Information Sciences 279 (2014) 671–690

Contents lists available at ScienceDirect

Information Sciences

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

Parallel attribute reduction algorithms using MapReduce

Jin Qian [a,*], Duoqian Miao [b,c], Zehua Zhang [b,c], Xiaodong Yue [b,d]

[a] Key Laboratory of Cloud Computing and Intelligent Information Processing of Changzhou City, Jiangsu University of Technology, Changzhou 213015, China
[b] Key Laboratory of Embedded System and Service Computing, Ministry of Education of China, Tongji University, Shanghai 201804, China
[c] Department of Computer Science and Technology, Tongji University, Shanghai 201804, China
[d] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

ARTICLE INFO

Article history:
Received 17 September 2012
Received in revised form 31 March 2014
Accepted 8 April 2014
Available online 19 April 2014

Keywords:
Rough set
Parallel attribute reduction
MapReduce
Data mining

ABSTRACT

Attribute reduction is the key technique for knowledge acquisition in rough set theory. However, it is still a challenging task to perform attribute reduction on massive data. During the process of attribute reduction on massive data, the key to improving the reduction efficiency is the effective computation of equivalence classes and attribute significance. Aiming at this problem, we propose several parallel attribute reduction algorithms in this paper. Specifically, we design a novel structure of $\langle key, value\rangle$ pair to speed up the computation of equivalence classes and attribute significance and parallelize the traditional attribute reduction process based on MapReduce mechanism. The different parallelization strategies of attribute reduction are also compared and analyzed from the theoretic view. Abundant experimental results demonstrate the proposed parallel attribute reduction algorithms can perform efficiently and scale well on massive data.

1. Introduction

Because of the fast-growing data in scientific and industrial areas, traditional data mining algorithms are facing the challenges from both the perspectives of data storage and computational complexity. Under the strategy of serial computing on single node, it is difficult to construct enough memory to fit in the massive data and explore an effective solution in the huge search space [12,35]. As an important preprocessing technique for data mining, attribute reduction, also called feature selection, is helpful to reduce feature space and improve classification performance through removing redundant and irrelevant attributes. However, due to the inefficient data structure and the high computational cost, most existing attribute reduction algorithms cannot handle the massive data well. Thus it is desired to design an efficient attribute reduction strategy for massive data [9].

As an effective tool of attribute reduction, rough set theory [10,28,29] has been successfully developed to find a reduct or multiple reducts. To achieve an efficient attribute reduction, different algorithms [4,14–16,21,23–25,28,30–32,34,38,40,42,46,47,50,51,56] in rough set theory were proposed in the last decades. Xu et al. [42] designed a fast attribute reduction algorithm, in which the universe partition U/A is calculated recursively to reduce the searching space. Hu et al. [15] only computed the objects in $U - POS_A(D)$ for the neighborhood rough set model to improve the efficiency of the attribute reduction process when the size of the selected attribute set A increases. Qian et al. [30] proposed positive

* Corresponding author. Address: School of Computer Engineering, Jiangsu University of Technology, Changzhou 213001, PR China. Tel.: +86 519 86953252.
E-mail addresses: qjqjlqyf@163.com (J. Qian), miaoduoqian@163.com (D. Miao), zehua.zzh@gmail.com (Z. Zhang), yswantfly@gmail.com (X. Yue).

http://dx.doi.org/10.1016/j.ins.2014.04.019
0020-0255/

approximation to characterize the granulation structure of a rough set using a granulation order for improving the time efficiency of the positive region reduction algorithm. Using the technique of counting sort, Qian et al. [32] proposed a fast attribute reduction algorithm based on positive region, discernibility matrix and information entropy. Li et al. [21] presented quick attribute reduction algorithms for the assignment reduct, the distribution reduct, and the maximum distribution reduct in the inconsistent decision table. Yang et al. [46] presented the reduct of test cost multigranulation decision system. The best time complexity among these attribute reduction algorithms is $\max(O(|C||U|), O(|C|^2|U/C|))$ ($|\cdot|$ denotes the cardinality of a set, $|C|$ and $|U|$ denote the number of conditional attributes and objects respectively). This indicates that if a data set is high-dimensional or huge-volumed, the reduction algorithms above cannot perform well on it. Moreover, sampling techniques can be applied to reduce the data size but sampling methods should satisfy some hypothesis and may lose valuable data samples. Therefore, researchers try to parallelize the traditional attribute reduction algorithms to improve their efficiency on massive data.

Parallel computing may be a good solution to attribute reduction algorithm, in which many calculations are carried out simultaneously in task and/or data parallel [2]. Task parallelism targets at running different tasks in parallel while data parallelism aims to perform the same operation on multiple data sets. The strategies of both task parallelism and data parallelism are utilized to parallelize the attribute reduction process. For small datasets, most existing parallel attribute reduction algorithms [26,36,39] focus on task parallelism and decompose the whole computation task into several subtasks which can be processed on different nodes. These parallel algorithms assume that data can be loaded into the main memory of a single machine and the minimal reducts can be found in task parallel. Mohammad et al. [26] distributed the entire data into multiple nodes and applied migration to find out sufficiently good reducts in a large search space through parallel computing. Combining parallel genetic algorithm with symbiotic evolution, Wang and Wu [39] also proposed a new approach to attribute reduction for informations system with a large number of attributes. Susmaga [36] decomposed a reduct generation task into a number of subtasks and performed the parallel computations in a tree-like manner. For large datasets, parallel calculations of attribute reduction algorithms [6,18] emphasize data parallelism. Deng et al. [6] decomposed the whole dataset into many sub-tables and proposed an approach for parallel reduction from a series of decision subsystems. Liang et al. [18] considered the sub-tables within a large-scale dataset as small granularities. The reducts of small granularities can be computed separately and finally be fused together to generate the reduct of the whole data. However, these reducts are not guaranteed to be the same as those discovered from the whole dataset since these subsystems (sub-tables) do not exchange the information for each other. Thus, all these aforementioned algorithms cannot generate the precise reducts for massive datasets. Moreover, these algorithms generally must encode the complex program to explicitly manage datasets, and consider the details of tolerating machine failures and restarting individual tasks.

Recently, a simple parallel computation approach—MapReduce [5], has been received considerable attention, which is a scalable parallel programming model for data-intensive and computation-intensive applications on machine clusters. MapReduce model has the significant advantages as follows. It hides many system-level details from the user and automatically parallelizes the computation across large-scale clusters of machines. It also handles machine failures and schedules inter-machine communications to make efficient use of networks and disks. It has been demonstrated that MapReduce is helpful to achieve an effective solution of complex computational task on massive data through coding with map/reduce functions. Furthermore, the distributed file systems inherent in MapReduce [8,11] provides us a scalable mechanism for storing massive datasets. MapReduce has been successfully applied in data mining [1,12], machine learning [3,22,35,53] and web indexing [55]. To the best of our knowledge, there are only a few works of using MapReduce programming model for attribute reduction in rough set theory [33,45]. Therefore, it is required to thoroughly discuss and develop some parallel attribute reduction algorithms using MapReduce for large data.

Our interest in this paper is to investigate the following issue: How to use MapReduce programming model to design an efficient parallel algorithm for attribute reduction? We first propose the parallelism of traditional attribute reduction algorithms. Then, we discuss the parallel and serial operations among the reduction algorithms and analyze three parallelization strategies. By constructing the proper $\langle key, value\rangle$ pairs and implementing map/reduce functions, we realize the parallel computation of equivalence classes and attribute significances. Finally, we design the parallel algorithms for acquiring the core attributes and a reduct in both data and task parallel. All these parallel algorithms are implemented using MapReduce on Hadoop [11]. Experimental results demonstrate that our proposed algorithms can efficiently deal with large-scale datasets.

The rest of this paper is organized as follows. Section 2 reviews the preliminary concepts in rough set theory and the MapReduce framework. In Section 3, the parallelism of the attribute reduction process is proposed and different parallelization strategies are compared and further analyzed. Based on this, the parallel attribute reduction algorithms in both forms of data parallel and task parallel are designed using MapReduce. Section 4 presents the experimental results and validates the efficiency of the proposed parallel attribute reduction algorithms. Finally, the paper work is concluded in Section 5.

2. Preliminaries

In this section, we will review the basic notions of the Pawlak rough set model in [25,28,29,32], and MapReduce programming model in cloud computing [5].

2.1. Pawlak rough set model

For classification tasks, we consider a decision table which is defined as: $S = (U, At = C \cup D, \{V_a | a \in At\}, \{I_a | a \in At\})$, where $U = \{x_1, x_2, \ldots, x_n\}$ is a finite non-empty set of objects, At is a finite nonempty set of attributes, $C = \{c_1, c_2, \ldots, c_m\}$ is a set of conditional attributes describing the objects, and D is a set of decision attributes that indicates the classes of objects. V_a is a nonempty set of values of $a \in At$. I_a is an information function that maps an object x in U to exactly one value v in V_a, that is, $I_a(x) = v$. For simplicity, we assume $D = \{d\}$ in this paper, where d is a decision attribute which has k different decision values, and $V_d = \{1, 2, \ldots, \mathrm{k}\}$. A table with multiple decision attributes can be easily transformed into a table with a single decision attribute by considering the Cartesian product of the original decision attributes.

Let $A \subseteq U \times U$ an equivalence relation on U. The equivalence relation A determines a partition of U, denoted by $U/IND(A)$, or simply π_A. The equivalence class of $U/IND(A)$ containing object x is given by $[x]_{IND(A)} = \{y \in U | (x, y) \in IND(A)\}$. For simplicity, we write $[x]_A$ instead of $[x]_{IND(A)}$ if $IND(A)$ is understood.

Consider a partition $\pi_D = \{D_1, D_2, \ldots, D_k\}$ of the universe U with respect to the decision attribute D and another partition $\pi_A = \{A_1, A_2, \ldots, A_r\}$ defined by a set of conditional attributes A. The equivalence classes induced by the partition are the basic blocks to construct the Pawlak rough set approximations.

Definition 1. For a decision class $D_i \in \pi_D$, the lower and upper approximations of D_i with respect to a partition π_A are defined by Pawlak [28]:

$$\begin{aligned} &\underline{apr}_A(D_i) = \{x \in U | [x]_A \subseteq D_i\}; \\ &\overline{apr}_A(D_i) = \{x \in U | [x]_A \cap D_i \neq \emptyset\}. \end{aligned} \tag{1}$$

Definition 2. For a decision table S, a positive region and boundary region of a partition π_D with respect to a partition π_A are defined as:

$$\begin{aligned} &POS_A(D) = \bigcup_{1 \leqslant i \leqslant k} \underline{apr}_A(D_i); \\ &BND_A(D) = \bigcup_{1 \leqslant i \leqslant k} (\overline{apr}_A(D_i) - \underline{apr}_A(D_i)). \end{aligned} \tag{2}$$

For the partition π_D, we can calculate its lower and upper approximations in terms of k two-class problems. $POS_A(D)$ is the positive region of decision and indicates the union of all the equivalence classes defined by π_A, in which each equivalence class induces a certain decision. $BND_A(D)$ is the boundary region of decision and is formed by the union of all the equivalence classes defined by π_A which induce the partial decisions. When A equals C, $POS_C(D)$ denotes the positive region in a decision table.

Definition 3. For a decision table S, all the objects in $POS_C(D)$ are called consistent objects. The objects in $U - POS_C(D)$ are called inconsistent objects. If $POS_C(D) = \mathrm{U}$, then the decision table is consistent; otherwise it is inconsistent.

In what follows, we present the measures of attribute significance for attribute reduction based on positive region and information entropy.

