Received: 15 June 2016 | Revised: 8 September 2016 | Accepted: 10 September 2016

DOI 10.1002/cpe.4015

WILEY

RESEARCH ARTICLE

A parallel C4.5 decision tree algorithm based on MapReduce

Yashuang Mu | Xiaodong Liu | Zhihao Yang | Xiaolin Liu

School of Control Science and Engineering, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology,

Correspondence
Xiaodong Liu, School of Control Science and Engineering, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China.
Email: xdliuros@dlut.edu.cn

Funding Information
Natural Science Foundation of China, Grant/Award Number: 61673082 , 61533005

Summary

In the supervised classification, large training data are very common, and decision trees are widely used. However, as some bottlenecks such as memory restrictions, time complexity, or data complexity, many supervised classifiers including classical C4.5 tree cannot directly handle big data. One solution for this problem is to design a highly parallelized learning algorithm. Motivated by this, we propose a parallelized C4.5 decision tree algorithm based on MapReduce (MR-C4.5-Tree) with 2 parallelized methods to build the tree nodes. First, an information entropy-based parallelized attribute selection method (MR-A-S) on several subsets for MR-C4.5-Tree is proposed to confirm the best splitting attribute and the cut points. Then, a data splitting method (MR-D-S) in parallel is presented to partition the training data into subsets. At last, we introduce the MR-C4.5-Tree learning algorithm that grows in a top-down recursive way. Besides, the depth of the constructed decision tree, the number of samples and the maximal class probability in each tree node are used as the termination conditions to avoid the over-partitioning problem. Experimental studies show the feasibility and the good performance of the proposed parallelized MR-C4.5-Tree algorithm.

KEYWORDS

C4.5, decision trees, MapReduce, parallel computing

1 | INTRODUCTION

As the amount of data generated in our daily life is increasing extremely quickly[1–5], it is unavoidable to learn from big data in many data mining areas such as pattern recognition, machine learning, image processing, and information retrieval[6]. Many data mining approaches have been proposed so far to handle the small or medium data sets, but few of them can be applied to the analysis of large data sets.[7,8] The main bottlenecks in learning from big data can be summed up in the following aspects:

- **Memory restrictions:** It is difficult to keep the whole training data set or most of it in memory on 1 computer.
- **Time complexity:** Finishing the computation within a tolerable time on a single computer is very hard.
- **Data complexity:** The high-dimensional and multi-modal features of the data make a far-reaching influence on the performance and efficiency of research results.

Because of the above bottlenecks, the parallelization of algorithm becomes a common and reliable choice. MapReduce frameworks[9] such as Hadoop and Spark have been proved to be effective on the parallelization of algorithm for many data mining operations. MapReduce is very convenient for distributed computing, which abstracts away from many of the difficulties in parallelizing data management operations across a cluster of commodity machines.

In this work, we mainly focus on the parallelized decision tree learning algorithms. Decision trees,[10] which provide a simple and understandable method to describe the progress of decision making based on the past knowledge, are tree structures made up by internal nodes and leaf nodes. Each internal node can be split into 1 or more children relying on an attribute of the training data for making a decision, and each leaf node is associated with a class label or an outcome. Because of its user-friendly and highly transparent characteristics, decision trees are promising tools for designing a parallelized learning algorithm.

Some descriptions could be found from the following literatures about the parallelization of sequential decision tree for the bottlenecks on big data. J Shafer et al[11] first developed an algorithm named SPRINT, which attempted to improve the structures of data in decision tree growth to avoid the memory restrictions, but it required an all-to-all broadcast of instances IDs when the good split points were computed. Joshi et al[12] presented a new parallel formulation of decision tree classifier similar to SPRINT. Rather than building the tree in a depth-first manner, it used a breadth-first tree growth to avoid possible load imbalance in a parallel computing framework. Srivastava et al[13] proposed 2 parallel decision tree models respectively based on synchronous construction method and partitioned construction method, but the modes suffered from high communication overhead and load imbalance for bushy trees. Sheng et al[14] provided a parallel decision tree that divided the input data into 4 subsets and offered it to user authentication only by constructing a parallel tree for each user. Panda et al[15] designed a PLANET algorithm based on a series of distributed computations. However, it was limited by commodity hardware for massive-scale parallel computing. Ben-Haim et al[16] studied a parallel decision tree algorithm for large streaming data classification by compressing data to a fixed amount of memory and constructing histograms at processors, which may not avoid the memory restrictions primarily. Using a scalable regression tree algorithm, Yin et al[17] presented an open-source implement of the PLANET on the basis of MapReduce framework of the Hadoop, but the impact of Hadoop Distributed File System (HDFS) input/output performance remained to be improved. The solution proposed by Calistru et al[18] was a method to adapt the dsCART algorithm to horizontal parallelism by implementing the MapReduce programming model. It should be noted that some parameters in this solution would influence the behavior of the algorithm seriously. These researches exhibit good effectiveness and performance on big data classification. Nevertheless, the over-partitioning problem is neglected in these works, which may lead to an over-fitted and redundant tree, especially for large data set. For the over-partitioning problem, Ran Wang[6] designed an extreme learning machine tree model based on parallel computation methods. However, it cannot work on the data sets with mixed type attributes.

To be able to solve the over-partitioning and the mixed type attributes problems on large data set, we suggest a parallel decision tree learning algorithm named MR-C4.5-Tree under the guidance of the MapReduce framework of Hadoop[19] and the C4.5 algorithm.[20,21] Hadoop is very convenient for distributing computing problems, and the C4.5 algorithm with an ability to process data sets with the mixed type attributes is one of the most broadly used decision tree algorithms. The suggested MR-C4.5-Tree takes the advantages and implements the distributed computation on each tree node building. To build the tree nodes, the suggested MR-C4.5-Tree parallel decision tree learning algorithm contains 2 distributed computations: MR-A-S and MR-D-S. The MR-A-S is MapReduce framework-based parallel method for the best splitting attribute selection of the MR-C4.5-Tree. As long as the best splitting attribute and the cut points are confirmed by the MR-A-S, the MR-D-S will split the training data set into several subsets in a parallel way. Moreover, to be able to address the over-partitioning problem, the depth of the constructed decision tree, the number of samples, and the maximal class probability in each tree node are used as the termination conditions. The main contributions of the proposed highly parallel MR-C4.5-Tree method are as follows:

- It overcomes the memory restrictions and time complexity problem of the above-mentioned bottlenecks because it is based on MapReduce framework.
- It resolves the over-partitioning problem as when one of some proper stopping conditions is satisfied, the progress of structure tree will be stopped.
- It can perform well on the data with mixed type attributes as the proposed solution is based on the C4.5 algorithm.

The remaining of this paper is organized as follows. Section 2 makes an introduction about Hadoop. Section 3 reviews the basic knowledge of the C4.5 algorithm. In Section 4, the parallelized decision tree algorithm MR-C4.5-Tree is introduced. Section 5 shows our experimental results. Finally, Section 6 presents the conclusion and scopes the directions of future research.

2 | HADOOP INTRODUCTION

Hadoop[19] is an open-source software based on MapReduce parallel computing model and provides a simple programming model for users to make distributed computing.[22] Hadoop mainly consists of Hadoop MapReduce, HDFS, Hadoop Yet Another Resource Negotiator, and Hadoop Common. In the following sections, we simply introduce Hadoop MapReduce module and the HDFS. The Hadoop MapReduce module involves a "Map" stage, distributing a data set among multiple servers and operating on the data, and a "Reduce" stage where the partial results are recombined. To store data, Hadoop uses its own distributed file system, HDFS, which makes data available to multiple computing nodes.

