) = False$;*
(2) *If $\exists! j \in \{1, 2, \ldots, r\}$, $Ass(E_k, D_j) = True (k \in \{1, 2, \ldots, t\})$, then $E_k \subseteq \underline{R}_B(D_j)$, denoted by $Boolean(E_k) = True$.*

Proof. The proof follows directly from the definition of lower approximation. □

According to Theorems 5 and 6, lower and upper approximations are computed by associations between equivalence classes and decision classes. However, if we compute the approximations directly, memory may overflow since equivalence classes and decision classes both contain too many objects while dealing the large data set. Hence, we design an Algorithm for Computation of the Indexes of Rough Set Approximations by means of Associations (ACIRSAA), which gives the set of information set for each decision class. The detailed process is shown in Algorithm 8.

Algorithm 8. An Algorithm for Computation of the Indexes of Rough Set Approximations by means of Associations (ACIRSAA)

```
Input:
Associations between equivalence classes and decision classes: $E_k + D_j$ (means $Ass(E_k,D_j) = True$) $k \in \{1,2,\dots,t\}$,
$j \in \{1,2,\dots,r\}$.
Output:
The indexes of the approximations: $\underline{RI}_C(D_j)$ and $\overline{RI}_C(D_j), j = 1,2,\dots,r$.
begin
    % Computing the indexes of upper approximations
    for each $D_j$ do
        $\overline{RI}_C(D_j) \leftarrow \emptyset$;
        for each $E_k$ do
            if $Ass(E_k,D_j) = True$ then
                $\overline{RI}_C(D_j) \leftarrow \overline{RI}_C(D_j) \cup \vec{E_k}$;
            end
        end
    end
    % Computing the indexes of lower approximations
    for each $E_k$ do
        Let $Boolean(E_k) = True$;
    end
    for each $E_k$ do
        if $Ass(E_k, D_{j_1}) = True$ and $Ass(E_k, D_{j_2}) = True, j_1 \neq j_2, j_1, j_2 \in \{1,2,\dots,r\}$ then
            $Boolean(E_k) = False$;
        end
    end
    for each $D_j$ do
        $\underline{RI}_C(D_j) \leftarrow \emptyset$;
        for each $E_k$ do
            if $Boolean(E_k) = True$ and $Ass(E_k,D_j) = True$ then
                $\underline{RI}_C(D_j) \leftarrow \underline{RI}_C(D_j) \cup \vec{E_k}$;
            end
        end
    end
    return $\underline{RI}_C(D_j)$ and $\overline{RI}_C(D_j), j = 1,2,\dots,r$.
end
```

Example 7 (*Continuation of Example* 6). According to **Algorithm ACIRSAA**, since $Ass(E_1,D_1) = True$, $Ass(E_1,D_2) = True$, $Ass(E_2,D_2) = True$, $Ass(E_3,D_2) = True$, $Ass(E_3,D_3) = True$, $Ass(E_4,D_2) = True$ and $Ass(E_2,D_3) = True$, then $\overline{RI}_C(D_1) = \vec{E_1}$, $\overline{RI}_C(D_2) = \vec{E_1} \cup \vec{E_2} \cup \vec{E_3} \cup \vec{E_4}$, $\overline{RI}_C(D_3) = \vec{E_2} \cup \vec{E_3}$ and $Boolean(E_1) = False$, $Boolean(E_2) = False$, $Boolean(E_3) = False$, $Boolean(E_4) = True$. Hence, $\underline{RI}_C(D_1) = \emptyset$, $\underline{RI}_C(D_2) = \vec{E_4}$, $\underline{RI}_C(D_3) = \emptyset$.

After computing the indexes of approximations, we output the approximations directly. The frame diagram of the parallel method for computing rough set approximations is shown in Fig. 4.