Definition 4. For a decision table S, let $A \subseteq C$ and $c \in C - A$, then the significance of attribute c is defined by:

$$sig_1(c, A, D) = \gamma_{A \cup \{c\}}(D) - \gamma_A(D) \tag{3}$$

where $\gamma_{A \cup \{c\}}(D) = \frac{|POS_{A \cup \{c\}}(D)|}{|U|}$.

Definition 5. For a decision table S, let $\pi_D = \{D_1, D_2, \ldots, D_k\}$, then information entropy of D is given by:

$$Info(D) = -\sum_{j=1}^{k} p(D_j) \log_2 p(D_j) \tag{4}$$

where $p(D_j) = \frac{n^j}{n}$ $(j = 1, \ldots, k)$, n is the number of the objects in U, n^j is the number of the objects in D_j.

Definition 6. For a decision table S, $\pi_A = \{A_1, A_2, \ldots, A_r\}$, $\pi_D = \{D_1, D_2, \ldots, D_k\}$, then the entropy $Info(A, D)$, conditional entropy of A conditioned on D, is given by

$$Info(A, D) = -\sum_{i=1}^{r} p(A_i) \sum_{j=1}^{k} p(D_j | A_i) \log_2 p(D_j | A_i) \tag{5}$$

where $p(A_i) = \frac{n_i}{n}$, $p(D_j|A_i) = \frac{n_i^j}{n_i}$ $(i = 1, \ldots, r;\ j = 1, \ldots, k)$, n_i is the number of the objects in A_i, n_i^j is the number of the objects of decision value j in A_i.

Definition 7. For a decision table S, let $A \subseteq C$ and $c \in C - A$, the significance of attribute c is defined as

$$sig_2(c, A, D) = Info(A, D) - Info(A \cup \{c\}, D) \tag{6}$$

Definition 8 [54]. For a decision table S, $A = \{a_1, a_2, \ldots, a_l\} \subseteq C$ be a subset of condition attributes. The information set with respect to A for any object $x \in U$ can be denoted by the tuple

$$\vec{x}_A = \langle I_{a_1}(x), I_{a_2}(x), \ldots, I_{a_l}(x) \rangle \tag{7}$$

For any $c \in C - A$, the information set with respect to $A \cup \{c\}$ for any object $x \in U$ can be denoted by

$$\vec{x}_{A \cup \{c\}} = \langle I_{a_1}(x), I_{a_2}(x), \ldots, I_{a_l}(x), I_c(x) \rangle \tag{8}$$

2.2. MapReduce programming model

MapReduce [5], proposed by Google, is a software framework for implementing the parallel algorithm. Unlike OpenMP and MPI, MapReduce provides a strategy to distribute computation without burdening the programmer with details of distributing computing, e.g. check-pointing and execution monitoring. Furthermore, MapReduce implementations usually supply their own distributed file systems that provide a scalable mechanism for storing large amounts of data.

In MapReduce programming model, the user of the MapReduce library expresses the computation as two functions: map and reduce. Conceptually, the map and reduce functions supplied by users have the following types:

$$map : \langle K_1, V_1 \rangle \rightarrow [\langle K_2, V_2 \rangle]$$
$$reduce : \langle K_2, [V_2] \rangle \rightarrow [\langle K_3, V_3 \rangle]$$

where all K_i and V_i $(i = 1, \ldots, 3)$ are user-defined data types, and the convention $[\cdots]$ is used throughout this paper to denote a list.

The MapReduce paradigm can be summarized as follows: (1) Data in the MapReduce framework can be represented as $\langle key, value \rangle$ pairs. (2) Map phase takes pairs $\langle K_1, V_1 \rangle$ as input and produces a set of intermediate $\langle K_2, V_2 \rangle$ pairs. The MapReduce library groups together all intermediate values V_2 associated with the same K_2 and passes them to the reduce phase. (3) Reduce phase accepts an K_2 and a set of values for that key, merges together these values to form a possibly smaller set of values, and finally outputs $\langle K_3, V_3 \rangle$ pairs.

Since MapReduce provides a robust and scalable framework and automatically splits the whole dataset into many data splits in natural sequence, we mainly focus on designing the proper $\langle key, value \rangle$ pairs as well as implementing the map and reduce functions for parallel attribute reduction algorithms.

3. Parallel attribute reduction algorithms using MapReduce

In this section, we first analyze the parallelism of classical attribute reduction algorithms using MapReduce. Then we compare three different parallelism strategies of attribute reduction and present how the reduction computations can be transformed into map and reduce operations. We also design a novel $\langle key, value \rangle$ pair to compute the equivalence classes and attribute significance. Finally, the parallel attribute reduction algorithms in the forms of both data and tasks are constructed through implementing the map and reduce functions.

3.1. The parallelism of classical attribute reduction algorithms

Before analyzing the parallelism of attribute reduction algorithm, we first have to discuss the parallel computation of equivalence classes.

Definition 9. For a decision table S, let S_i $(i = 1, 2, \ldots, L)$ denote a sub-decision table, if it satisfies (1) $S = \bigcup_{1 \leqslant i \leqslant L} S_i$; (2) $S_i \cap S_j = \emptyset$, where $i, j = 1, 2, \ldots, L$ and $i \neq j$, then S_i is called a data split.

By Definition 9, S_i corresponds to a data split in MapReduce framework. Thus, we can divide the whole dataset into many data splits using MapReduce.

Theorem 1. *For a decision table S, let $A \subseteq C$, $\pi_A = \{A_1, A_2, \ldots, A_r\}, S_i$ (i = 1,2,...,L) be a data split of S, $S_i/A = \{A_{i,1}, A_{i,2}, \ldots, A_{i,r}\}$, then $A_p = \bigcup_{1 \leqslant i \leqslant L} A_{i,p}$ (p = 1,2,...,r).*

Proof. If any two objects with respect to A are the same, they can be merged into one equivalence class. Thus, the same equivalence classes among different data splits can be combined as a larger equivalence class. Therefore, for $\pi_A = \{A_1, A_2, \ldots, A_r\}$ and $S_i/A = \{A_{i,1}, A_{i,2}, \ldots, A_{i,r}\}$, we have $A_p = \bigcup_{1 \leqslant i \leqslant L} A_{i,p}$ $(p = 1, 2, \ldots, r)$. □

According to Theorem 1, the computation of equivalence classes can be performed on each data split independently and the results obtained from parallel computing can be merged into the same equivalence classes as obtained in serial computing. Thus, we can implement the data parallelism of equivalence class computation using MapReduce. Zhang et al. [54] proposed a parallel method for computing lower and upper approximations. For attribute reduction, we only compute the attribute significance among different equivalence classes instead of considering the details of equivalence classes. Based on the parallel computation of equivalence classes, we can further analyze the parallelism of classical attribute reduction algorithms.

3.1.1. The parallelism of discernibility matrix algorithm

As we all know, Skowron and Rauszer [34] proposed a discernibility matrix for representing the set of attributes that can discern object pairs based on the definition of positive region. Hu and Cercone [13] improved the discernibility matrix that can preserve the boundary region partition. When a decision table S is consistent, two discernibility matrices are the same. However, they are different when a decision table S is inconsistent. Misusing a reduct definition that is only for a consistent table will cause a problem for an inconsistent table in the Pawlak rough set model [25]. Most classical attribute reduction algorithms based on the improved discernibility matrix [37,44,49,52] regarded the inconsistent objects in $U - POS_C(D)$ as a special decision equivalence class. Here we denote the decision values of the objects in $U - POS_C(D)$ as a new decision value. Thus, the decision equivalence classes of an inconsistent decision table of k decisions can be re-formed as $\pi_D = \{D_1, D_2, \ldots, D_k, D_{k+1}\}$, and the original inconsistent decision table are transformed into a **'consistent'** decision table. The main idea of these attribute reduction algorithms is to find such an attribute discerning the largest number of object pairs, i.e. the attribute most often occurring in the discernibility matrix. Ngugen and Ngugen [27], Korzeń and Jaroszewicz [17] and Qian et al. [32] implemented efficient heuristics for computing the number of object discerning pairs from the value distribution of attributes. But if the number of the objects exceeds 1 million, all these algorithms cannot load the data into the main memory. Thus, they are infeasible to process the massive data. In order to solve this problem, we present a technique below for counting the number of object discerning pairs in cloud computing [33].

Consider a partition $\pi_D = \{D_1, D_2, \ldots, D_k, D_{k+1}\}$ and $\pi_A = \{A_1, A_2, \ldots, A_r\}$. In A_p, there exists n_p^i objects whose decision value is i. Obviously, all the objects of decision value i in any equivalence class form D_i. In other words, $n_1^i + \cdots + n_r^i$ equals n^i, the number of the objects in D_i. Similarly, we know that $n_1^1 + \cdots + n_r^1 + \cdots + n_1^{k+1} + \cdots + n_r^{k+1}$ equals n, the number of the objects in U. A discernibility object pair is generated from any two objects in which have different decision value and conditional combined values on A. That is, A can discern such a discernibility object pair.

Definition 10. For a decision table S, let $A \subseteq C, \pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, the discernibility object pair set of A with respect to D can be defined as DOP_A^D:

$$DOP_A^D = \{\langle x, y\rangle | x \in D_i, y \in D_j, \exists a \in A, I_a(x) \neq I_a(y)\} \tag{9}$$

where $D_i \in \pi_D, D_j \in \pi_D, 1 \leqslant i < j \leqslant k+1$.

Theorem 2. *Given a decision table S, $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, for $a \in A, DOP_A^D = \bigcup_{a \in A} DOP_{\{a\}}^D$.*

Proof. It is obviously proven according to Definition 10. □

According to Definition 10, we need to generate the set of discernibility object pairs DOP_A^D. When a decision table S is massive, the space complexity of DOP_A^D is $O(n^2)$. In order to handle large data in cloud computing, we below compute the number of the object discerning pairs by the attribute discernibility.

Definition 11. For a decision table S, let $A \subseteq At, \pi_A = \{A_1, A_2, \ldots, A_r\}$, then the attribute discernibility of A can be defined as DIS_A, where

$$DIS_A = \sum_{1 \leqslant p < q \leqslant r} n_p n_q \tag{10}$$

Proposition 1. *For a decision table S, let $A \subseteq C$, $\pi_A = \{A_1, A_2, \ldots, A_r\}$, $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, then the attribute discernibility of A with respect to D $DIS_A^D = \sum_{1 \leqslant i < j \leqslant k+1} \sum_{1 \leqslant p < q \leqslant r} n_p^i n_q^j$.*

Proof. It is obviously proven according to Definition 11. □

However, since equivalence classes are distributed on different nodes in cloud computing, computing DIS_A^D is a complex task according to the above formula. In [33], we propose an efficient method of computing the number of object discerning pairs.

Proposition 2. *For a decision table S, let* $A \subseteq C$ *and* $\pi_A = \{A_1, A_2, \ldots, A_r\}$, $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, $\pi_{A\cup D} = \{A_1^1, A_1^2, \ldots, A_1^{k+1}, A_2^1, A_2^2, \ldots, A_2^{k+1}, \ldots, A_r^1, A_r^2, \ldots, A_r^{k+1}\}, n_{l_1}$ *and* n_{l_2} *denote the number of the objects from any two different equivalence classes in* $\pi_{A\cup D}$, *then* $DIS_A^D = \sum_{1\leqslant p<q\leqslant r} n_p n_q + \sum_{1\leqslant i<j\leqslant k+1} n^i n^j - \sum_{1\leqslant l_1<l_2\leqslant r(k+1)} n_{l_1} n_{l_2}$.