2.1 | Hadoop MapReduce

Hadoop MapReduce is a programming framework for parallel processing of large data sets.[22] The programming framework can be written in various languages such as C/C++, Java, Python, and Ruby. Besides, the MapReduce program can decompose big data analysis into many single and inherently parallel tasks on nodes of the clusters. The MapReduce work contains 2 processing phases: Map and Reduce. Usually, the Map phase with a Map function processes the input data set and produces some intermediate outputs. Then, these intermediate outputs are combined in the Reduce phase via a Reduce function by some way to export the final results. Figure 1 presents the detailed processing procedure of the MapReduce framework.

As shown in Figure 1, both of the Map and Reduce phases use the $< key,value >$ pairs as inputs and outputs of the 2 functions. In the Map phase, the Map function treats a single $< key,value >$ pair as an input and outputs a list of intermediate $< key,value >$ pairs. The form can be depicted as follows:

$$\text{map} < key_1, value_1 > \Rightarrow \text{list} < key_2, value_2 > . \tag{1}$$

Then, all the output intermediate $< key,value >$ pairs are grouped by *key* between the 2 functions. Finally, in the Reduce phase, a new output $< key,value >$ pair is generated based on the aggregated pairs by *key* in the Reduce function. The form can be represented as follows:

$$\text{reduce} < key_2, \text{list}(value_2) > \Rightarrow < key_2, value_3 > . \tag{2}$$

2.2 | Hadoop distributed file system

A distributed filesystem named HDFS is closely related to the Hadoop.[22] The HDFS runs on a cluster of computers designed for storing big data. Just as a disk has the concept of block size, the HDFS also has a block size, and the unit of the block is much larger (the default is 128 MB). The file stored on HDFS can be larger than any single machine disk of the cluster because the file can be stored on 2 or more disks in cluster as independent unit.

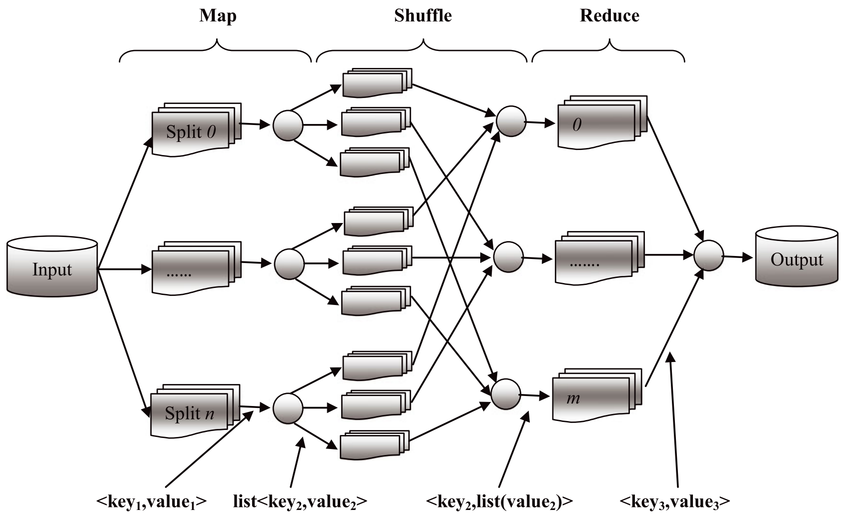

FIGURE 1 The detailed processing procedure of the MapReduce framework

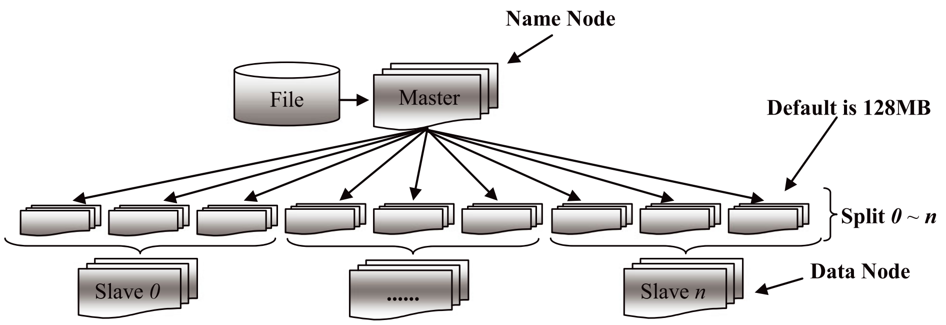

FIGURE 2 The distribution of a file in Hadoop Distributed File System

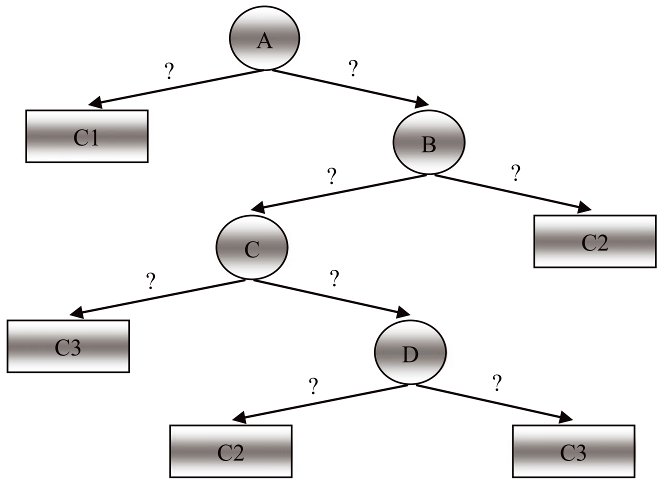

FIGURE 3 An example of tree model

An HDFS cluster contains 2 type nodes. The first type, called name node, is operating in a master-worker pattern as a rule of master. The second type is data nodes as a rule of workers and the number of data nodes is usually more than 1. The name node manages the name space of HDFS. It not only tells the data nodes to store and retrieve blocks but also periodically gets a report about the lists of blocks from the data nodes. The main work of the data nodes is storing data and handling data mining tasks from the name node. Figure 2 shows the distribution of a file on HDFS in detail.

3 | THE C4.5 DECISION TREE ALGORITHM

In this section, we mainly make a review about the C4.5 decision tree algorithm. In 1993, Ross Quinlan[20] developed the C4.5 decision tree algorithm based on his earlier ID3 algorithm.[21] Especially, it became the no. 1 popular algorithm in the *Top 10 Algorithm in Data Mining* preeminent paper published by Springer LNCS in 2008.[23] Nowadays, considered as a standard algorithm to include classification rules in the form of decision tree, the C4.5 decision tree has become one of the most broadly used tree models for classifications.

3.1 Tree models

Let $\mathbb{A} = \{A_1, A_2, \ldots, A_n\}$ be a set of n attributes with domains $\mathbb{D}_1, \mathbb{D}_2, \ldots, \mathbb{D}_n$, respectively. Let Y be an output with domains $\mathbb{D}_Y$. Consider a data set $\mathbf{X} = \{(x_i, y_i)|x_i \in \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_n, y_i \in \mathbb{D}_Y\}$ sampled from an unknown distribution, where x_i has an output y_i associated with it. For a given data set $\mathbf{X}$, the goal in supervised learning is to learn a model (or function) $F : \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_n \rightarrow \mathbb{D}_Y$ that best approximates the true distribution of $\mathbf{X}$. If $\mathbb{D}_Y$ is categorical, the learning problem is a classification problem; if $\mathbb{D}_Y$ is continuous, it is a regression problem.[15]