Theorem 7. *The rough set approximations obtained by the parallel method are the same as those obtained by the serial method.*

Proof. Followed directly from Definitions 3 and 4 and Theorems 6 and 7. □

4. Experimental analysis

In Section 3, we proposed three parallel algorithms, Algorithms PACRSEC, PACRSDC, PACAED and one serial algorithm, Algorithm ACIRSAA. Three parallel algorithms are coded as one program in Java, and the serial algorithm ACIRSAA is coded in C++. We evaluate the performance of the proposed three parallel algorithms with respect to its speedup, scaleup and size-

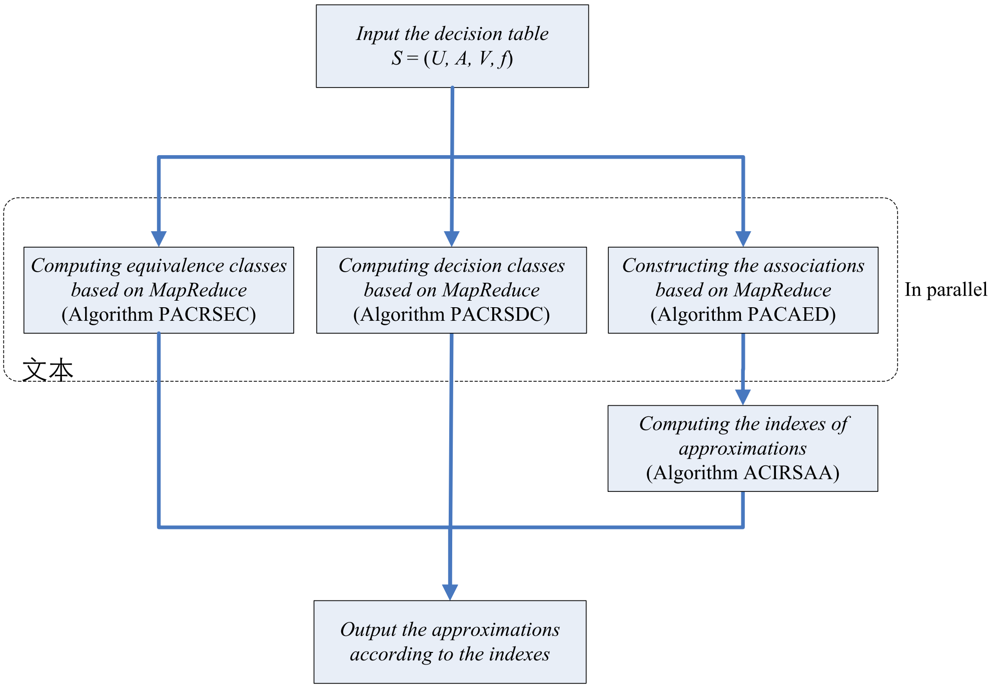
Input the decision table
S = (U, A, V, f)
Computing equivalence classes based on MapReduce (Algorithm PACRSEC)
Computing decision classes based on MapReduce (Algorithm PACRSDC)
Constructing the associations based on MapReduce (Algorithm PACAED)
In parallel
文本
Computing the indexes of approximations (Algorithm ACIRSAA)
Output the approximations according to the indexes

Fig. 4. The parallel method for computing rough set approximations.

up [35,38]. We run the experiments on the Hadoop Mapreduce platform [40], where Hadoop MapReduce is a programming model and software framework for developing applications that rapidly process massive data in parallel on large clusters of compute nodes, each of which has four 2.13 GHz cores and 12 GB of memory. Hadoop version 0.20.2 and Java 1.6.0.01 are used as the MapReduce system for all experiments. In this paper, we only evaluate the efficiency of the algorithms and not its accuracy since the parallel algorithms produce the same results as those of the serial method (cf. Theorem 7).

For the following evaluations, we utilize the large data set KDDCup-99 from the machine learning data repository, University of California at Irvine [24]. The data set KDDCup-99 consists of approximately five million records. Each records consists of 1 decision attribute and 41 condition attributes, where 6 are categorical and 35 are numeric. Since our parallel algorithms deal with categorical attributes, the 35 numeric attributes are discretized firstly and the data set is renamed KDD99. Besides, three synthetic data sets have been generated by means of the WEKA data generator [9]. The data sets are outlined in Table 4.