Proof. Since $DIS_{A\cup D} = \sum_{1\leqslant l_1<l_2\leqslant r(k+1)} n_{l_1} n_{l_2}$, $DIS_A = \sum_{1\leqslant p<q\leqslant r} n_p n_q$ and $DIS_D = \sum_{1\leqslant i<j\leqslant r} n^i n^j$, while $DIS_A^D = DIS_A + DIS_D - DIS_{A\cup D}$, thus $DIS_A^D = \sum_{1\leqslant p<q\leqslant r} n_p n_q + \sum_{1\leqslant i<j\leqslant k+1} n^i n^j - \sum_{1\leqslant l_1<l_2\leqslant r(k+1)} n_{l_1} n_{l_2}$. □

Although DIS_A^D can be directly computed using MapReduce, it is required to compute the equivalence classes from A, D and $A \cup D$, thereby a large amount of $\langle key, value\rangle$ pairs are generated and affect the efficiency of the parallel algorithm. Thus we can calculate the number of those indiscernibility object pairs that A cannot discern. An indiscernibility object pair with respect to the conditional attributes A is generated from any two objects which have different decision values and the same combinational values on A.

Definition 12. For a decision table S, let $A \subseteq C, \pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, the indiscernibility object pair set of A with respect to D can be defined as $\widetilde{DOP_A^D}$:

$$\widetilde{DOP_A^D} = \{\langle x, y\rangle | x \in D_i, y \in D_j, \forall a \in A, I_a(x) = I_a(y)\} \tag{11}$$

where $D_i \in \pi_D, D_j \in \pi_D, 1 \leqslant i < j \leqslant k+1$.

Theorem 3. *Given a decision table S,* $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, *for* $a \in A$, $\widetilde{DOP_A^D} = \bigcap_{a\in A} \widetilde{DOP_{\{a\}}^D}$.

Proof. It is obviously proven according to Definition 12. □

Lemma 1. *For a decision table S,* $A \subseteq C, \pi_A = \{A_1, A_2, \ldots, A_r\}, \pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, *then the number of indiscernibility object pair set of A with respect to D* $\widetilde{DIS_A^D} = \sum_{1\leqslant p\leqslant r} \sum_{1\leqslant i<j\leqslant k+1} n_p^i n_p^j$.

Proof. According to Definition 12, it is easily proven. □

According to Definition 12 and Lemma 1, it is found that the indiscernibility object pairs are generated from the boundary region with respect to the conditional attributes A. If A is $\emptyset$, it means that all the objects in the universe are regarded as one equivalence class, i.e. $\widetilde{DIS_\emptyset^D} = \sum_{1\leqslant i<j\leqslant k+1} n^i n^j$. When $\widetilde{DIS_A^D}$ does not equal $\widetilde{DIS_C^D}$, we should add an attribute into A to discern those indiscernibility object pairs. Assume the added attribute is c, the remained indiscernibility object pairs are divided into two parts: some pairs can be discerned by c and the others cannot.

Theorem 4. *Given a decision table S,* $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, *for* $c \in C - A$, *then* $\widetilde{DIS_{A\cup\{c\}}^D} \leqslant \widetilde{DIS_A^D}$.

Proof. Suppose $U/A = \{A_1, A_2, \ldots, A_r\}$, the equivalence classes induced from $U/[A \cup \{c\}]$ are finer than those of U/A. Any equivalence class $A_p (p = 1, 2, \ldots, r)$ can be sub-divided as $A_p^1, A_p^2, \ldots, A_p^{k+1}$ in terms of decision attribute D. The numbers of these equivalence classes are denoted as $n_p^1, n_p^2, \ldots, n_p^{k+1}$, respectively. Suppose attribute c has l different attribute values, we can divide the equivalence class $A_p^i (i = 1, 2, \ldots, k+1)$ into l equivalence classes for candidate attribute set $A \cup \{c\}$. Note that some equivalence classes may be empty.

$$\begin{aligned}
&A_{p,1}^1 \cup A_{p,2}^1 \cup \cdots \cup A_{p,l}^1 = A_p^1\\
&A_{p,1}^2 \cup A_{p,2}^2 \cup \cdots \cup A_{p,l}^2 = A_p^2\\
&\qquad\cdots\\
&A_{p,1}^{k+1} \cup A_{p,2}^{k+1} \cup \cdots \cup A_{p,l}^{k+1} = A_p^{k+1}\\
&n_{p,1}^1 + n_{p,2}^1 + \cdots + n_{p,l}^1 = n_p^1\\
&n_{p,1}^2 + n_{p,2}^2 + \cdots + n_{p,l}^2 = n_p^2\\
&\qquad\cdots\\
&n_{p,1}^{k+1} + n_{p,2}^{k+1} + \cdots + n_{p,l}^{k+1} = n_p^{k+1}
\end{aligned}$$

For an equivalence class A_p, the number of the indiscernibility object pairs $\widetilde{DIS}_A^D$ equals $\sum_{1\leqslant i<j\leqslant k+1} n_p^i n_p^j$. After adding attribute c into A, the number of the indiscernibility object pairs $\widetilde{DIS}_{A\cup\{c\}}^D$ is the sum of $\sum_{1\leqslant i<j\leqslant k+1} n_{p,1}^i n_{p,1}^j, \sum_{1\leqslant i<j\leqslant k+1} n_{p,2}^i n_{p,2}^j, \ldots,$ $\sum_{1\leqslant i<j\leqslant k+1} n_{p,l}^i n_{p,l}^j$, namely $\sum_{1\leqslant i<j\leqslant k+1}\sum_{1\leqslant h\leqslant l} n_{p,h}^i n_{p,h}^j$.

For any equivalence class A_p, suppose any two decision values i and j, we can have

$$n_{p,1}^i n_{p,1}^j + n_{p,2}^i n_{p,2}^j + \cdots + n_{p,l}^i n_{p,l}^j \leqslant \left(n_{p,1}^i + n_{p,2}^i + \cdots + n_{p,l}^i\right)\left(n_{p,1}^j + n_{p,2}^j + \cdots + n_{p,l}^j\right) = n_p^i n_p^j.$$

Therefore, $\widetilde{DIS}_{A\cup\{c\}}^D \leqslant \widetilde{DIS}_A^D$ holds. □

Corollary 1. *Give a decision table S, $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, for $P \subseteq Q \subseteq C$, then $\widetilde{DIS}_Q^D \leqslant \widetilde{DIS}_P^D$.*

Proof. It is obviously proven according to Lemma 1 and Theorem 4. □

Theorem 4 and Corollary 1 indicate that the proposed measure for estimating the number of indiscernibility object pairs is monotonic.

Theorem 5. *Given a decision table S, $\pi_D = \{D_1, D_2, \ldots, D_{k+1}\}$, for $c \in C - A$, $\widetilde{DIS}_{A\cup\{c\}}^D + DIS_{A\cup\{c\}}^D = \sum_{1\leqslant i<j\leqslant k+1} n^i n^j$.*

Proof. It is obviously proven according to Proposition 1 and Lemma 1. □

As discussed above, since $\sum_{1\leqslant i<j\leqslant k+1} n^i n^j = \frac{1}{2}[n^2 - \sum_{1\leqslant h\leqslant k+1}(n^h)^2]$ can be computed in parallel, we can implement the parallelism of constructing the discernibility matrix through parallelizing the computation of equivalence classes as shown in Fig. 1.

Remark 1. Skowron's discernibility matrix and all the improved discernibility matrices consider the difference of an object pair (x, y) from any two objects among two different decision equivalence classes of the positive region. They also consider the discernibility object pairs from such two objects in which x and y come from the positive region and the boundary region respectively. However, they do not consider the differences of an object pair (x, y) from the objects among the boundary

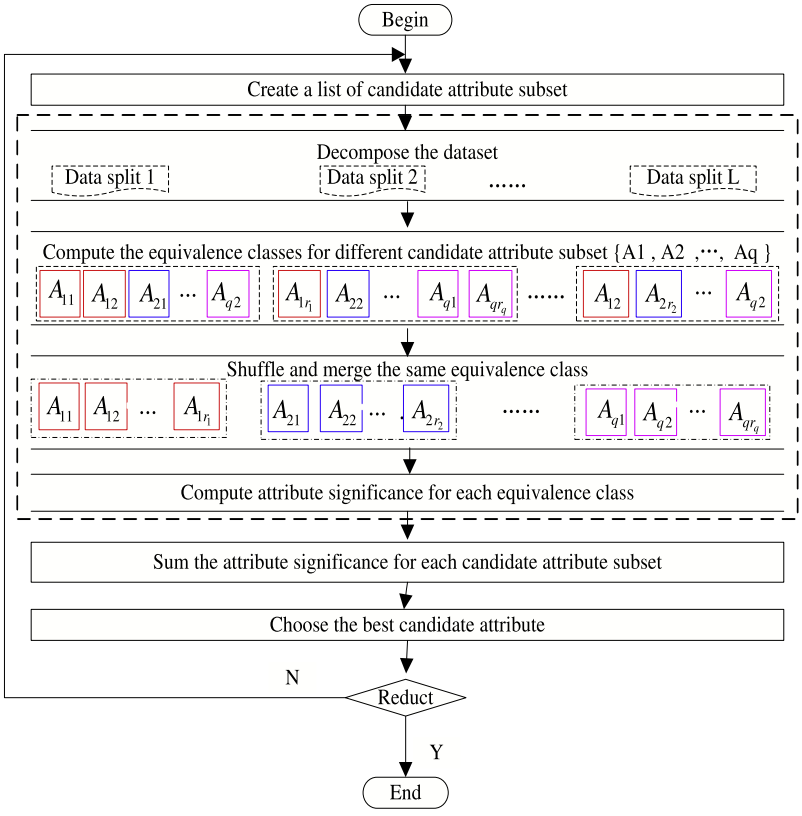

Fig. 1. The parallel and serial computing parts in cloud computing.

region $U - POS_C(D)$. Hu's discernibility matrix only consider the difference of an object pair (x, y) in which x and y come from different decision equivalence classes. Therefore, for Skowron' discernibility matrix, we regard the objects in $U - POS_C(D)$ as a special decision equivalence classes D_{k+1}. According to Definition 10, we do not consider the difference of object pair (x, y) in D_{k+1} as well, because these objects are inconsistent in the original decision table. Since the objects in $U - POS_C(D)$ may be distributed into different nodes in cloud computing, we have to use MapReduce to transform the inconsistent decision tables into the 'consistent' decision tables for Skowron's discernibility matrix algorithm, otherwise we will obtain the reduct of Hu's discernibility matrix algorithm. In fact, our method can be applied to Hu's discernibility matrix (omit **step** 2 in Algorithm 8) since Definition 12 also holds the boundary region partition preservation. When the decision table S is inconsistent, $\widetilde{DIS}_C^D \neq 0$. In this paper, we mainly discuss the parallelism of discernibility matrix algorithm for Skowron's discernibility matrix.

Moreover, the monotonicity of attribute significance measures is required to guide the attribute reduction process. One kind of monotonic measures which can be used to evaluate the discernibility is defined as follows.