Algorithm 1: Construct a C4.5-Tree.

```
Input: A root Node $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i$ is the $i$-th instance with $n$
       attributes $\{A_k\}_{k=1}^n$.
Output: A C4.5-Tree.
1  $\Omega$ is initialized as an empty set.
2  Add $\mathbf{X}$ to $\Omega$.
3  while $\Omega$ is not empty do
4      for each attribute $A_k, k = 1, 2, \ldots, n$ do
5          if $A_k$ is numerical then
6              Sort its values $x_{1k}, \ldots, x_{Nk}$ and record as $x^*_{1k}, \ldots, x^*_{Nk}$.
7              Find all cut points $cp_{ik} = \frac{x^*_{ik}+x^*_{i+1,k}}{2}, i = 1, \ldots, N-1$.
8              for each cut point $cp_{ik}$ do
9                  Calculate the Information Gain:
10                 $Gain(cp_{ik}) = Info(\mathbf{X}) - \left[\frac{|\mathbf{X}_i^1|}{|\mathbf{X}|}Info(\mathbf{X}_i^1) + \frac{|\mathbf{X}_i^2|}{|\mathbf{X}|}Info(\mathbf{X}_i^2)\right]$,
                   where
                   $\mathbf{X}_i^1 = \{\mathbf{x}_i \in \mathbf{X}|x_{ik} \le cp_{ik}\}, \mathbf{X}_i^2 = \{\mathbf{x}_i \in \mathbf{X}|x_{ik} > cp_{ik}\}$
                   and symbol $|\cdot|$ is the size of $\cdot$.
11             end
12             Select the optimal cut point $cp_k = cp_{i^*k}$ of $A_k$, where
               $i^* = \arg\max_i \{Gain(cp_{ik})\}_{i=1}^{N-1}$.
13             Calculate the split information of $cp_k$:
14                 $Split(cp_k) = -\left[\frac{|\mathbf{X}_{i^*k}^1|}{|\mathbf{X}|}\log_2\frac{|\mathbf{X}_{i^*k}^1|}{|\mathbf{X}|} + \frac{|\mathbf{X}_{i^*k}^2|}{|\mathbf{X}|}\log_2\frac{|\mathbf{X}_{i^*k}^2|}{|\mathbf{X}|}\right]$.
15         else
16             Cut points: $cp_k = \bigcup_{i=1}^C \{x'_{ik}\}$, where $x'_{ik} \in \{x_{1k}, \ldots, x_{Nk}\}$, $C$
               is the number of attribute values.
17             Calculate the Information Gain:
18                 $Gain(cp_k) = Info(\mathbf{X}) - \sum_{i=1}^C \frac{|\mathbf{X}^i|}{|\mathbf{X}|}Info(\mathbf{X}^i)$, where
               $\mathbf{X}^i = \{\mathbf{x}_i \in \mathbf{X}|x_{ik} = x'_{ik}\}$.
19             Calculate the split information of $cp_k$:
20                 $Split(cp_k) = -\sum_{i=1}^C \frac{|\mathbf{X}^i|}{|\mathbf{X}|}\log_2\frac{|\mathbf{X}^i|}{|\mathbf{X}|}$.
21         end
22         Calculate the gain ratio of $A_k$: $Ratio(A_k) = \frac{Gain(cp_k)}{Split(cp_k)}$.
23     end
24     Get the best attribute $A_{k^*}$ and cut points $cp_{k^*}$, where
       $k^* = \arg\max_k \{Ratio(A_k))\}_{k=1}^n$.
25     Split $\mathbf{X}$ into $m$ subsets $\{\mathbf{X}_i\}_{i=1}^m$ based on attribute $A_{k^*}$ and cut
       points $cp_{k^*}$.
26     Remove $\mathbf{X}$ and add $\{\mathbf{X}_i\}_{i=1}^m$ to $\Omega$.
27 end
```

Tree models represent F by recursively partitioning the data space $\mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_n$ into non-overlapping regions, with a simple model in each region (as an example tree model shown in Figure 3). The internal nodes (e.g., Node B in Figure 3) in tree models define region boundaries in the data space. Each region boundary is represented as a predicate on an attribute in $\mathbb{A}$ (e.g., symbol "?" in Figure 3). The path from the root to a leaf node (e.g., the left child of C in Figure 3) in the tree defines a region. Leaf nodes contain a region prediction that in most cases is a constant value or some simple function (e.g., C3 in Figure 3).

For the prediction of an unknown x, the tree is traversed to find the region containing x. The region containing x is the path from the root to a leaf in the tree along which all non-leaf predicates are true when to evaluate on x. The prediction given by the leaf is used as the value for $F(x)$.

3.2 The C4.5 algorithm

As an extension of the ID3 algorithm, the C4.5 algorithm uses the information gain ratio as the default criteria to choose the best splitting attributes. The information gain ratio effectively avoids the bias of selecting attributes with many values that occurs in the ID3 algorithm. The C4.5 algorithm makes some improvements on ID3, such as handling the data with numerical and nominal attributes, dealing with the data sets with missing attribute values, and pruning trees after creation. The principal disadvantage of the C4.5 algorithm is that it requires more amount of central processing unit time and memory as the samples increase.[23] In the training of the C4.5 algorithm, the tree is constructed in a top-down recursive way as presented in Algorithm 1.

In the Algorithm 1, the symbol *Info*(*) is the information entropy, which reflects the class impurity and the amount of information in *. The symbol *Info*(*) is defined as

$$Info(*) = -\sum_{i=1}^{m} p_i(*)\log_2(p_i(*)), \tag{3}$$

where m is the class number of * and $p_i(*)$ is the i-th class probability in *.

4 THE MR-C4.5-TREE APPROACH

In this section, we introduce the MR-C4.5-Tree algorithm. According to the process of traditional decision tree models, this section mainly consists of the following parts:

- Attribute selection.
- Data set splitting.
- MR-C4.5-Tree construction.

To be able to avoid the over-partitioning problem, 3 parameters, the depth of tree H^*, the minimum accuracy rate θ, and the minimum number of samples N^*, are provided as the termination conditions to construct a decision tree in this algorithm.

4.1 Attribute selection

The information gain ratio is a common solution for attribute selection problem in decision trees. In the traditional decision tree algorithm, if an attribute has a maximum information gain ratio, it will be selected as the best splitting attribute.

In our paper, we also use the information gain ratio as the solution to decide which attributes should be selected, similar to the traditional tree algorithm way. The difference from C4.5 is that the traditional decision tree algorithm computes the information gain ratio based on the whole data set while the proposed algorithm computes the information gain ratio based on the several small subsets that are divided from the whole data by Hadoop framework. To be able to find the target attribute from the subsets, we calculate the sum of the information gain ratio of the same attribute in the subsets and choose the attribute corresponding to the maximum sum as the best splitting attribute. The detailed process can be found in Algorithm 2.

As the Algorithm 2 is an attribute selection algorithm based on MapReduce framework of Hadoop, it can be named as MR-A-S algorithm. In the MR-A-S algorithm, the symbol *Info*(*) is the information entropy defined by Equation 3.

4.2 | Data set splitting

Once the best splitting attribute A_{k^*} and the cut points cp_{k^*} are confirmed, the following work is splitting the data set into several subsets. Algorithm 3, a data splitting algorithm (MR-D-S) based on the Hadoop MapReduce framework, introduces the splitting progress in detail.

Over-partitioning problem is a pervasive problem in decision trees. Setting some conditions to limit the decision tree size is a common way to solve the problem. As described in Algorithm 3, we also set some conditions to limit the size of tree. When the largest class probability in $\mathbf{X}_{id}$ is bigger than $\theta \in [0, 1]$ or the size of $\mathbf{X}_{id}$ is smaller than N^* or the examples in $\mathbf{X}_{id}$ belong to the same class, we will create $\mathbf{X}_{id}$ as a leaf node and do not add it to Ω for the next splitting.

4.3 | MR-C4.5-Tree construction

Based on the previous work of the attribute selection and data set splitting, the following task is constructing an MR-C4.5-Tree. Algorithm 4 describes the constructing progress about how to build a whole MR-C4.5-Tree. For each tree node building, we search the best splitting attribute by Algorithm 2 and split the node into several child nodes by Algorithm 3. Meanwhile, we limit the depth of the output MR-C4.5-Tree to be smaller than H^* to avoid over-partitioning during the constructing of MR-C4.5-Tree.