Since the available serial method (see Fig. 2) performs in-memory processing, it can not deal with large data sets. The proposed parallel method (see Fig. 4) performs on the Hadoop platform which may remove the data size limit due to transparent spilling. Therefore, we here only give the computational time of the parallel method in detail. Table 5 shows the com-

Table 4
A description of real and synthetic data sets.

	Data sets	Records	Attributes	Classes	Size (GB)
1	KDD99	4,898,421	41	23	0.48
2	Weka-1.7G	32,000,000	10	35	1.70
3	Weka-3.2G	40,000,000	15	45	3.20
4	Weka-6.4G	80,000,000	15	58	6.40

Table 5
The computational time of the parallel algorithms and Algorithm ACIRSAA (unit: seconds).

Data sets	Three parallel algorithms on different nodes								Algorithm ACIRSAA on 1 node
	1	2	3	4	5	6	7	8	
KDD99	246	174	144	124	109	107	105	99	0.08
Weka-1.7G	945	580	421	319	261	249	228	188	0.05
Weka-3.2G	1505	861	589	433	371	347	295	285	3.46
Weka-6.4G	3128	1661	1163	889	721	630	559	446	3.49

putational time of three parallel algorithms on different nodes and Algorithm ACIRSAA on 1 node. As the number of the nodes increases, the computational time of the parallel algorithms becomes smaller. Moreover, the computational time of Algorithm ACIRSAA is much smaller than that of the parallel algorithms.

In the following, we examine the speedup, scaleup and sizeup characteristics of the proposed parallel algorithms.

4.1. Speedup

To measure the speedup, we keep the data set constant and increase the number of nodes (computers) in the system. Speedup given by the larger system is defined by the following formula [35]:

$$Speedup(p) = \frac{T_1}{T_p}$$

where p is the number of nodes (computers), T_1 is the execution time on one node, T_p is the execution time on p nodes.

The ideal parallel algorithm demonstrates linear speedup: a system with p times the number of computers yields a speedup of p. However, linear speedup is difficult to achieve because the communication cost increases with the number of clusters becomes large.

We perform the speedup evaluation on data sets with quite different sizes and structures. The number of nodes (computers) varied from 1 to 8. Fig. 5 shows the speedup for these data sets. As the results show, the proposed parallel algorithms have a very good speedup performance. KDD99 has a lower speedup curve, because the size of KDD99 is too small. As the size of the data set increases, the speedup performs better. Therefore, the proposed parallel algorithms can treat massive data efficiently.

4.2. Scaleup

Scaleup is defined as the ability of a p-times larger system to perform a p-times larger job in the same execution time [35].

$$Scaleup(D, p) = \frac{T_{D_1}}{T_{D_p}}$$

where D is the data set, T_{D_1} is the execution time for D on one node, T_{D_p} is the execution time for $p \times D$ on p nodes.

To see how well the proposed parallel algorithms handle larger data sets when more nodes are available, we have performed scaleup experiments where we increased the size of the data sets in direct proportion to the number of nodes in the system. For the data set Weka-6.4G, *e.g.*, 10,000,000 records on one node and 80,000,000 records on eight nodes. Fig. 6 shows the performance results of the data sets. Clearly, the proposed parallel algorithms scale very well.