Definition 13. For a decision table S, let $A \subseteq C$ and $c \in C - A$, then the significance of attribute c is defined by:

$$sig_3(c, A, D) = \frac{\widetilde{DIS}_A^D - \widetilde{DIS}_{A\cup\{c\}}^D}{\sum_{1\leqslant i<j\leqslant k+1} n^i n^j} \tag{12}$$

3.1.2. The parallelism of positive region and information entropy algorithm

In the Pawlak rough set model, the attribute reduction algorithms based on positive region proceed in the following way. They start with an empty set or the core attributes, and iteratively add attributes one by one into a reduct to maximize the positive regions until the number of objects in the positive region reaches. Referring to this process, some efficient attribute reduction algorithms based on positive region were developed in [30,32,42]. These algorithms are effective for small datasets by employing the relatively reasonable attribute measures or sorting algorithms. It is assumed that all the data can be loaded into the main memory and can be deleted anytime. Unfortunately, they are infeasible for large data because discarding unnecessary objects and restoring the relevant objects consume too much time and space in cloud computing. We therefore always compute the number of the objects in the positive region from different candidate attribute set for the whole dataset in each iteration. Since the computations of the equivalence classes are intensive in a positive region algorithm, we will consider the strategy of computing the equivalence classes and the attribute significance in parallel. The serial computing part consists of only summing the number of the positive region objects and choosing the best candidate attributes for next iteration. As the size of the set of candidate attributes increases, it will generate more $\langle key, value\rangle$ pairs which result in the longer time of the serial computing parts. In the Pawlak rough set model, since $|BND_A(D)| + |POS_A(D)| = |U|$, we turn to compute the number of the objects in the boundary region instead of that in the positive region. Thus we propose a significance measure of attribute c for the boundary region algorithm as follows:

Definition 14. For a decision table S, let $A \subseteq C$ and $c \in C - A$, then the significance of attribute c for boundary region algorithm is defined by:

$$sig_{1'}(c, A, D) = \frac{|BND_A(D)| - |BND_{A\cup\{c\}}(D)|}{|U|} \tag{13}$$

Besides the algorithms based on positive region, the attribute reduction algorithms based on conditional information entropy are also developed in [19,20,24,30,32,38]. However, they are inappropriate to deal with large data as well. Thus, it is required to parallelize the attribute reduction algorithm based on information entropy. The parallelism of information entropy algorithms is similar to that of boundary region algorithm. The intensive computation of these algorithms is to generate a large number of equivalence classes and compute the information entropy for each candidate attribute subset according to the boundary region. Therefore the attribute reduction algorithms based on information entropy is the key to compute equivalence classes in parallel.

As mentioned above, the computations of the equivalence classes in parallel are of critical importance. We find that the whole attribute reduction process can be divided into the parallel and serial computing parts. Fig. 1 shows the process of attribute reduction and the regions surrounded by dash lines denote the parallel computing parts. The equivalence classes and the attribute significance for each equivalence class can be computed in parallel. However, the total attribute significance is calculated and the best candidate attribute subset is determined in serial.

3.2. The relationships among classical attribute reduction algorithms

Among classical attribute reduction algorithms, for any two objects x and y in a decision table, four cases in [32,43] are identified as follows:

1. $I_A(x) = I_A(y), I_d(x) = I_d(y)$;
2. $I_A(x) = I_A(y), I_d(x) \neq I_d(y)$;
3. $I_A(x) \neq I_A(y), I_d(x) \neq I_d(y)$;

4. $I_A(x) \neq I_A(y), I_d(x) = I_d(y)$.

Among the above four cases, since case 4 is implied in case 1 [43], here we only discuss the first three cases. For a decision table, case 1 represents that it is consistent, while case 2 indicates that it is inconsistent. For attribute reduction algorithms, case 1 is used to construct the positive region, case 2 can be used for computing the information gain, the number of the objects in the boundary region and the indiscernibility object pairs, and case 3 is employed to generate a discernibility matrix or the discernibility object pairs. In the Pawlak rough set model, since $|BND_A(D)| + |POS_A(D)| = |U|$, we can transform case 1 into case 2 by computing the boundary region. Moreover, since $DIS_A^D + DIS_A^D = \sum_{1 \leqslant i < j \leqslant k+1} n^i n^j$, we can transform the computations of the number of discernibility object pairs DIS_A^D (case 3) into the computations of the number of indiscernibility object pairs DIS_A^D (case 2). Therefore, considering only case 2 is sufficient for classical attribute reduction algorithms. All the reduct construction methods in terms of case 2 can be induced into attribute reduction algorithms based on boundary region. The relationships among three classical algorithms are illustrated as shown in Fig. 2. In Fig. 2, n_p denotes the number of the inconsistent objects while $\widetilde{n_p}$ represents the number of all consistent objects for each equivalence class. In [48], Yao discussed different attribute measurements on the granularity of partitions. Based on this, we can construct different parallel attribute reduction algorithms.

In what follows, we first discuss the parallelization strategies of attribute reduction, then design the parallel computation of transforming original decision tables into the consistent decision tables and extracting core attributes, finally introduce some details of how to implement parallel attribute reduction algorithms based on positive region (boundary region), discernibility matrix and information entropy using MapReduce.

3.3. Parallelization strategies for attribute reduction

We can categorize these parallel attribute reduction algorithms as task-parallelism and/or data-parallelism algorithms. The differences of the parallelism strategies are shown in Fig. 3, where S denotes the whole dataset, S_i $(i = 1, 2, \ldots, L)$ is a data split, T_j $(j = 1, 2, \ldots, M)$ is a task, "EC" denotes the parallel calculations of the different equivalence classes and the attribute significance, and "AS" denotes the counting parts of the attribute significance for different candidate attribute subsets.

In Fig. 3(a), the algorithms based on task-parallelism perform the task independently but require to access the entire dataset at once. These algorithms are only feasible for small datasets. Classical parallel methods for attribute reduction belong to this category. In Fig. 3(b), parallel algorithms using MapReduce are based on data-parallelism, which partition a massive dataset into multiple data splits and broadcast them to different nodes. Since attribute reduction methods need

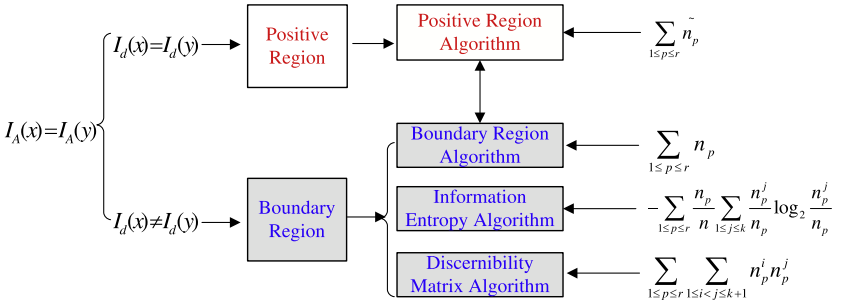

Fig. 2. The relationships among classical attribute reduction algorithms.

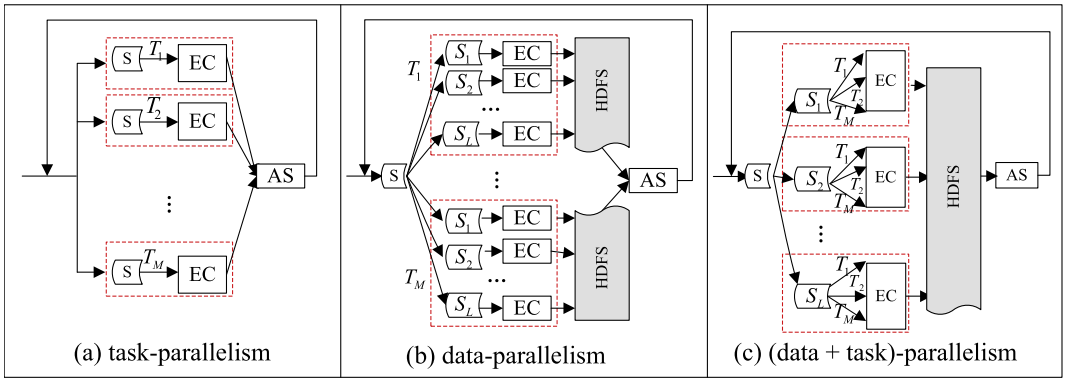

Fig. 3. The main differences of three parallelization strategies.

computing the equivalence classes for each data split in each iteration, we first divide the overall data into multiple splits using MapReduce, then compute the equivalence classes from different candidate attributes in task parallel for each data split as shown in Fig. 3(c).

For massive datasets, parallel attribute reduction algorithms must employ data parallelism strategy with MapReduce. Here we compare two parallel attribute reduction algorithms in Fig. 3(b) and (c). Next we assess the performance of the prototype based on the following assumptions:

- A parallel attribute reduction algorithm consists of both parallel and serial parts.
- The processing capability of each node on per data split is the same constant.
- The communication time is directly proportional to the amount of data transferred across the cluster nodes.

Considering the influence from the workload (S), the number of the tasks (M) and the number of the nodes (N), we evaluate the performance of parallel computing by the following equation:

$$T(S,M,N) = T_{pp}(S,M,N) + T_{sp}(S,M,N) + T_c(S,M,N) \tag{14}$$

where $T(S,M,N)$ represents the execution time across multiple computing nodes. $T_{pp}(S,M,N)$ is the execution time of the processes that can be parallelised. $T_{sp}(S,M,N)$ is the execution time of serial processes. $T_c(S,M,N)$ denotes the communication time between nodes.

For each task, $T_c(S,M,N)$ can be presented by the following equation:

$$T_c(S,M,N) = (N-1) * \left(t * \frac{L}{N} + t'\right) \tag{15}$$

In $T_c(S,M,N)$, t is the communication time of transferring a data split between nodes, L is the number of data splits for S, $\frac{L}{N}$ is the number of data splits for each node when N computing nodes are running tasks concurrently, and t' is the communication time required to establish a session between nodes which is independent of the workload S. All these parameters are constants. We can find that two parallel attribute reduction algorithms in Fig. 3(b) and (c) may have the same parallel and serial execution time but different communication time.

Suppose the size of a reduct is R. We denote the communication time in Fig. 3(b) and (c) as $T_c^D(S,M,N)$ and $T_c^{DT}(S,M,N)$ respectively. For the attribute reduction in Fig. 3(b), we must execute R iterations. The first iteration computes M attribute significances in serial, this means it must finish M tasks. The second iteration executes $M-1$ tasks, and so on. For the attribute reduction in Fig. 3(c), we execute R iterations as well. However, in the first iteration M attribute significances are computed in parallel. In other words, M tasks are processed concurrently, and the communication time is $(N-1) * \left[M * \left(t * \frac{L}{N}\right) + t'\right]$. The second iteration executes $M-1$ tasks, and the communication time is $(N-1) * \left[(M-1) * \left(t * \frac{L}{N}\right) + t'\right]$. Correspondingly, we can get the communication time of the R-th iteration, $(N-1) * \left[\left(t * \frac{L}{N}\right) + t'\right]$. Thus we can obtain the following equations:

$$T_c^D(S,M,N) = \sum_{1 \leqslant i \leqslant R} \left[(M-i+1) * (N-1) * \left(t * \frac{L}{N} + t'\right)\right] \tag{16}$$

$$T_c^{DT}(S,M,N) = (N-1) * \left(\sum_{1 \leqslant i \leqslant R} \left[(M-i+1) * \left(t * \frac{L}{N}\right)\right] + R * t'\right) \tag{17}$$

According to Eqs. (16) and (17), we can get the following equation.

$$T_c^D(S,M,N) = T_c^{DT}(S,M,N) + (N-1) * \sum_{1 \leqslant i \leqslant R} [(M-i) * t'] \tag{18}$$

Since the parallel attribute reduction algorithms using only data parallelism need to consume more time for job scheduler and data communications, we focus on the parallel attribute reduction algorithms with both data and task parallelism.

3.4. Parallel computation of the 'consistent' decision table for Skowron's discernibility matrix

As discussed above, the classical attribute reduction algorithms based on discernibility matrix always considered the 'consistent' decision table. If the decision table is inconsistent, those algorithms regard the objects in $U - POS_C(D)$ as a special equivalence class, D_{k+1}, whose decision value is $k+1$. Therefore, we must transform an inconsistent decision table into a 'consistent' decision table to construct discernibility matrix. In this process, the most intensive calculation is to compute the equivalence classes. It is well known that the computations of the equivalence classes are independent of each other. Thus, the computations of different equivalence classes from a subset of attributes can be executed in parallel.