5 | EXPERIMENTAL STUDIES

In this section, we describe the experimental studies using the MR-C4.5-Tree algorithm. First, Section 5.1 introduces the data sets used in experiments. Later, Section 5.2 provides the computing environment. Then, to be able to verify the feasibility and the performance of the proposed algorithm, some comparisons on the data sets between the MR-C4.5-Tree and C4.5 are presented in Section 5.3. Section 5.4 shows the analysis about parallel performance of the MR-C4.5-Tree.

Algorithm 2: The proposed MR-A-S algorithm.

```
   Input: A training data $\mathbf{X}$.
   Output: An selected attribute $A_{k^*}$ and the cut points $cp_{k^*}(\mathbf{X})$.
1  Initialize a Hadoop Job SELECTJOB:
2      Set SelectTaskMapper as the Mapper class.
3      Set SelectTaskReducer as the Reducer class.
4      Adjust the block size of HDFS until the data set $\mathbf{X}$ can be split
   into m subsets $\{\mathbf{X}_j\}_{j=1}^m$.
5  In the j-th SelectTaskMapper:
   Input: $\mathbf{X}_j = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i$ is the i-th instance with n attributes
       $\{A_k\}_{k=1}^n$.
   Output: $\langle key, value\rangle = \langle A_k, [Ratio_k(\mathbf{X}_j, A_k), cp_k(\mathbf{X}_j)]\rangle$.
6  for each attribute $A_k, k = 1, 2, \dots, n$ do
7      if $A_k$ is numerical then
8          Sort its values $x_{1k}, \dots, x_{Nk}$ and record as $x_{1k}^*, \dots, x_{Nk}^*$.
9          Find all cut points $cp_{ik} = \frac{x_{ik}^* + x_{i+1,k}^*}{2}, i = 1, \dots, N-1$.
10         for each cut point $cp_{ik}$ do
11             Calculate the Information Gain:
12             $Gain(\mathbf{X}_j, cp_{ik}) = Info(\mathbf{X}_j) - \left[\frac{|\mathbf{X}_{ij}^1|}{|\mathbf{X}_j|} Info(\mathbf{X}_{ij}^1) + \frac{|\mathbf{X}_{ij}^2|}{|\mathbf{X}_j|} Info(\mathbf{X}_{ij}^2)\right]$,
               where
               $\mathbf{X}_{ij}^1 = \{\mathbf{x}_i \in \mathbf{X}_j | x_{ik} \le cp_{ik}\}, \mathbf{X}_{ij}^2 = \{\mathbf{x}_i \in \mathbf{X}_j | x_{ik} > cp_{ik}\}$ and
               symbol $|\cdot|$ is the size of $\cdot$.
13         end
14         Select the optimal cut point $cp_k(\mathbf{X}_j) = cp_{i^*k}$ of $A_k$:
15             $i^* = \arg\max_i \left\{Gain(\mathbf{X}_j, cp_{ik})\right\}_{i=1}^{N-1}$.
16         Calculate the split information of $cp_k(\mathbf{X}_j)$:
17             $Split(\mathbf{X}_j, cp_k(\mathbf{X}_j)) = -\left[\frac{|\mathbf{X}_{i^*k}^1|}{|\mathbf{X}_j|}\log_2\frac{|\mathbf{X}_{i^*k}^1|}{|\mathbf{X}_j|} + \frac{|\mathbf{X}_{i^*k}^2|}{|\mathbf{X}_j|}\log_2\frac{|\mathbf{X}_{i^*k}^2|}{|\mathbf{X}_j|}\right]$.
18     else
19         Cut points: $cp_k = \bigcup_{i=1}^C \{x'_{ik}\}$, where $x'_{ik} \in \{x_{1k}, \dots, x_{Nk}\}$, C is
           the number of attribute values.
20         Calculate the Information Gain:
21             $Gain(\mathbf{X}_j) = Info(\mathbf{X}_j) - \sum_{i=1}^C \frac{|\mathbf{X}_j^i|}{|\mathbf{X}_j|} Info(\mathbf{X}_j^i)$, where
           $\mathbf{X}_j^i = \{\mathbf{x}_i \in \mathbf{X}_j | x_{ik} = x'_{ik}\}$ and symbol $|\cdot|$ is the size of $\cdot$.
22         Calculate the split information of $cp_k$:
23             $Split(\mathbf{X}_j, cp_k(\mathbf{X}_j)) = -\sum_{i=1}^C \frac{|\mathbf{X}_j^i|}{|\mathbf{X}_j|}\log_2\frac{|\mathbf{X}_j^i|}{|\mathbf{X}_j|}$.
24     end
25     Calculate the gain ratio of $A_k$:
26         $Ratio(\mathbf{X}_j, A_k) = \frac{Gain(\mathbf{X}_j, cp_k(\mathbf{X}_j))}{Split(\mathbf{X}_j, cp_k(\mathbf{X}_j))}$.
27     Mapper Output: $\langle key, value\rangle = \langle A_k, [Ratio_k(\mathbf{X}_j, A_k), cp_k(\mathbf{X}_j)]\rangle$.
28 end
29 In the SelectTaskReducer:
   Input: $\langle key, value\rangle =$
       $\langle A_k, List([Ratio_k(\mathbf{X}_j, A_k), cp_k(\mathbf{X}_j)], j = 1, 2, \dots, m)\rangle$.
   Output: $\langle key, value\rangle = \langle A_{k^*}, [Ratio_{k^*}(\mathbf{X}), cp_{k^*}(\mathbf{X})]\rangle$.
30 for each attribute $A_k$ do
31     $Ratio_k(\mathbf{X}) = \sum_{j=1}^m Ratio_k(\mathbf{X}_j, A_k)$.
32     $cp_k(\mathbf{X}) = \frac{\sum_{j=1}^m cp_k(\mathbf{X}_j)}{m}$ ($A_k$ is numerical) or $cp_k(\mathbf{X}) = \bigcup_{j=1}^{j=m} cp_k(\mathbf{X}_j)$ ($A_k$
       is not numerical).
33 end
34 Select the optimal index $k^* = \arg\max_k \{Ratio_k(\mathbf{X})\}_{k=1}^n$.
35 Reducer Output: $\langle key, value\rangle = \langle A_{k^*}, [Ratio_{k^*}(\mathbf{X}_j), cp_{k^*}(\mathbf{X})]\rangle$.
36 return $A_{k^*}$ and $cp_{k^*}(\mathbf{X})$.
```