4.3. Sizeup

Sizeup is defined as the following formula [35]:

$$Sizeup(D, p) = \frac{T_{S_p}}{T_{S_1}}$$

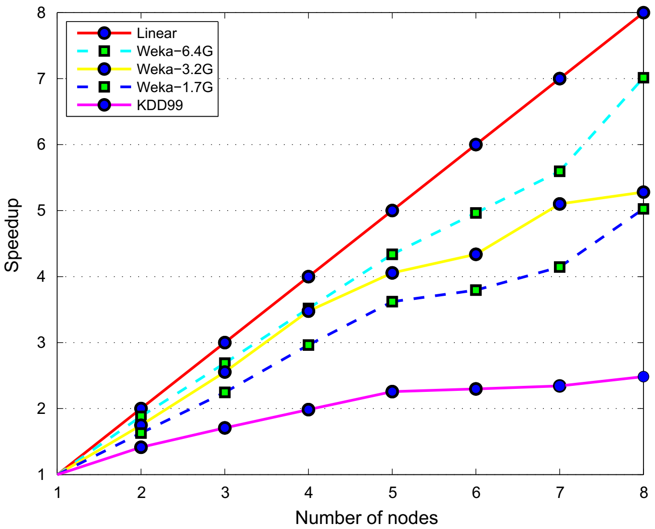

Fig. 5. Speedup.

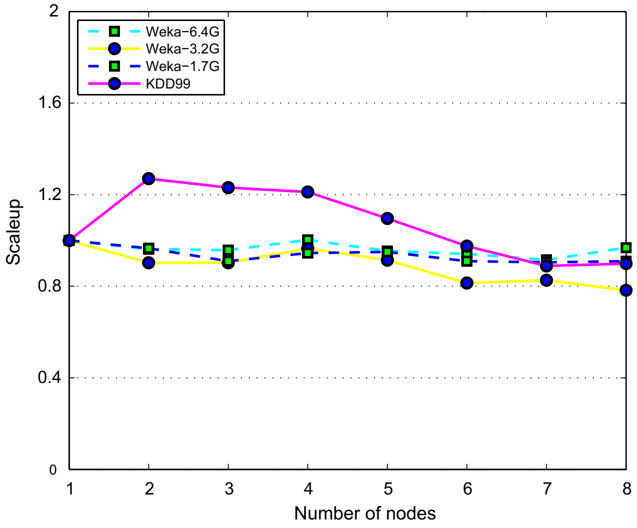

Fig. 6. Scaleup.

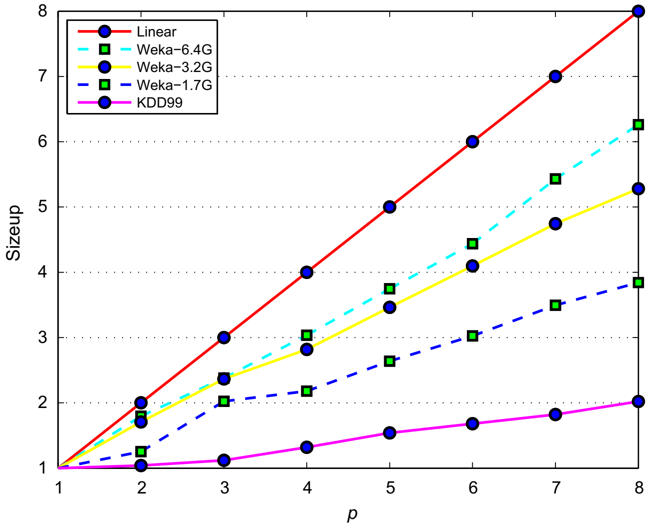

Fig. 7. Sizeup.

where T_{Sp} is the execution time for $p \times D$, T_{S_1} is the execution time for D.

Sizeup analysis holds the number of computers in the system constant, and grows the size of the data sets by the factor p. Sizeup measures how much longer it takes on a given system, when the size of data set is p-times larger than that of the original data set. For our last set of experiments, we fixed the number of nodes (computers) to four to measure the performance of sizeup. Fig. 7 shows the sizeup results on four nodes. The graph shows that the proposed parallel algorithms have a good sizeup performance.