The massive data is stored in Distributed File System [11] in the form of a sequence file of $\langle key, value\rangle$ pairs, which represent the objects in the input dataset. In each pair, the key denotes an object and the value is null. The MapReduce framework partitions the whole dataset into many data splits, and globally broadcasts them to all mappers. For each map task, map function receives a candidate subset of attributes as a global parameter, computes the equivalence classes for each object from a data split, and writes them into local files on the corresponding node. These files consist of two parts:

$\langle key', value' \rangle$, where key' is the combinational value of an equivalence class, and $value'$ is the decision value of the corresponding equivalence class. We here design a Parallel Algorithm for the Computation of a 'Consistent' Decision Table (PACCDT) using MapReduce. The pseudocodes of **Algorithm PACCDT** which consist of **Algorithm PACCDT-Map** and **Algorithm PACCDT-Reduce**, are shown in Algorithms 1 and 2 respectively.

Algorithm 1. PACCDT-Map(key, value)

Input: Conditional attributes, C; a data split, S_i
Output: $\langle key', value' \rangle$, where key' is an equivalence class and $value'$ is a decision value
1: **for** each object x in S_i **do**
2: $key' \leftarrow \vec{x}_C$; $//\vec{x}_C$ is an equivalence class induced from C
3: $value' \leftarrow I_d(x)$; $//I_d(x)$ is a decision value
4: EmitIntermediate $\langle key', value' \rangle$;
5: **end for**

Algorithm 2. PACCDT-Reduce(key, V)

Input: The same equivalence class, key; the list of the decision values, V
Output: $\langle key', value' \rangle$, where key' is an equivalence class and $value'$ is a decision value
1: num $\leftarrow$ the number of the different values in V;
2: **if** num == 1 **then**
3: $value' \leftarrow$ the decision value in V;
4: **else**
5: $value' \leftarrow$ k + 1;
6: **end if**
7: **for** each $v \in V$ **do**
8: Emit $\langle key', value' \rangle$
9: **end for**

By Algorithm 1, we compute the equivalence classes induced from the whole conditional attributes for each data split and store them into the local files. The MapReduce framework copies the same equivalence classes from different Mapper nodes to the corresponding Reducer nodes and sorts these equivalence classes. By Algorithm 2, we can check the consistency of decision values among the same equivalence classes. If all objects have the same decision value, we output all objects, otherwise we denote the decision value of all objects by $k+1$ and output them. That is, all the objects with the decision value $k+1$ form the boundary region in the original decision table, which can be regarded as a special equivalence class. Such a decision table is a 'consistent' decision table which preserves the indiscernibility object pairs in the original decision table.

To handle the inconsistency, a parallel algorithm PACCDT for acquiring the 'consistent' decision table (S') is proposed using MapReduce. This parallel algorithm is designed based on the MapReduce driver, which executes in the form of map and reduce functions. Algorithm PACCDT is illustrated in Algorithm 3.

Algorithm 3. Algorithm PACCDT

Input: A decision table, S; the whole conditional attributes, C
Output: A 'consistent' decision table, S'
1: Set the number of the reduce tasks;
2: Initiate a MapReduce job, compute the equivalence classes by executing Algorithm 1 and acquire the consistent decision table by Algorithm 2.

By Algorithm 3, we can acquire a 'consistent' decision table. Note that if we set few reduce tasks, we will obtain large consistent datasets. However, the time cost in Reduce phase will be very high. Therefore, we recommend setting the number of the reduce tasks as the number of the cluster nodes in this MapReduce job configuration. Under this strategy, a large dataset will be divided into multiple consistent decision sub-tables.

In the next subsections, we implement a parallel algorithm of computing the core attributes, and parallelize three classical attribute reduction algorithms using MapReduce.

3.5. Parallel algorithm of computing the core attributes

Many researchers try to improve the efficiency of attribute reduction algorithms through studying the core attributes [32,44]. The core attributes are helpful to reduce search space and the size of discernibility matrix. Thus the fast algorithm

for extracting the core attributes is desirable. To the best of our knowledge, the time complexity of the fastest algorithm in computing core attributes is $O(|C||U|)$, which is infeasible for massive data. It is found that the computation of equivalence classes is intensive in computing the core attributes. Therefore, we design a Parallel Algorithm of Computing the Core Attributes. **Algorithm PACCA** includes **Algorithm PACCA-Map** and **Algorithm PACCA-Reduce**. The pseudocodes of the algorithms are shown in Algorithms 4 and 5 respectively.

Algorithm 4. PACCA-Map(key, value)

Input: Conditional attributes, C; any candidate attribute, $c \in C$; a data split, S_i
Output: $\langle key', value' \rangle$, where key' is an equivalence class with attribute c and $value'$ is a decision value
1: **for** each object x in S_i **do**
2: **for** each attribute $c \in C$ **do**
3: $key' \leftarrow \mathrm{c} + \vec{x}_{C-\{c\}}$;
4: //Here, c is flag, which means the equivalence class $\vec{x}_{C-\{c\}}$.
5: $value' \leftarrow I_d(x)$;
6: EmitIntermediate $\langle key', value' \rangle$;
7: **end for**
8: **end for**

Algorithm 5. PACCA-Reduce(key,V)

Input: An equivalence class with attribute c, key; the list of the decision values, V
Output: $\langle key', value' \rangle$, where key' is a candidate attribute c and $value'$ is the attribute significance of the corresponding attribute, $c\_AttriSig$
1: $c\_AttriSig \leftarrow 0$;
2: **for** each $j \in V$ **do**
3: $n_p^j \leftarrow$ the frequency of decision value equal j in $[C - \{c\}]_p (1 \leqslant j \leqslant k)$;
4: **end for**
5: $c\_AttriSig \leftarrow \sum_{1 \leqslant j \leqslant k} n_p^j$;
6: **if** all the decision values are different **then**
7: $key' \leftarrow$ c; // c is flag from the equivalence class key
8: $value' \leftarrow c\_AttriSig$;
9: Emit $\langle key', value' \rangle$;
10: **end if**

By Algorithm 4, we can compute the different equivalence classes induced from different candidate attribute sets. By Algorithm 5, we can compute the single attribute significance in the equivalence classes $[C - \{c\}]_p$ in parallel for boundary region algorithm. For information entropy and discernibility matrix algorithm, the computations of $c\_AttriSig$ in Algorithm 5 (line 5) are modified as "5: $c\_AttriSig \leftarrow -\frac{n_p}{n} \sum_{j=1}^{k} \frac{n_p^j}{n_p} \log_2 \frac{n_p^j}{n_p}$" and "5: $c\_AttriSig \leftarrow \sum_{1 \leqslant j_1 < j_2 \leqslant k+1} n_p^{j_1} n_p^{j_2}$" respectively.

Algorithm PACCA requires a MapReduce job as well. The attribute significance values of different candidate attribute subsets are read from the output files. And then the significance values of the candidate attributes are summed up. Algorithm PACCA is summarized in Algorithm 6.

Algorithm 6. PACCA

Input: A decision table, S; the whole conditional attributes, C
Output: Core attributes, Core(C)
1: Core(C) $\leftarrow \emptyset$, AttriSigValue[ ] $\leftarrow 0$;
2: Initate a MapReduce job, compute equivalence classes induced from candidate attribute set $C - \{c_i\}$ (where $i = 1, 2, \ldots, m$) by executing Algorithm 4, and calculate each attribute significance by Algorithm 5;
3: Read the attribute significance from the HDFS, add up $c\_AttrSig$ into the corresponding element in array AttriSigValue[ ];
4: **for** $i = 1$ to m **do**
5: **if** AttriSigValue[i] $\neq |U - POS_C(D)|$ **then**
6: Core(C) = Core(C) $\cup \{c_i\}$;
7: **end if**
8: **end for**
9: **return** Core(C);

By Algorithm 6, we can acquire the core attributes for positive region (boundary region) algorithm. For the core attributes in information entropy and discernibility matrix algorithm, the computations of judging attribute c in Algorithm 6 (line 5) are modified as "5: *ifAttriSigValue*$[i] \neq Info(D|C)$" and "5: *ifAttriSigValue*$[i] \neq DIS_C^D$" respectively.

3.6. Parallel algorithms for attribute reduction using MapReduce

As discussed above, three classical algorithms compute the equivalence classes of different candidate attribute sets and the corresponding attribute significance iteratively. Thus we can design a general Parallel Algorithm for Attribute Reduction (PAAR). The pseudocode of **Algorithm PAAR** includes **Algorithm PAAR-Map** for computing the equivalence classes and **Algorithm PAAR-Reduce** for computing the attribute significance. Algorithm PAAR-Map is shown in Algorithm 7 and Algorithm PAAR-Reduce is the same as Algorithm 5.

Algorithm 7. PAAR-Map(key, value)

Input: Selected attributes, A; any candidate attribute, c ($c \in C - A$); a data split, S_i
Output: $\langle key', value' \rangle$, where key' is an equivalence class with attribute c and $value'$ is a decision value
1: **for** each object x in S_i **do**
2: **for** each attribute $c \in C - A$ **do**
3: $key' \leftarrow \mathrm{c} + \vec{x}_{A\cup\{c\}}$;
4: //Here, c is flag which means the equivalence class $\vec{x}_{A\cup\{c\}}$
5: $value' \leftarrow I_d(x)$;
6: EmitIntermediate $\langle key', value' \rangle$;
7: **end for**
8: **end for**

By Algorithm 7, we can compute the equivalence classes of different candidate attribute subsets in data and task parallel on each data split.

For boundary region algorithm, when computing $sig_{1'}(c, A, D)$ for $c \in C - A$, we only calculate the number of boundary region objects, $|BND_{A\cup\{c\}}(D)|$, in reduce phase. For information entropy and discernibility matrix algorithm, when computing $sig_2(c, A, D)$ and $sig_3(c, A, D)$ for $c \in C - A$, it is found that the values of $Info(A, D)$ and DIS_A^D keep constant in iteration, thus we calculate only the information entropy $Info(A \cup \{c\}, D)$ and the number of indiscernibility object pairs $DIS_{A\cup\{c\}}^D$ in reduce phase respectively.

Parallel attribute reduction algorithms based on boundary region (positive region), discernibility matrix and information entropy require MapReduce job to read the attribute significance values of different candidate attribute subsets from the output files in reduce phase. These parallel algorithms sum up the values of candidate attributes, and determine which attributes to be added into a reduct. In this way, we renew the candidate attributes which are used for the next iteration. This procedure must be executed serially in each iteration. Algorithm PAAR can be summarized as follows.

Algorithm 8. A general parallel algorithm for attribute reduction using MapReduce

Input: A decision table, S
Output: A reduct, *Red*
1: $Red \leftarrow \emptyset$, Core(C) $\leftarrow \emptyset$;
2: Compute a consistent decision table by executing Algorithm 3;
3: Compute the core attributes, Core(C), by Algorithm 6;
4: $Red \leftarrow$ Core(C);
5: **while** $CC_\triangle(Red, D) \neq CC_\triangle(C, D)$ **do**
6: //Here $CC_\triangle(\cdot,\cdot)$ is the classification capability
7: Initate a MapReduce job and execute Algorithms 7 and 5;
8: **for** each attribute $c \in C - Red$ **do**
9: Compute $sig_\triangle(c, Red, D)(\triangle = \{1, 1', 2, 3\})$;
10: **end for**
11: $sig_\triangle(c', Red, D) \leftarrow$ best($sig_\triangle(c, Red, D)$) (if the attribute like that is not only one, select one attribute arbitrarily);
12: $Red \leftarrow Red \cup \{c'\}$;
13: **end while**
14: Red.