Algorithm 3: The proposed MR-D-S algorithm.

```
Input: A training data $\mathbf{X}$; an attribute $A_{k^*}$; the cut point $cp_{k^*}$; an
       empty set $\Omega$.
Output: $\Omega$.
1  Initialize a Hadoop Job SPLITJOB:
2      Set SplitTaskMapper as the Mapper class.
3      Set SplitTaskReducer as the Reducer class.
4      Adjust the block size of HDFS until the data set $\mathbf{X}$ can be split
   into $m$ subsets $\{\mathbf{X}_j\}_{j=1}^{m}$.
5  In the $j$-th SplitTaskMapper:
   Input: $\mathbf{X}_j = \{\mathbf{x}_i\}_{i=1}^{N}$, where $\mathbf{x}_i$ is the $i$-th instance with $n$ attributes
       $\{A_k\}_{k=1}^{n}$.
   Output: $\langle key, value \rangle = \langle id, \mathbf{x}_i \rangle$, where the $id$ is the label of output
       subset.
6  for each instance $\mathbf{x}_i$, $i = 1, 2, \ldots, N$ do
7      if $A_{k^*}$ is numerical then
8          if $\mathbf{x}_{ik^*} > cp_{k^*}$ then
9              $id = 1$.
10         else
11             $id = 0$.
12         end
13     else
14         $id = \mathbf{x}_{ik^*}$.
15     end
16     Mapper Output:$\langle key, value \rangle = \langle id, \mathbf{x}_i \rangle$.
17 end
18 In the SplitTaskReducer:
   Input: $\langle key, value \rangle = \langle id, \mathbf{X}_{id} \rangle$, where $\mathbf{X}_{id}$ is the set of $List(\mathbf{x})$.
   Output: $\langle key, value \rangle = \langle id, \mathbf{X}_{id} \rangle$.
19 for each $\mathbf{X}_{id}$ do
20     Define $K$ is the class number; $\{P_i\}_{i=1}^{k}$ is the class probability and
       $N$ is the size of $\mathbf{X}_{id}$.
21     if $K > 1$ and $\max\{P_i\}_{i=1}^{k} \leq \theta$ and $N \geq N^*$ then
22         Build a no leaf node.
23         Add $\mathbf{X}_{id}$ to $\Omega$.
24     else
25         Build a leaf node.
26     end
27     Reducer Output:$\langle key, value \rangle = \langle id, \mathbf{X}_{id} \rangle$.
28 end
29 return $\Omega$.
```

5.1 | Data sets

To be able to test the behaviors of our approach in real-world applications, we collect 20 real-world data sets for the experimental studies in this paper. As described in Table 1, the collected 20 data sets contain 14 data sets only with numerical attributes, 2 data sets only with nominal attributes, and 4 data sets with mixed type attributes. The *Electricity* data set comes from the Massive Online Analysis[24] that is one of the most popular open-source frameworks for data stream mining. The *Electricity* data set was collected from the Australian New South Wales Electricity Market. The other 19 data sets related with many areas such as social sciences, life sciences, and medical sciences are obtained from the UCI Machine Learning Repository[25] that is a collection of databases, domain theories, and data generators used for the empirical analysis of machine learning algorithms.

Algorithm 4: The proposed MR-C45-Tree construction algorithm.

```
Input: A training data set $\Omega_0$; the maximum depth $H^*$.
Output: An MR-C4.5-Tree.
1  if Tree's depth $> H^*$ then
2      return
3  end
4  $\Omega$ is initialized as an empty set.
5  Select one node from $\Omega_0$, denoted by $\mathbb{X}$.
6  for $\mathbb{X}$ : $\Omega_0$ do
7      Get the best attribute $A_{k^*}$ and cut points $cp_{k^*}$ from $\mathbb{X}$ by
       Algorithm 2.
8      Split $\mathbb{X}$ into $n$ child nodes $\{\mathbb{X}_i\}_{i=1}^{n}$ based on $A_{k^*}$ and $cp_{k^*}$ by
       Algorithm 3.
9      Add $\{\mathbb{X}_i\}_{i=1}^{n}$ to $\Omega$.
10 end
11 Tree's depth =Tree's depth + 1.
12 if $\Omega$ is not empty then
13     $\Omega_0 = \Omega$.
14     Recursive search the new node from $\Omega_0$ by Algorithm 4.
15 end
```

5.2 | Computing environment

In our experiments, the proposed MR-C4.5-Tree algorithm is implemented with Java language. For convenience of coding, the MapReduce framework of Hadoop[19] and the data structures of WEKA[26] are used in the MR-C4.5-Tree classification system. We preset the parameters $H^* = 10$, $N^* = 4$ and $\theta = 0.90$ during the experiments according to the proposed algorithm. For each data set, all the results are the average values based on the 10-fold cross validation technique.

We evaluate the performance of the MR-C4.5-Tree classification system in a small cluster operating environment containing 1 host computer and 5 servant computers. The detailed construction of the cluster is given in Table 2.

5.3 | Comparisons between the MR-C4.5-Tree and C4.5

In this section, we make some comparisons between the MR-C4.5-Tree and C4.5 algorithm. The proposed MR-C4.5-Tree algorithm is performed on the cluster with the number of processors varying from 1 to 8 on the data sets in Table 1, respectively. The C4.5 algorithm is implemented with the class *J48.java* in WEKA meanwhile setting the *Unpruned* parameters as true and other parameters as false.

We mainly focus on the comparisons about the accuracy on testing data, the size of constructed trees, and the number of leaf nodes as respectively shown in Tables 3-5. The marked results in bold face present that the current algorithm is significantly better than the other. From the comparisons, the following observations could be found.

TABLE 1 The detailed information of the data sets in experiments

		Attributes				
No.	Data Sets	Numerical	Nominal	Class	Data Size	Instances
1	Auto Mpg	5	2	3	11.5 KB	398
2	Breast Cancer	0	9	2	23.8 KB	286
3	Breast Cancer W-D	30	0	2	117.6 KB	569
4	Breast Cancer W-O	0	9	2	16.0 KB	699
5	Breast Cancer W-P	33	0	2	42.0 KB	198
6	Credit Approval	6	9	2	30.0 KB	690
7	Glass	9	0	7	9.8 KB	214
8	Ionosphere	33	0	2	73.2 KB	351
9	Iris	4	0	3	4.4 KB	150
10	New Thyroid	5	0	3	4.8 KB	215
11	Parkinsons	22	0	2	34.3 KB	195
12	Pima	8	0	2	23.2 KB	768
13	Sonar	60	0	2	85.7 KB	208
14	SPECTF Heart	44	0	2	35.2 KB	267
15	Vehicle	18	0	4	53.8 KB	846
16	Wine	13	0	3	11.0 KB	178
17	Yeast	8	0	10	58.6 KB	1484
18	Adult	8	6	2	3.6 MB	32,561
19	Electricity	1	7	2	3.1 MB	45,312
20	Skin segmentation	3	0	2	3.4 MB	245,057

TABLE 2 The detailed construction of the cluster

No.	Nodes	CPU, GHz	RAM, GB	Operation System	Bit
1	Master	Pentium 5 3.30	8	Ubuntu 14.04.1 LTS	64
2	Slave1	Pentium 5 3.30	4	Ubuntu 14.04.1 LTS	64
3	Slave2	Pentium 5 3.30	4	Ubuntu 14.04.1 LTS	64
4	Slave3	Pentium 5 3.30	4	Ubuntu 14.04.1 LTS	64
5	Slave4	Pentium 5 3.30	2	Ubuntu 14.04.1 LTS	64
6	Slave5	Pentium 5 3.30	8	Ubuntu 14.04.1 LTS	64

Abbreviations: CPU, central processing unit; RAM, random-access memory.

First, we perform the MR-C4.5-Tree algorithm on each data set for the comparison of testing accuracy with the C4.5 algorithm. As shown in Table 3, the results of the 2 compared methods are extremely approximate, and the proposed algorithm even displays better performance on some data sets.

Second, as depicted by the last line in Table 3, the results with good performance of the proposed algorithm on different processors are 9, 11, 7, 7, 9, 11, 10, and 9 data sets out of 20, while the C4.5 algorithm is 7 data sets out of 20.

Third, the comparisons about the size of tree and the number of leaf nodes can be found in Tables 4 and 5, respectively. There are striking differences between the proposed algorithm and the C4.5 algorithm on the scale of the output tree. The scale of the MR-C4.5-Tree is far less than the C4.5 tree because the stop conditions H^*, N^*, and θ resolve the over-partitioning problem.

5.4 | Parallel performance of the MR-C4.5-Tree

In this section, we use MR-A-S (Algorithm 2), the main method of the MR-C4.5-Tree, to present the parallel performance. In the experiments, we randomly select the Adult data set from Table 1 to evaluate the parallel performance of the MR-A-S algorithm by computing the Speedup, Scaleup, and Sizeup.[27] The definitions of Speedup, Scaleup, and Sizeup can be found as follows.