5. Conclusions

Many rough sets based algorithms for data mining have been proposed in the past decades. However, the enlarging data in applications make algorithms based on rough sets a challenging task. Since the computation of rough set approximations is the necessary step, the development of its efficient algorithms becomes an important task. In this paper, we proposed a parallel method for computing rough set approximations. The algorithms corresponding to the parallel method based on MapReduce were successfully designed. We used speedup, scaleup and sizeup to evaluate the performances of the proposed parallel algorithms. The experimental results on the real and synthetic data sets showed that the proposed parallel algorithms could effectively deal with large data sets in data mining. Our future research work will focus on applications of the proposed parallel method in attributes selection and rules extraction from massive data based on rough sets.

Acknowledgments

This work is supported by the National Science Foundation of China (Nos. 60873108, 61175047, 61100117), the Youth Social Science Foundation of the Chinese Education Commission (No. 11YJC630127), the Fundamental Research Funds for the Central Universities (No. SWJTU11ZT08) and the Doctoral Innovation Foundation of Southwest Jiaotong University (No. 2012ZJB), the Young Software Innovation Foundation of Sichuan Province (No. 2011-017), China.

References

[1] J. Berlińska, M. Drozdowski, Scheduling divisible MapReduce computations, Journal of Parallel and Distributed Computing 71 (2011) 450–459.
[2] J. Błaszczyński, R. Słowiński, M. Szeląg, Sequential covering rule induction algorithm for variable consistency rough set approaches, Information Sciences 181 (2011) 987–1002.
[3] H. Chen, T. Li, S. Qiao, D. Ruan, A rough set based dynamic maintenance approach for approximations in coarsening and refining attribute values, International Journal of Intelligent Systems 25 (2010) 1005–1026.
[4] C.H. Cheng, T.L. Chen, L.Y. Wei, A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting, Information Sciences 180 (2010) 1610–1629.
[5] Y. Cheng, D. Miao, Q. Feng, Positive approximation and converse approximation in interval-valued fuzzy rough sets, Information Sciences 181 (2011) 2086–2110.
[6] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, in: Proceedings of Operating Systems Design and Implementation (OSDI), San Francisco, CA, 2004, pp. 137–150.
[7] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (2008) 107–113.
[8] J. Ekanayake, S. Pallickara, G. Fox, MapReduce for data intensive scientific analyses, in: Proceedings of Fourth IEEE International Conference on eScience, Indianapolis, Indiana, USA, 2008, pp. 277–284.
[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explorations 11 (1) (2009) 10–18.
[10] J. Han, M. Kamber, Data Mining: Concepts and Techniques, second ed., Morgan Kaufman, San Francisco, 2006.
[11] P.Y. Hsu, Y.L. Chen, C.C. Ling, Algorithms for mining association rules in bag databases, Information Sciences 166 (2004) 31–47.
[12] Q. Hu, J. Liu, D. Yu, Mixed feature selection based on granulation and approximation, Knowledge-Based Systems 21 (2008) 294–304.
[13] Q. Hu, Z. Xie, D. Yu, Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation, Pattern Recognition 40 (2007) 3509–3521.
[14] Q. Hu, D. Yu, J. Liu, C. Wu, Neighborhood rough set based heterogeneous feature subset selection, Information Sciences 178 (2008) 3577–3594.
[15] Q. Hu, D. Yu, Z. Xie, Neighborhood classifiers, Expert Systems with Applications 34 (2008) 866–876.
[16] Y.C. Hu, G.H. Tzeng, C.M. Chen, Deriving two-stage learning sequences from knowledge in fuzzy sequential pattern mining, Information Sciences 159 (2004) 69–86.
[17] R. Lämmel, Google's MapReduce programming model – revisited, Science of Computer Programming 70 (2008) 1–30.