By Algorithm 8, we can acquire a 'consistent' decision table, core attributes and one reduct. For positive region and information entropy algorithm, we omit **step 2**. Since there has no core attributes in some decision tables, we can omit **step 3** to improve the reduction performance. We denote the general Parallel Algorithms for Attribute Reduction based on Positive Region, Boundary Region, Discernibility Matrix and Information Entropy in data and task parallel as **PAAR-PR**, **PAAR-BR**, **PAAR-DM** and **PAAR-IE**, respectively. We only acquire one reduct by a parallel attribute reduction algorithm. In general, these reducts are not all the same since the attribute significance measurements are different. In other words, the inequality $sig_1(c_i, Red, D) \geqslant sig_1(c_j, Red, D)$ for positive region algorithm does not hold that $sig_2(c_i, Red, D) \geqslant sig_2(c_j, Red, D)$ and $sig_3(c_i, Red, D) \geqslant sig_3(c_j, Red, D)$ for the other two algorithms. This will lead to different selected candidate attributes and generate different reducts (see the reducts in Table 2).

Theorem 6. *The reducts obtained by the parallel attribute reduction algorithms are the same as those obtained by the corresponding serial methods.*

Proof. As indicated in [4], there are four basic steps in a typical attribute reduction method—subset generation, subset evaluation, stopping criterion and result validation. The only difference between the parallel algorithms and the corresponding serial methods is the subset evaluation procedure (see Fig. 1). The subset evaluation process mainly includes the computations of the equivalence classes and the attribute significance.

For a decision table $S, A \subseteq C, \pi_A = \{A_1, A_2, \ldots, A_r\}$ for the serial methods. For the parallel algorithms, suppose S_i $(i = 1, 2, \ldots, L)$ be a data split of $S, S_i/A = \{A_{i,1}, A_{i,2}, \ldots, A_{i,r}\}$. $sig_\triangle(c, A, D)$ and $sig_\triangle^{DT}(c, A, D)$ denote the attribute significances of attribute c in the serial methods and in the corresponding parallel algorithms respectively.

(1) According to Theorem 1, $S = \bigcup_{1 \leqslant i \leqslant L} S_i, A_p = \bigcup_{1 \leqslant i \leqslant L} A_{i,p}$ $(p = 1, 2, \ldots, r)$. This means that the equivalence classes in the serial methods are the same as those in the corresponding parallel algorithms.
(2) According to Definitions 4, 7, 13 and 14, $sig_\triangle(c, A, D) = sig_\triangle^{DT}(c, A, D)(\triangle = \{1, 1', 2, 3\})$ because the equivalence classes are the same for the serial methods and the corresponding parallel algorithms.
Therefore, the reducts obtained by the parallel algorithms are the same as those obtained by the corresponding serial methods. □

4. Experimental evaluation

This section presents the experimental results of a parallel algorithm for computing the consistent decision table and core attributes, as well as our parallel attribute reduction algorithms in data and task parallel. We primarily focus on the performances such as speedup, scaleup and sizeup [41] of the parallel algorithms for attribute reduction. We do not consider the relative accuracy of these algorithms since the parallel algorithms produce the same results as those in serial, and different attribute reduction algorithms will generate different reducts in general (see Ref. [32] and Table 2).

4.1. Experiment setup

We run parallel algorithms on a cluster of 17 nodes. For distributed experiments, one is set as a master node and the rest are configured as slave nodes. Each node has 2 GB of main memory and uses Intel Pentium Processor No. E5300 with Dual-Core(2 cores in all, each has a clock frequency of 2.6 GHz), and connects via an Ethernet (100 Mbit/s). For each node, we install Cygwin 2.697(a Linux-like environment in Windows), Hadoop 0.20.2 [11] and Java 1.6.20. We make the following changes to the default Hadoop configuration: we run two map and one reduce tasks in parallel on each node, and set the replication factor to 1.

We conduct an extensive series of experiments on two commonly used machine learning data sets, i.e. Mushroom and Gisette from the UCI Machine Learning repository [7] and four synthetic large data (DS3-6). We discretize Mushroom and Gisette, and duplicate Mushroom 5000 times and Gisette 17 times to generate two large datasets DS1 and DS2 for the validity and correctness of our parallel algorithms. Each dataset has only one decision attribute. The attribute values of each

Table 1
Description of the datasets.

No.	Datasets	Objects	Attributes	Classes	Consistency
1	DS1	40,620,000	22	2	Yes
2	DS2	102,000	5000	2	Yes
3	DS3	20,000,000	30	10	No
4	DS4	40,000,000	50	10	Yes
5	DS5	100,000	10,000	10	Yes
6	DS6	1000,000	2000	10	Yes

synthetic dataset are random integers from 1 to 10. When the size of the conditional attribute set is large, the dataset is consistent in general. Thus, we artificially generate the inconsistent objects and add these objects into DS3. Table 1 summarizes the characteristics of each dataset.

4.2. The running time on different datasets

For large data, we partition it into many data splits in data parallel using MapReduce, and deal with each data split in task parallel. Fig. 4 shows the running time of some important selected attributes in each iteration for six datasets on 16 nodes. Note that the first value in each subgraph denotes the running time of computing the positive regions, boundary regions and information entropy on the original dataset and of computing the 'consistent' dataset for discernibility matrix algorithm.

From Fig. 4, one can check that the three boundary region algorithms exhibit a similar pattern of increase in the running time especially for DS2, DS5 and DS6 with high dimensions, whereas the positive region algorithm consumes more time. In Fig. 4(g), the total running time of PAAR-PR is 4–8 times as much as that of PAAR-BR, PAAR-DM and PAAR-IE on DS2, DS5 and DS6, because PAAR-PR needs to compute a large number of positive region objects when the number of selected attributes increases. In general, PAAR-DM consumes more time than PAAR-IE and PAAR-BR for the same length of the reduct on DS2, DS4 and DS6, since PAAR-DM uses BigInteger package in JAVA program language for the series of operations on large integer. Different reducts for four attribute reduction algorithms are listed in Table 2.

4.3. The performance evaluations on different datasets

In the following, we examine the speedup, scaleup and sizeup characteristics of our parallel algorithms in data and task parallel.

4.3.1. Speedup

In order to measure the speedup, we keep the dataset constant but increase the number of computers in the system. The perfect parallel algorithm demonstrates a linear speedup: a system with m times the number of computers yields a speedup

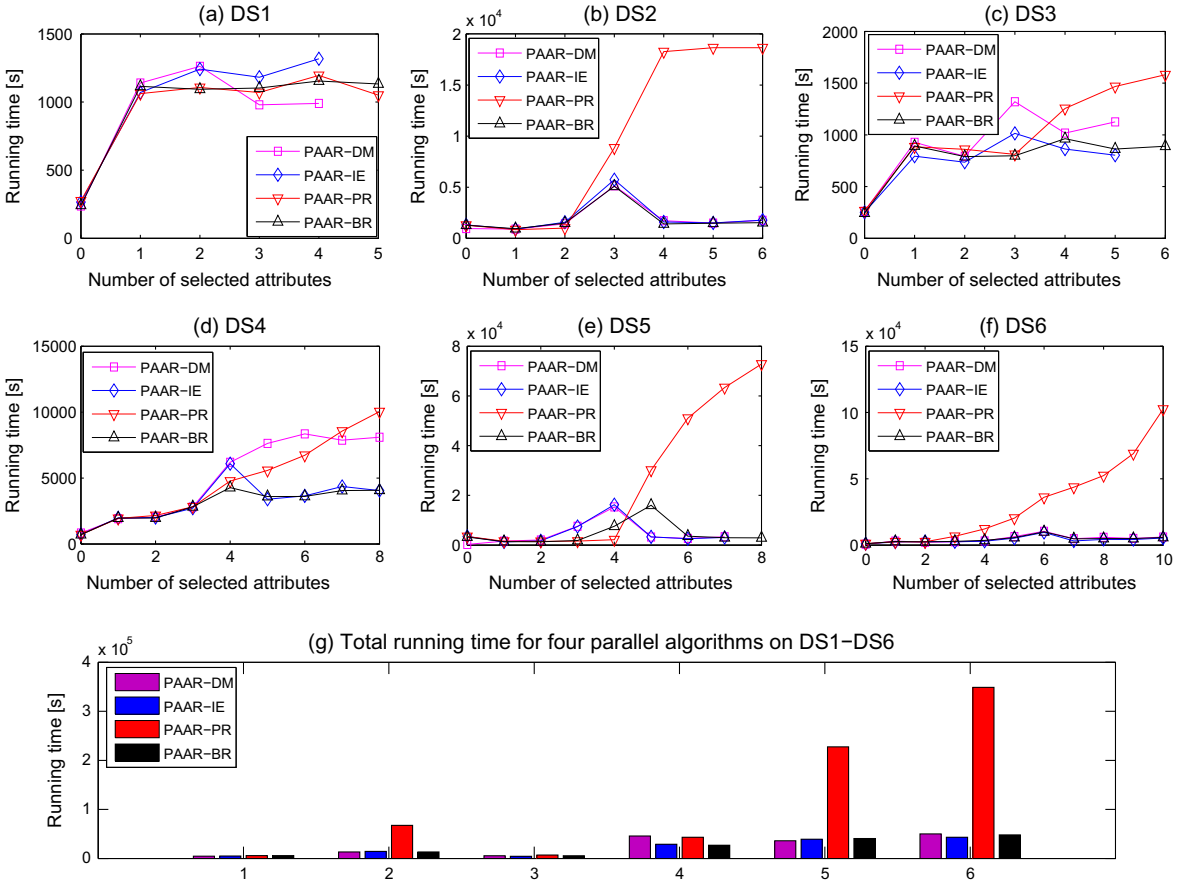

Fig. 4. The comparisons of four algorithms for six datasets on 16 nodes.

Table 2
Different reducts of the datasets.

No.	PAAR-BR (PAAR-PR)	PAAR-DM	PAAR-IE
1	{5,20,8,12,3}	{5,20,22,21}	{5,20,22,21}
2	{1,2,2530,3601,222,3}	{1804,2280,3192,1484,347,1}	{395,905,4991,2210,156,1}
3	{1,2,3,28,7,4}	{3,25,17,6,9}	{25,18,29,3,6}
4	{1,2,3,14,15,43,42,4}	{21,46,42,4,7,26,11,1}	{30,6,1,9,5,20,3,2}
5	{1,2,3,3130,9178,7282,1775,10}	{3335,1925,361,6778, 1241,8923,17}	{3335,4939,2073,9396,9058,7929,2}
6	{1,2, 3,1562,1678,1643,1134,1927,1679,8,4}	{1817,1656,1334,1377,1966,1157,1322,1364,1194,805,3}	{1689,1145,1467,1345, 672,1256,1896,357, 968,1846,1}

of m. However, linear speedup is difficult to achieve because of the serial computing, the communication costs, the faults and the overheads in job scheduling, monitoring and control. We evaluate the speedup on datasets with different nodes. The number of nodes varied from 1 to 16. Fig. 5 shows the speedup of parallel computations of the consistent decision table and the relative speedup of a parallel algorithm for the core attributes for DS1-6. From Fig. 5(a), the speedup is lower than what we expected, since the algorithm only generates many $\langle key, value\rangle$ pairs in the map phase and the reduce function plays little role for parallel computing, namely the proportion of the parallel computing time to the serial computing time is lower. From Fig. 5(b), the relative speedup of computing the core attributes in parallel is much better for DS1, DS3 and DS4.

Fig. 6 shows the speedup of four parallel attribute reduction algorithms for DS1-6. From Fig. 6, PAAR-PR, PAAR-BR, PAAR-DM and PAAR-IE can process large-scale datasets efficiently. However, one can check that the speedup is lower on DS2, DS5 and DS6 with high dimensions. The reason is that the serial computing time is overwhelming in the total time. We improve computing the total attribute significance for each candidate attribute in data parallel rather than in serial. Thus, the total running time is reduced greatly. Fig. 7 only illustrates the ratio and the speedup of two algorithms PAAR-DM and PAAR-DM-D (in data parallel once more) for computing the total attribute significance.