- **Speedup:** It measures how much faster in an M computers system and can be expressed by

$$\text{Speedup}(M) = \frac{\text{executing time on } one \text{ computer}}{\text{executing time on } M \text{ computers}}. \tag{4}$$

TABLE 3 The comparison of testing accuracy between the MR-C4.5-Tree and C4.5

	Processors No. of the Proposed MR-C4.5								
Data Sets	**One**	**Two**	**Three**	**Four**	**Five**	**Six**	**Seven**	**Eight**	**C4.5**
Auto Mpg	0.747	0.754	0.758	0.766	0.771	0.787	0.772	0.763	**0.803**
Breast cancer	0.647	0.675	0.672	0.670	0.669	0.670	0.655	0.662	**0.686**
Breast cancer W-D	0.903	0.899	0.891	0.905	0.917	0.906	0.889	0.903	**0.926**
Breast cancer W-O	**0.939**	**0.947**	0.930	0.933	**0.942**	**0.939**	**0.940**	**0.949**	0.937
Breast cancer W-P	**0.737**	**0.716**	**0.702**	0.661	**0.758**	**0.737**	0.691	0.670	0.700
Credit approval	**0.832**	**0.848**	**0.840**	**0.851**	**0.819**	**0.830**	**0.847**	**0.844**	0.805
Glass	**0.570**	**0.538**	**0.589**	0.436	0.473	0.485	0.491	0.453	0.504
Ionosphere	0.869	**0.893**	0.883	0.883	0.849	0.866	0.886	0.869	0.892
Iris	**0.933**	**0.927**	**0.920**	**0.920**	**0.933**	**0.947**	**0.947**	0.896	0.920
New thyroid	0.879	0.842	0.851	0.786	0.809	0.823	0.827	0.833	**0.906**
Parkinsons	**0.738**	**0.738**	0.687	0.733	**0.758**	0.723	**0.795**	**0.815**	0.738
Pima	**0.723**	**0.741**	**0.749**	**0.729**	**0.724**	**0.732**	**0.737**	**0.742**	0.707
Sonar	0.476	**0.615**	0.380	**0.512**	**0.625**	**0.591**	**0.529**	**0.582**	0.489
SPECTF heart	0.652	0.670	**0.809**	**0.772**	**0.834**	**0.772**	**0.813**	**0.739**	0.718
Vehicle	0.554	0.557	0.561	0.570	0.632	0.612	0.649	0.650	**0.682**
Wine	0.851	0.758	0.826	0.820	0.826	0.809	0.753	0.803	**0.854**
Yeast	0.449	0.423	0.443	0.459	0.474	**0.530**	**0.492**	**0.507**	0.486
Adult	**0.848**	**0.844**	**0.842**	**0.841**	**0.841**	**0.839**	**0.838**	**0.838**	0.823
Electricity	**0.761**	**0.758**	0.716	**0.747**	0.721	**0.773**	**0.772**	**0.756**	0.732
Skin segmentation	0.909	0.916	0.897	0.915	0.916	0.915	0.914	0.910	**0.955**
Avg	0.7509	0.75230	0.7473	0.7455	**0.7646**	**0.7643**	0.7587	0.7592	0.7632
	(9/20)	(11/20)	(7/20)	(7/20)	(9/20)	(11/20)	(10/20)	(9/20)	(7/20)

- **Scaleup:** It means the ability of an M-times system for executing an M-times job during the same computing time as the original time, which can be measured as

$$\text{Scaleup}(M) = \frac{\text{executing time for processing data on } \textit{one} \text{ computer}}{\text{executing time for processing } M \times \textit{data} \text{ on } M \text{ computers}}. \tag{5}$$

- **Sizeup:** It evaluates how much longer an M-times data set takes on a given system than the original data set.

$$\text{Sizeup}(M) = \frac{\text{executing time for processing } M \times \textit{ data}}{\text{executing time for processing } \textit{data}}. \tag{6}$$

First, the parallel algorithm MR-A-S is evaluated on the cluster with the processors number varying from 1 to 8, and the data set sizes are 1-times, 20-times, 40-times, 60-times, and 80-times of the original data set, respectively. Figure 4 displays the Speedup performance of MR-A-S for the data sets with different times of the original data. The parallel system does not demonstrate a linear speedup because, in practice, the communication cost gradually enlarges with the increasing number of processors. As presented in Figure 4, the Speedup and the size of data set are highly correlated. The Speedup tends to be approximately linear when the data size increases, especially for the much bigger data sets. We further study the running time as shown in Figure 5 because the size of the data sets is different. The slopes of the curves decrease with the increasing of the processors. It is consistent with the actual situation that the bigger samples are, the higher Speedup can be achieved; the more processors are, the less running time can be spent.

Then, for the Scaleup performance, several scalability experiments are performed. According to the definition of Scaleup, the data sets that are 1-times, 3-times, 5-times, and 7-times of the original data set are performed on 1, 3, 5, and 7 processors, respectively. The Scaleup could be equal to 1 if the parallel system is ideal. However, in practice, the truth is that the Scaleup has a downtrend when the size of data set and the number of processors are gradually increasing, which can be verified in Figure 6. With the data set becoming larger, the Scaleup performance reduces slowly because of the good scalability of the proposed algorithm.

Finally, to evaluate the Sizeup of the proposed algorithm, we have studied a series of experiments in which the number of processors is fixed and the data sets are 1-times, 20-times, 40-times, 60-times, and 80-times of the original data set, respectively. As displayed in Figure 7, the proposed algorithm presents a good Sizeup performance.

TABLE 4 The comparison of the size of constructed tree between the MR-C4.5-Tree and C4.5

	Processors No. of the Proposed MR-C4.5								
Data Sets	**One**	**Two**	**Three**	**Four**	**Five**	**Six**	**Seven**	**Eight**	**C4.5**
Auto Mpg	**25.0**	**35.6**	**45.7**	**72.5**	**65.0**	**69.0**	**67.4**	**51.7**	99.8
Breast cancer	**84.3**	**79.8**	**77.0**	**75.5**	**74.7**	**74.3**	**71.1**	**69.0**	218.1
Breast cancer W-D	**6.4**	**3.6**	**4.4**	**5.6**	**5.6**	**4.6**	**4.6**	**4.2**	34.8
Breast cancer W-O	**38.6**	**40.6**	**41.6**	**43.3**	**42.8**	**43.2**	**40.7**	**41.1**	143.0
Breast cancer W-P	**20.0**	**27.8**	**26.8**	**20.8**	**20.2**	**15.6**	**16.0**	**12.8**	44.8
Credit approval	**26.0**	**29.5**	**28.4**	**29.5**	**32.3**	**36.9**	**56.3**	**46.6**	158.7
Glass	**33.8**	**37.0**	**34.6**	**26.4**	**22.0**	**18.8**	**17.4**	**16.4**	65.5
Ionosphere	**10.6**	**13.4**	**15.4**	**11.6**	**16.6**	**14.8**	**14.2**	**14.2**	38.4
Iris	**5.4**	**5.8**	**6.8**	**5.6**	**6.0**	**5.0**	**5.6**	**5.2**	12.6
New thyroid	**6.0**	**8.6**	**7.0**	**7.8**	**9.0**	**8.6**	**8.8**	**8.6**	22.2
Parkinsons	**14.2**	**14.8**	**12.2**	**10.6**	**10.2**	**9.4**	**8.0**	**8.0**	26.8
Pima	**25.8**	**28.6**	**33.0**	**30.8**	**44.6**	**41.6**	**46.2**	**40.2**	216.4
Sonar	**19.2**	**26.2**	**29.6**	**24.2**	**23.6**	**23.2**	**21.2**	**19.8**	41.6
SPECTF heart	**20.0**	**30.2**	**24.0**	**20.8**	**16.6**	**14.2**	**12.6**	**13.2**	51.2
Vehicle	**29.0**	**33.0**	**31.8**	**35.8**	**46.8**	**47.4**	**49.0**	**38.6**	230.8
Wine	**7.6**	**8.0**	**8.6**	**7.6**	**8.0**	**8.0**	**8.2**	**7.8**	14.2
Yeast	**29.8**	**31.8**	**37.2**	**37.6**	**40.2**	**40.6**	**44.2**	**47.0**	715.0
Adult	**32.6**	**46.6**	**86.1**	**112.6**	**115.1**	**136.3**	**142.2**	**141.5**	10043
Electricity	**32**	**29**	**29**	**29.1**	**29.3**	**29**	**28.9**	**28.4**	6222
Skin segmentation	**12.2**	**12.2**	**12.6**	**12.8**	**12.8**	**12.4**	**13.2**	**12.4**	478.2
Avg	**23.9**	**27.1**	**29.6**	**31.0**	**32.1**	**32.6**	**33.8**	**31.3**	943.9
	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(0/20)