[18] K. Lagus, S. Kaski, T. Kohonen, Mining massive document collections by the websom method, Information Sciences 163 (2004) 135–156.
[19] T. Li, D. Ruan, W. Geert, J. Song, Y. Xu, A rough sets based characteristic relation approach for dynamic attribute generalization in data mining, Knowledge-Based Systems 20 (2007) 485–494.
[20] Y. Li, S. Zhu, X.S. Wang, S. Jajodia, Looking into the seeds of time: discovering temporal patterns in large transaction sets, Information Sciences 176 (2006) 1003–1031.
[21] J.J.H. Liou, G.H. Tzeng, A dominance-based rough set approach to customer behavior in the airline market, Information Sciences 180 (2010) 2230–2238.
[22] D. Liu, T. Li, D. Ruan, J. Zhang, Incremental learning optimization on knowledge discovery in dynamic business intelligent systems, Journal of Global Optimization 51 (2011) 325–344. 10.1007/s10898-010-9607-8.
[23] Z. Lv, Y. Hu, H. Zhong, J. Wu, B. Li, H. Zhao, Parallel k-means clustering of remote sensing images based on MapReduce, in: F. Wang, Z. Gong, X. Luo, J. Lei (Eds.), Web Information Systems and Mining, Lecture Notes in Computer Science, vol. 6318, Springer, Berlin/Heidelberg, 2010, pp. 162–170.
[24] D. Newman, S. Hettich, C. Blake, C. Merz, UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science, Irvine, CA, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
[25] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data, System Theory, Knowledge Engineering and Problem Solving, vol. 9, Kluwer Academic Publishers, Dordrecht, 1991.
[26] Z. Pawlak, A. Skowron, Rough sets and boolean reasoning, Information Sciences 177 (2007) 41–73.
[27] Z. Pawlak, A. Skowron, Rough sets: some extensions, Information Sciences 177 (2007) 28–40.
[28] Z. Pawlak, A. Skowron, Rudiments of rough sets, Information Sciences 177 (2007) 3–27.
[29] Y. Qian, J. Liang, C. Dang, Incomplete multigranulation rough set, IEEE Transactions on Systems, Man and Cybernetics – Part A 40 (2010) 420–431.
[30] Y. Qian, J. Liang, W. Pedrycz, C. Dang, Positive approximation: an accelerator for attribute reduction in rough set theory, Artificial Intelligence 174 (2010) 597–618.
[31] Y. Qian, J. Liang, Y. Yao, C. Dang, MGRS: a multi-granulation rough set, Information Sciences 180 (2010) 949–970.
[32] C. Ranger, R. Raghuraman, A. Penmetsa, G.R. Bradski, C. Kozyrakis, Evaluating MapReduce for multi-core and multiprocessor systems, in: Proceedings of International Symposium on High-Performance Computer Architecture (HPCA), Phoenix, 2007, pp. 13–24.
[33] J. Wan, W. Yu, X. Xu, Design and implement of distributed document clustering based on MapReduce, in: Proceedings of the Second Symposium International Computer Science and Computational Technology (ISCSCT), Huangshan, PR China, 2009, pp. 278–280.
[34] T. White, Hadoop: The Definitive Guide, first ed., O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA, 2009.
[35] 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. 10.1023/A:1009884809343.
[36] Y. Yang, Z. Chen, Z. Liang, G. Wang, Attribute reduction for massive data based on rough set theory and MapReduce, in: J. Yu, S. Greco, P. Lingras, G. Wang, A. Skowron (Eds.), Rough Set and Knowledge Technology, Lecture Notes in Computer Science, vol. 6401, Springer, Berlin/Heidelberg, 2010, pp. 672–678.
[37] Y. Yuan, Y. Wu, X. Feng, J. Li, G. Yang, W. Zheng, VDB-MR: MapReduce-based distributed data integration using virtual database, Future Generation Computer Systems 26 (2010) 1418–1425.
[38] W. Zhao, H. Ma, Q. He, Parallel k-means clustering based on MapReduce, Lecture Notes in Computer Science 5931 (2009) 674–679.
[39] D. Zinn, S. Bowers, S. Köhler, B. Ludäscher, Parallelizing XML data-streaming workflows via MapReduce, Journal of Computer and System Sciences 76 (2010) 447–463.
[40] Hadoop: open source implementation of MapReduce, <http://hadoop.apache.org/mapreduce/>.