4.3.2. Scaleup

Scaleup is defined as the ability of a m-times larger cluster to perform a m-times larger job in the same run-time as the original system. To demonstrate how well the three parallel algorithms handle larger datasets when more slave nodes are available, we have conducted the scaleup experiments where we make the size of the datasets grow in proportion to the number of slave nodes in the cluster nodes. For the dataset DS3, 20,000,000 objects are calculated on 4 node and 40,000,000 and 80,000,000 objects (2 and 4 times DS3) are handled on 8 and 16 nodes respectively. Fig. 8 shows the scaleup performance results of the datasets. The higher the scaleup value, the better the performance is. In Fig. 8, the values of scaleup are all higher than 0.6, which indicate our proposed algorithms scale well.

4.3.3. Sizeup

Sizeup mainly measures how much longer it takes on a given system, when the dataset size is m-times larger than the original data set. For the measure of sizeup, we fix the number of nodes to 4, 8 and 16 respectively, and increase the size of dataset from 1 to 4 GB. Fig. 9 only shows the sizeup performance results of dataset DS4 on 4, 8 and 16 nodes. The higher value of the sizeup indicates the longer time the system takes when the size of a dataset increases. Therefore our proposed algorithms have a good sizeup performance.

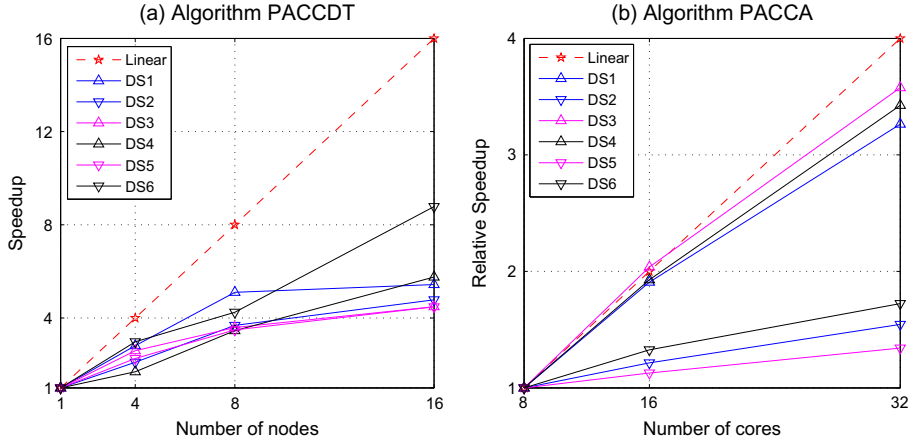

Fig. 5. The speedup of parallel computing the consistent decision table and core attributes.

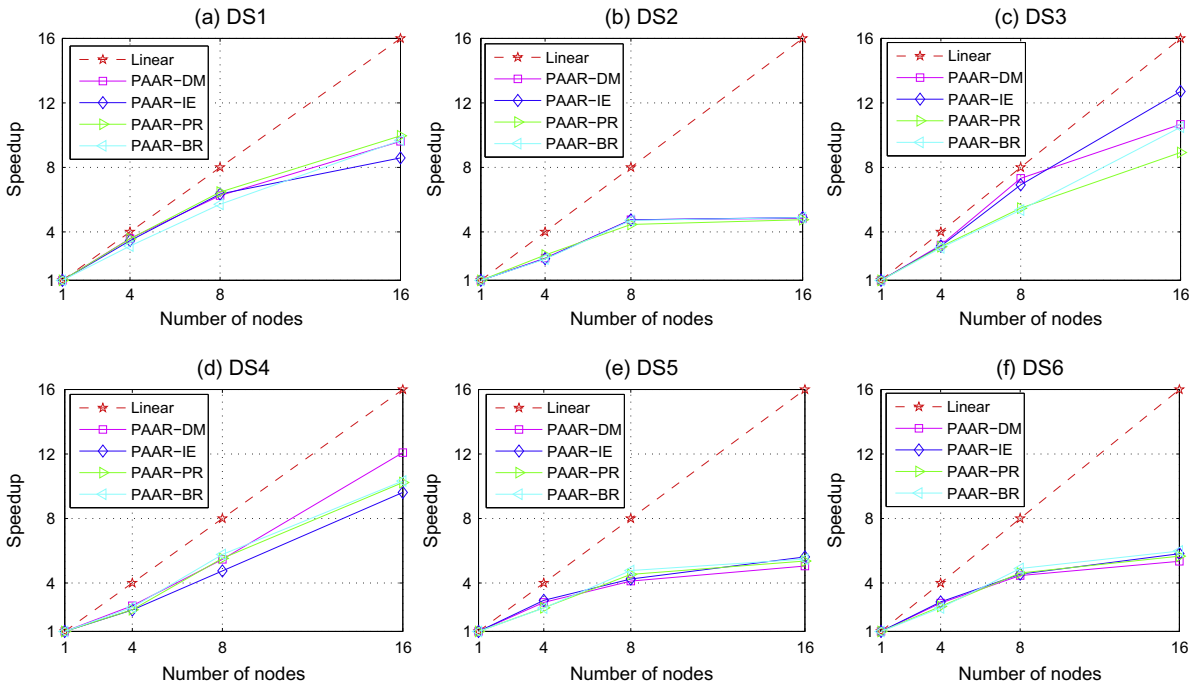

Fig. 6. The speedup of different parallel algorithms.

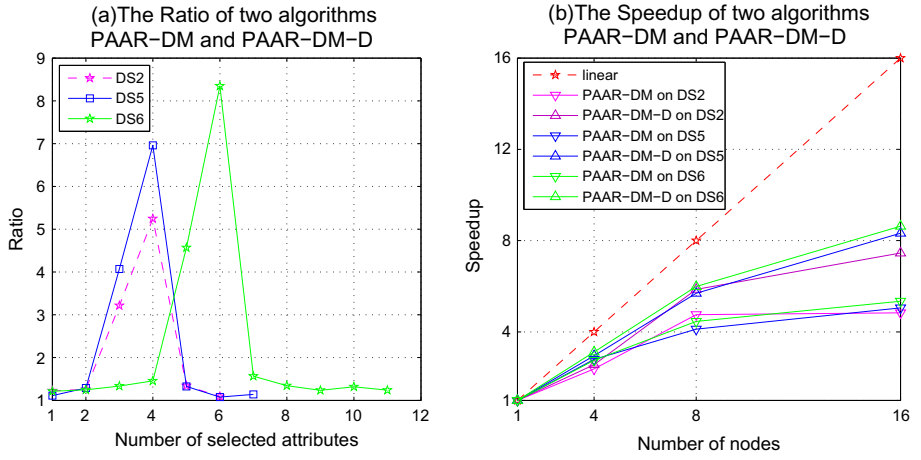

Fig. 7. The ratio and the speedup of PAAR-DM and PAAR-DM-D.

4.4. Discussion

Yang et al. [45] first computed the reduct red_i from sub-decision table S_i (data split), then judged whether new attributes AttrSet should be added in $\cup red_i$, and finally acquired the reduct Red by deleting redundant attributes using MapReduce. Since the sub-decision tables did not exchange information, this Red may be **approximate**. Moreover, the set of the candidate reducts may be large which lead to a serious problem in deleting redundant attributes. Qian et al. [33] proposed the approach to attribute reduction based on discernibility matrix by computing the number of discernibility object pairs. In this paper, we further discuss and implement the parallelism of three classical attribute reduction algorithms. Our parallel algorithms first decomposed a large decision table S into many data splits in data parallel using MapReduce, then computed the equivalence classes on each data split in task parallel, thereby acquired the consistent decision table, the core attributes and a reduct. Since all equivalence classes are shuffled and sorted, the reducts by our parallel algorithms PAAR-PR, PAAR-BR, PAAR-DM and PAAR-IE are **exact**.

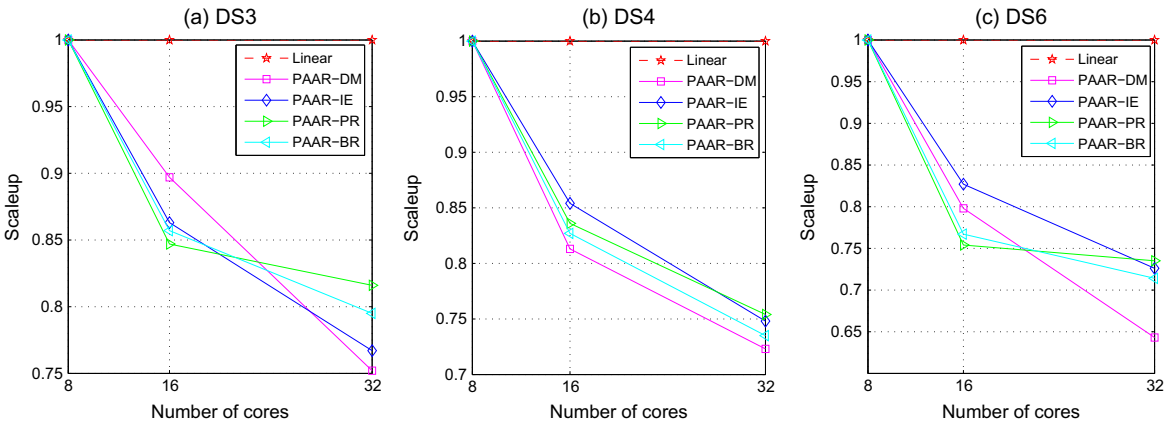

Fig. 8. The scaleup of different parallel algorithms.

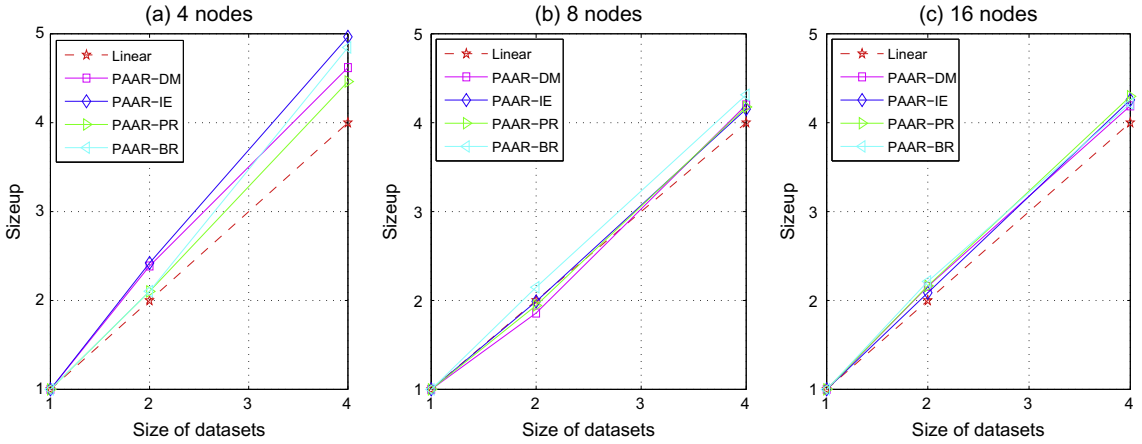

Fig. 9. The sizeup of different parallel algorithms.

Results on six datasets relevant to data parallelism and task parallelism are shown in Figs. 4 and 6 respectively. The principal details in these experimental observations are illustrated as follows.