TABLE 5 The comparison of the leaf's number between the MR-C4.5-Tree and the C4.5

	Processors No. of the Proposed MR-C4.5								
Data Sets	**One**	**Two**	**Three**	**Four**	**Five**	**Six**	**Seven**	**Eight**	**C4.5**
Auto Mpg	**13.1**	**20.6**	**29.5**	**53.3**	**49.1**	**54.0**	**53.3**	**41.1**	61.0
Breast cancer	**63.1**	**58.8**	**57.4**	**56.0**	**55.5**	**54.2**	**54.2**	**52.9**	181.0
Breast cancer W-D	**3.7**	**2.3**	**2.7**	**3.3**	**3.3**	**2.8**	**2.8**	**2.6**	17.9
Breast cancer W-O	**33.8**	**35.7**	**36.3**	**38.0**	**37.5**	**37.8**	**35.7**	**36.2**	128.8
Breast cancer W-P	**10.5**	**14.4**	**13.9**	**10.9**	**10.6**	**8.3**	**8.5**	**6.9**	22.9
Credit approval	**13.8**	**16.1**	**15.6**	**17.3**	**19.2**	**24.0**	**40.1**	**32.6**	99.2
Glass	**17.4**	**19.0**	**17.8**	**13.7**	**11.5**	**9.9**	**9.2**	**8.7**	33.3
Ionosphere	**5.8**	**7.2**	**8.2**	**6.3**	**8.8**	**7.9**	**7.6**	**7.6**	19.7
Iris	**3.2**	**3.4**	**3.9**	**3.3**	**3.5**	**3.0**	**3.3**	**3.1**	6.8
New thyroid	**3.5**	**4.8**	**4.0**	**4.4**	**5.0**	**4.8**	**4.9**	**4.8**	11.6
Parkinsons	**7.6**	**7.9**	**6.6**	**5.8**	**5.6**	**5.2**	**4.5**	**4.5**	13.9
Pima	**13.4**	**14.8**	**17.0**	**15.9**	**22.8**	**21.3**	**23.6**	**20.6**	108.7
Sonar	**10.1**	**13.6**	**15.3**	**12.6**	**12.3**	**12.1**	**11.1**	**10.4**	21.3
SPECTF heart	**10.5**	**15.6**	**12.5**	**10.9**	**8.8**	**7.6**	**6.8**	**7.1**	26.1
Vehicle	**15.0**	**17.0**	**16.4**	**18.4**	**23.9**	**24.2**	**25.0**	**19.8**	115.9
Wine	**4.3**	**4.5**	**4.8**	**4.3**	**4.5**	**4.5**	**4.6**	**4.4**	7.6
Yeast	**15.4**	**16.4**	**19.1**	**19.3**	**20.6**	**20.8**	**22.6**	**24.0**	358
Adult	**43.4**	**66.4**	**93.1**	**116.6**	**110.1**	**146.9**	**142.8**	**153.7**	7062
Electricity	**17.1**	**15.1**	**15**	**15.3**	**15.5**	**15**	**15.2**	**14.7**	3381
Skin segmentation	**6.6**	**6.6**	**6.8**	**6.9**	**6.9**	**6.7**	**7.1**	**6.7**	239.6
Avg	**15.2**	**17.7**	**19.5**	**21.3**	**21.4**	**23.2**	**23.8**	**22.8**	584
	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(20/20)	(0/20)

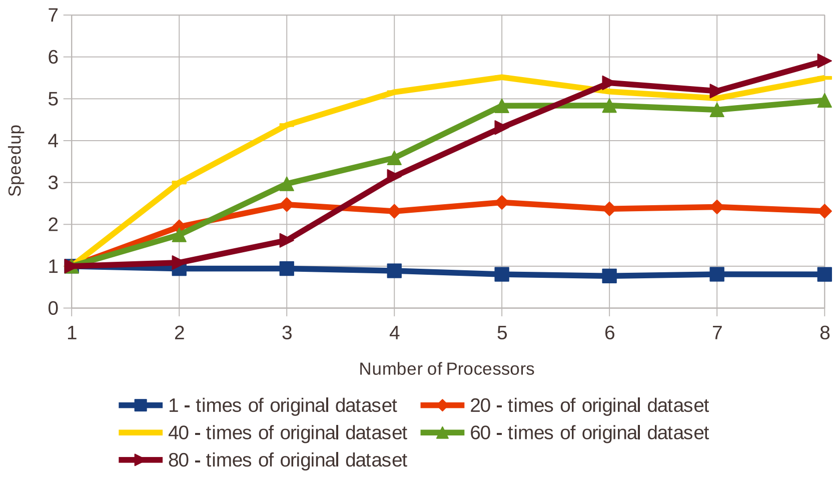

FIGURE 4 Speedup of the MR-A-S algorithm for the Adult data set

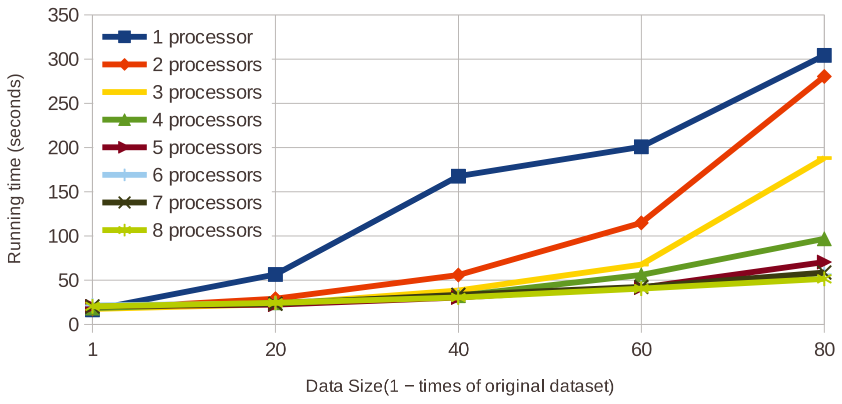

FIGURE 5 Running time vs. data size for the Adult data set

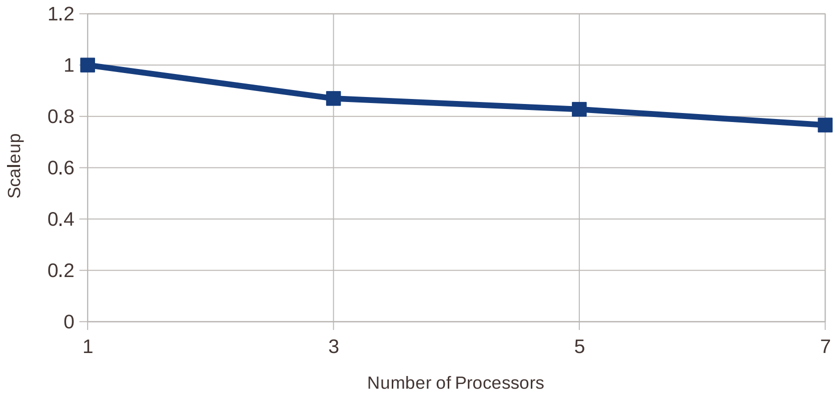

FIGURE 6 Scaleup of the MR-A-S algorithm for the Adult data set

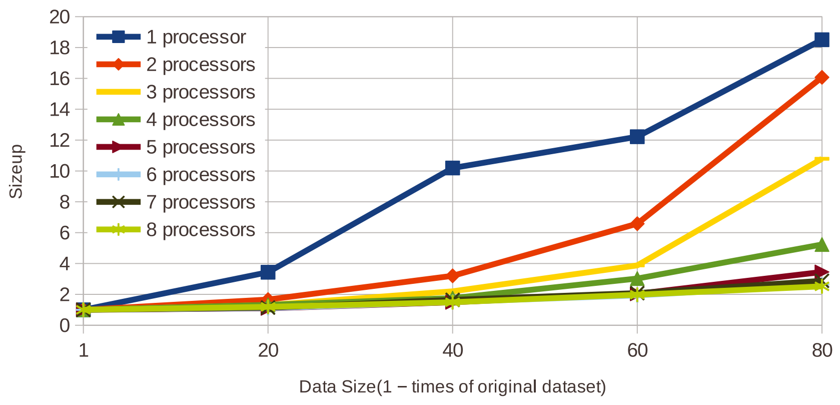

FIGURE 7 Sizeup of the MR-A-S algorithm for the Adult data set.

6 | CONCLUSION

The classical C4.5 algorithm is one of the most broadly used approaches on small and medium data sets in real-world applications. However, there are several drawbacks of the C4.5 algorithm performed on large data sets. In this paper, we develop a parallelized C4.5 decision tree learning algorithm called MR-C4.5-Tree to solve the challenges by using the MapReduce programming framework. To be able to construct MR-C4.5-Tree nodes, 2 parallelized methods used for selecting best splitting attributes and splitting data into subsets are separately suggested. By comparing with classical C4.5 algorithm, the results of statistical tests have shown that the proposed MR-C4.5-Tree is feasible and effective. Furthermore, the experimental studies about Speedup, Scaleup, and Sizeup confirm the good parallel performance of the proposed MR-C4.5-Tree algorithm.

In this paper, 3 termination conditions, the depth of tree, the minimum of samples, and the minimum accuracy rate, are provided to overcome the over-partitioning problem, but it needs to be preseted for the MR-C4.5-Tree construction. In the future work, the confirmation of termination conditions may be a direction for further study, or any other variations can be presented based on the MR-C4.5-Tree learning algorithm.

ACKNOWLEDGEMENTS

This work is supported by the Natural Science Foundation of China under grant 61673082 and 61533005.

REFERENCES

1. Wu X, Fan W, Peng J, Zhang K, Yu Y. Iterative sampling based frequent itemset mining for big data. *Int J Mach Learn and Cybern*. 2015;6(6):875–882.
2. Sagiroglu Seref, Sinanc Duygu. Big data: a review. *Collaboration Technologies and Systems (cts), 2013 International Conference on*. IEEE, San Diego, California, USA; 2013:42–47.
3. Zaslavsky A, Perera C, Georgakopoulos D. Sensing as a service and big data. arxiv preprint arxiv:1301.0159; 2013.
4. Suthaharan Shan. Big data classification: problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Perform Eval Rev*. 2014;41(4):70–73.
5. Jadhav DK. Big data: the new challenges in data mining. *Int J Innovative Res in Comput Sci at Technol*. 2013;1(2):39–42.
6. Wang R, He Y-L, Chow C-Y, Ou Fa-F, Zhang Jian. Learning elm-tree from big data based on uncertainty reduction. *Fuzzy Sets and Systems*. 2015;258:79–100.
7. Yadav C, Wang S, Kumar Manoj. Algorithm and approaches to handle large data-a survey. arxiv preprint arxiv:1307.5437; 2013.
8. Pakize SR, Gandomi A. Comparative study of classification algorithms based on mapreduce model. *Int J of innovative Res in Adv Eng, ISSN*. 2014;1(7):2349–2163.
9. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. *Commun ACM*. 2008;51(1):107–113.
10. Quinlan JR. Induction of decision trees. *Mach learning*. 1986;1(1):81–106.
11. Shafer J, Agrawal R, Mehta M. Sprint: a scalable parallel classifier for data mining. *Proceedings 1996 International Conference very large data bases*. Citeseer, Mumbai (Bombay), India; 1996:544–555.
12. Joshi MV, Karypis G, Kumar V. Scalparc: a new scalable and efficient parallel classification algorithm for mining large datasets. *Parallel Processing Symposium, 1998. ipps/spdp 1998. Proceedings of the first Merged International … and Symposium on Parallel and dDstributed processing 1998*. IEEE, Orlando, Florida, USA; 1998:573–579.
13. Srivastava A, Han E-H, Kumar V, Singh V. *Parallel Formulations of Decision-tree Classification Algorithms*. Bonn Germany: Springer; 2002.
14. Sheng Y, Phoha VV, Rovnyak SM. A parallel decision tree-based method for user authentication based on keystroke patterns. *Syst, Man, and Cybern, Part B: Cybern, IEEE Transactions on*. 2005;35(4):826–833.
15. Panda B, Herbach JS, Basu S, Bayardo RJ. Planet: massively parallel learning of tree ensembles with mapreduce. *Proceedings of the VLDB Endowment*. 2009;2(2):1426–1437.
16. Ben-Haim Ys, Tom-Tov E. A streaming parallel decision tree algorithm. *The J of Mach Learn Res*. 2010;11:849–872.
17. Yin W, Simmhan Y, Prasanna VK. Scalable regression tree learning on hadoop using openplanet. *Proceedings of Third International Workshop on Mapreduce and its Applications ate*. ACM, Delft, Netherlands; 2012:57–64.
18. Calistru IT, Cotofrei P, Stoffel K. A parallel approach for decision trees learning from big data streams. *Business information systems*. Bonn Germany: Springer; 2015:3–15.
19. Apache Hadoop. Available online: http://hadoop.apache.org/ Accessed May 1, 2014.
20. Quinlan JR. *C4.5: Programs for Machine Learning*. San Francisco, California, USA: Morgan Kaufmann Publishers Inc.; 1993.
21. Quinlan JR. Improved use of continuous attributes in c4.5. *J of Artificial Intelligence Res*. 1996:77–90.
22. White T. *Hadoop: The Definitive Guide, 4th Edition*. Sebastopol, California, USA: O'Reilly Media; 2015.

23. Wu X, Kumar V, Quinlan JR, et al. Top 10 algorithms in data mining. *Knowledge and information systems*. 2008;14(1):1–37.
24. MOA (Massive Online Analysis). Available online: http://moa.cs.waikato.ac.nz/datasets/ Accessed May 1, 2014.
25. UCI Machine Learning Repository. Available online: http://archive.ics.uci.edu/ml/. Accessed May 1, 2014.
26. WEKA. Available online: http://www.cs.waikato.ac.nz/ml/weka/ Accessed May 1, 2014.
27. He Q, Shang T, Zhuang F, Shi Z. Parallel extreme learning machine for regression based on mapreduce. *Neurocomputing*. 2013;102:52–58.

How to cite this article: Mu Y, Liu X, Yang Z, Liu X. A parallel C4.5 decision tree algorithm based on MapReduce. *Concurrency Computat: Pract Exper*. 2017;29:e4015. https://doi.org/10.1002/cpe.4015