- **The more the number of $\langle key, value\rangle$ pairs, the longer the running time.**
  As illustrated in Fig. 4, the running time of algorithm PAAR-PR is longer than that of PAAR-BR since PAAR-PR generates more $\langle key, value\rangle$ pairs as the size of the candidate attribute set increases.
- **The larger the number of $\langle key, value\rangle$ pairs, the more the times of data parallel.**
  Using task parallel, parallel attribute reduction algorithms generate more $\langle key, value\rangle$ pairs for some datasets with high dimensions, which result in the lower speedup. The reason is that the serial computing time dominates in the total time. In fact, all the $\langle key, value\rangle$ pairs forms a large dataset as well. We should compute the attribute significance in data parallel once more instead of in serial computing. This strategy can effectively improve the speedup as indicated in Fig. 7.

5. Conclusions

Classical attribute reduction algorithms cannot deal with large data. To solve the problem, we analyzed the parallelism of classical attribute reduction algorithms and illustrated parallel/serial computing parts. By designing the proper $\langle key, value\rangle$ pairs, we implemented the map and reduce functions for the equivalence classes and attribute significance, thereby proposed the parallel attribute reduction algorithms for large data using MapReduce. Experimental results showed that the proposed parallel algorithms can deal with massive data in cloud computing.

Furthermore, the parallelization of other attribute reduction algorithms combined with genetic algorithm and ant colony optimization, as well as the extended rough set models will be further studied.

Acknowledgments

The authors would like to thank both the anonymous referees and the editor for their valuable comments and suggestions. This work was partially supported by the National Natural Science Foundation of China under Grant Nos: 61075056, 61273304, 61103067, the Key Laboratory of Cloud Computing and Intelligent Information Processing of Changzhou City under Grant No: CM20123004, the Key Laboratory of Embedded System and Service Computing, Ministry of Education under Grant No: ESSCKF201303, the Natural Science Foundation and Doctoral Research Foundation of Jiangsu University of Technology under Grant Nos: kyy12018, kyy13003.

References

[1] A. Verma, X. Llora, D.E. Goldberg, R.H. Campbell, Scaling genetic algorithms using MapReduce, in: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, IEEE Computer Society, 2009, pp. 13–18.
[2] G.S. Almasi, A. Gottlieb, Highly Parallel Computing, Benjamin-Cummings publishers, Redwood City, CA, 1989.
[3] C.T. Chu, S. Kim, Y.A. Lin, et al, MapReduce for machine learning on multicore, in: Proceedings of the 20th Conference on Advances in Neural Information Processing Systems, NIPS, 2006, pp. 281–288.
[4] M. Dash, H. Liu, Consistency-based search in feature selection, Artif. Intell. 151 (2003) 155–176.
[5] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–114.
[6] D.Y. Deng, D.X. Yan, J.Y. Wang, Parallel reducts based on attribute significance, in: J. Yu, S. Greco, P. Lingras, et al. (Eds.), Rough Set and Knowledge Technology, Lecture Notes in Computer Science, vol. 6401, Springer, Berlin/Heidelberg, 2010, pp. 336–343.
[7] A. Frank, A. Asuncion, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2010. <http://archive.ics.uci.edu/ml>.
[8] S. Ghemawat, H. Gobioff, S.T. Leung, The Google file system, SIGOPS – Oper. Syst. Rev. 37 (5) (2003) 29–43.
[9] I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains, Pattern Recognit. 43 (2010) 5–13.
[10] I. Guyan, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.
[11] Apache Hadoop. <http://lucene.apache.org/hadoop/>.
[12] L.X. Han, C.S. Liew, J.V. Hemert, M. Atkinson, A generic parallel processing model for facilitating data mining and integration, Parallel Comput. 37 (2011) 157–171.
[13] X.H. Hu, N. Cercone, Learning in relational databases: a rough set approach, Comput. Intell. 11 (1995) 323–338.
[14] Q.H. Hu, Z.X. Xie, D.R. Yu, Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation, Pattern Recognit. 40 (2007) 3509–3521.
[15] Q.H. Hu, D.R. Yu, J.F. Liu, C.X. Wu, Neighborhood rough set based heterogeneous feature subset selection, Inform. Sci. 178 (2008) 3577–3594.
[16] Q.H. Hu, W. Pedrycz, D.R. Yu, J. Lang, Selecting discrete and continuous features based on neighborhood decision error minimization, IEEE Trans. Syst. Man Cybernet. – Part B: Cybernet. 40 (1) (2010) 137–150.
[17] M. Korzeń, S. Jaroszewicz, Finding reducts without building the discernibility matrix, in: Proceedings of 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), IEEE Computer Society, 2005, pp. 450–455.
[18] J.Y. Liang, F. Wang, C.Y. Dang, Y.H. Qian, An efficient rough feature selection algorithm with a multi-granulation view, Int. J. Approx. Reason. 53 (2012) 912–926.
[19] Q.H. Liu, F. Li, F. Min, M. Ye, An efficient knowledge reduction algorithm based on new conditional information entropy, Control Decis. 20 (2005) 878–882 (in Chinese).
[20] H.W. Liu, J.G. Sun, L. Liu, H.J. Zhang, Feature selection with dynamic mutual information, Pattern Recognit. 42 (2009) 1330–1339.
[21] M. Li, C.X. Shang, S.Z. Feng, J.P. Fan, Quick attribute reduction in inconsistent decision tables, Inform. Sci. 254 (2014) 155–180.
[22] A.W. McNabb, C.K. Monson, K.D. Seppi, Parallel PSO using MapReduce, in: Proceedings of 2007 IEEE Congress on Evolutionary Computation, CEC, IEEE Computer Society, 2007, pp. 7–16.
[23] F. Min, H. He, Y.H. Qian, W. Zhu, Test-cost-sensitive attribute reduction, Inform. Sci. 181 (2011) 4928–4942.
[24] D.Q. Miao, G.R. Hu, A heuristic algorithm for reduction of knowledge, J. Comput. Res. Dev. 36 (1999) 681–684 (in Chinese).
[25] D.Q. Miao, Y. Zhao, Y.Y. Yao, F.F. Xu, H.X. Li, Relative reducts in consistent and inconsistent decision tables of the Pawlak rough set model, Inform. Sci. 179 (2009) 4140–4150.
[26] M.R. Mohammad, Ś. Dominik, J. Wróblewski, Parallel island model for attribute reduction, in: S.K. Pal et al. (Eds.), PReMI 2005, LNCS 3776, Springer-Verlag, 2005, pp. 714–719.
[27] S.H. Nguyen, H.S. Nguyen, Some efficient algorithms for rough set methods, in: Proceedings of the International Conference on Information Processing and Management of Uncertainty on Knowledge Based Systems (IPMU'96), Granada, Spain, 1996, pp. 1451–1456.
[28] Z. Pawlak, Rough sets, Int. J. Comput. Inform. Sci. 11 (1982) 341–356.
[29] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data, Kluwer Academic Publishers, Boston, 1991.
[30] Y.H. Qian, J.Y. Liang, W. Pedrycz, C.Y. Dang, Positive approximation: an accelerator for attribute reduction in rough set theory, Artif. Intell. 174 (2010) 597–618.
[31] Y.H. Qian, S.Y. Li, J.Y. Liang, Z.Z. Shi, F. Wang, Pessimistic rough set based decisions: a multigranulation fusion strategy, Inform. Sci. 264 (2014) 196–210.
[32] J. Qian, D.Q. Miao, Z.H. Zhang, W. Li, Hybrid approaches to attribute reduction based on indiscernibility and discernibility relation, Int. J. Approx. Reason. 52 (2) (2011) 212–230.
[33] J. Qian, D.Q. Miao, Z.H. Zhang, Knowledge reduction algorithms in cloud computing, Chin. J. Comput. 34 (12) (2011) 2332–2343 (in Chinese).
[34] A. Skowron, C. Rauszer, The discernibility matrices and functions in information systems, in: R. Slowiński (Ed.), Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, Kluwer, Dordrecht, 1992.
[35] A. Srinivasan, T.A. Faruquie, Sachindra Joshi, Data and task parallelism in ILP using MapReduce, Mach. Learn. 86 (1) (2012) 141–168.
[36] R. Susmaga, Tree-like parallelization of reduct and construct computation, in: S. Tsumoto et al. (Eds.), RSCTC 2004, LNAI 3066, Springer-Verlag, 2004, pp. 455–464.
[37] J. Wang, J. Wang, Reduction algorithms based on discernibility matrix: the ordered attributes method, J. Comput. Sci. Technol. 16 (2001) 489–504.
[38] G.Y. Wang, H. Yu, D.C. Yang, Decision table reduction based on conditional information entropy, Chin. J. Comput. 25 (2002) 760–766 (in Chinese).
[39] L.H. Wang, G.F. Wu, Attribute reduction based on parallel symbiotic evolution, Chin. J. Comput. 26 (5) (2003) 630–635 (in Chinese).
[40] C.Z. Wang, Q. He, D.G. Chen, Q.H. Hu, A novel method for attribute reduction of covering decision systems, Inform. Sci. 254 (2014) 181–196.
[41] X. Xu, J. Jager, H.P. Kriegel, A fast parallel clustering algorithm for large spatial databases, Data Mining Knowl. Discov. 3 (1999) 263–290.
[42] Z.Y. Xu, Z.P. Liu, B.R. Yang, et al, A quick attribute reduction algorithm with complexity of $\max(O(|C||U|), O(|C|^2|U/C|))$, Chin. J. Comput. 29 (3) (2006) 611–615 (in Chinese).
[43] D. Yamaguchi, Attribute dependency functions considering data efficiency, Int. J. Approx. Reason. 51 (1) (2009) 89–98.
[44] M. Yang, An incremental updating algorithm of the computation of a core based on the improved discernibility matrix, Chin. J. Comput. 29 (3) (2006) 407–413 (in Chinese).

[45] 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, et al. (Eds.), Rough Set and Knowledge Technology, Lecture Notes in Computer Science, vol. 6401, Springer, Berlin/Heidelberg, 2010, pp. 672–678.
[46] X.B. Yang, Y.S. Qi, X.N. Song, J.Y. Yang, Test cost sensitive multigranulation rough set: model and minimal cost selection, Inform. Sci. 250 (2013) 184–199.
[47] Y.Y. Yao, Y. Zhao, Discernibility matrix simplification for constructing attribute reducts, Inform. Sci. 7 (2009) 867–882.
[48] Y.Y. Yao, L.Q. Zhao, A measurement theory view on the granularity of partitions, Inform. Sci. 213 (2012) 1–13.
[49] D.Y. Ye, Z.J. Chen, An improved discernibility matrix for computing all reducts of an inconsistent decision table, in: Proceedings of the Fifth IEEE International Conference on Cognitive Informatics (ICCI2006), IEEE Computer Society, 2006, pp. 305–308.
[50] D.Y. Ye, Z.J. Chen, S.L. Ma, A novel and better fitness evaluation for rough set based minimum attribute reduction problem, Inform. Sci. 222 (2013) 413–423.
[51] S.C. Yusta, Different metaheuristic strategies to solve the feature selection problem, Pattern Recognit. Lett. 30 (2009) 525–534.
[52] Y. Zhao, Y.Y. Yao, F. Luo, Data analysis based on discernibility and indiscernibility, Inform. Sci. 177 (2007) 4959–4976.
[53] W.Z. Zhao, H.F. Ma, Q. He, Parallel K-Means clustering based on MapReduce, in: M.G. Jaatun, G. Zhao, C. Rong (Eds.), Cloud Computing, CloudCom2009, Springer-Verlag, 2009, pp. 674–679.
[54] J.B. Zhang, T.R. Li, D. Ruan, et al, A parallel method for computing rough set approximations, Inform. Sci. 194 (2012) 209–223.
[55] D. Zinn, S. Bowers, S. Köhler, B. Ludäscher, Parallelizing XML data-streaming workflows via MapReduce, J. Comput. Syst. Sci. 76 (6) (2010) 447–463.
[56] N. Zhong, J.Z. Dong, Using rough sets with heuristics for feature selection, J. Intell. Inform. Syst. 16 (2001) 199–214.