Received March 15, 2020, accepted March 29, 2020, date of publication April 2, 2020, date of current version April 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984778

A Novel Configuration Tuning Method Based on Feature Selection for Hadoop MapReduce

JUN LIU[1,2], SULE TANG[1], GUANGXIA XU[1,2], CHUANG MA[1,2], AND MINGWEI LIN[3]
[1]School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2]Chongqing Key Laboratory of Cyberspace and Information Security, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[3]College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China

Corresponding author: Jun Liu (liujuncqcs@163.com)

This work was supported in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2017jcyjAX0270, Grant cstc2017jcyjAX0071, and Grant cstc2018jcyjA0672, in part by the Major Programs of Chong Qing Science and Technology Commission under Grant cstc2018jszx-cyztzxX0011, and in part by the National Natural Science Foundation of China under Grant 61772099 and Grant 61572090.

ABSTRACT Configuration parameter optimization is an important means of improving the performance of the MapReduce model. The existing parameter tuning methods usually optimize all configuration parameters in MapReduce. However, it is exceedingly challenging to tune all the parameters for the MapReduce model because there are massive configuration parameters in MapReduce. In this paper, a novel configuration parameter tuning method based on a feature selection algorithm is proposed, and it is composed of the feature selection objective function and feature selection process. The objective function is based on the kernel clustering algorithm, in which anisotropic Gaussian kernel is adopted instead of the traditional Gaussian kernel to accurately judge the importance of each parameter in MapReduce. Then, the relationship between the configuration parameters in MapReduce and the features in the feature selection algorithm is defined. Moreover, the importance of each parameter is reflected by the kernel width of anisotropic Gaussian kernels. At the same time, the method of gradient descent is introduced to update the kernel width and control the feature selection process of the iterative algorithm. Finally, experimental results show that the proposed algorithm performs suitably for the MapReduce model.

INDEX TERMS Parameter tuning, Hadoop MapReduce, kernel function, K-means.

I. INTRODUCTION

In recent decades, the scale of data in various fields has grown rapidly. High-performance computing and distributed data processing technology are widely used in the data analysis of various fields. Hadoop [1], Spark [2], Storm [3], and MARP [33] are currently popular platforms using distributed processing technology. MPI and Hadoop are two widely used parallel model [33]. Compared with MPI, Hadoop based on the MapReduce model is still used by many fields, such as industry, scientific computing, and bioinformatics, because of its advantages of scalable, efficient and fault-tolerant [4], [5], [30], [33]. However, with the increasing demand for data processing, the efficiency problems of jobs in MapReduce have become substantially more complex. There are often many factors that influence the efficiency problems simultaneously. These factors include massive configuration parameters, an inefficient task scheduler, inefficient data locality, the absence of load balancing and unreasonable allocation of resource slots etc., [33]. These factors lead to inefficiencies for jobs in Hadoop. In these factors, massive configuration parameters are the primary problem, because the task scheduler, data locality, copy placement and other optimizations need to be based on reasonable configuration parameters [1], [6].

Parameter tuning is a very complicated engineering problem. Generally, configuration parameter tuning is employed to optimize the combination of various configuration parameters of a system or model to achieve better performance. There are more than 190 configuration parameters in MapReduce [7]. It includes I/O management, slot source allocation, memory management, concurrency, map and reduce configuration etc., [8], [9]. It is hard for a platform administrator to fully understand and correctly configure these configuration parameters, because it is an NP- problem to exactly and correctly configure more than 190 configuration parameters to achieve optimal performance for MapReduce.

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Liu.

At present, the methods of configuration parameter optimization for MapReduce mainly include the combination of configuration parameters, and parameter optimization methods based on simulators, experience principles, and machine learning [10], [11], [14], [16]. In the process of parameter optimization, these methods take all parameter into account. The combination of all configuration parameters results in a high complexity of the algorithm and the effectiveness is low. One of the main reasons for these problems are that the current studies ignore the fact that the various configuration parameters in MapReduce are of varying importance. Although there are more than 190 configuration parameters in MapReduce, the configuration parameters have a different influence on the execution time of MapReduce jobs. Some configuration parameters can directly affect the execution time of MapReduce. The other configuration parameters do not have a significant impact on the execution time of MapReduce. Moreover, Zou *et al*. [34] and Zeng *et al.* [35] has achieved good results in the field of bioinformatics data using feature ranking algorithms.

Therefore, to effectively improve the optimization of MapReduce configuration parameters, the paper presents a configuration parameter optimization method based on feature selection.

1) The foundation principle of MapReduce is analyzed, which shows that the importance of each configuration parameter is different, and the influence degree of each configuration parameter on the execution time of MapReduce is different.

2) This paper presents a clustering feature selection algorithm (IK-means for short) based on the kernel function penalty, which solves the problem that platform management personnel encounter due to the difficult of configuring the excessive configuration parameters in MapReduce. In IK-means, to accurately judge the influence degree of each feature parameter, the anisotropic Gaussian kernel function is introduced instead of the traditional Gaussian kernel function, and the importance of each feature is reflected by the parameters (also known as kernel width) in different directions of the anisotropic Gaussian kernel function.

3) The method of gradient descent is proposed to minimize the kernel width vectors of the anisotropic Gaussian kernel, so that the clustering effect of the selected features can be closest to the clustering effect of the original features, to achieve the purpose of feature selection.

The paper is organized as follows. Section II describes existing parameter tuning algorithms in Hadoop. In section III, the importance analysis of the configuration parameters is presented. Section IV introduces the feature selection algorithm based on the kernel function. Section V provides an empirical evaluation of the developed methods.

II. RELATED WORK

At present, the method to tune the configuration parameters of MapReduce mainly focuses on the software-defined network, machine learning, and simulator method.

The software-defined network [1] divides the jobs in Hadoop into CPU- or I/O- intensive, and the interrelations among Hadoop parameters are represented by the constructed fitness function. The construction of the fitness function is based on the job samples that can be considered as CPU- or I/O- intensive in Hadoop. Then, a genetic algorithm is introduced to optimize the configuration parameters by genetic programming in Hadoop. Moreover, the software-defined network [1] takes into account the tuning of some significant configuration parameters. However, how to obtain these significant configuration parameters is also a problem.

H-Tune [6] is a parameter optimization algorithm based on machine learning. The main idea of H-Tune is to predict the performance of an application with a given configuration rather than to execute the application in Hadoop. A performance profiler is first introduced to collect and analyze the performance data of the application. Then, the execution predictor is introduced to predict the performance of an application with a given configuration on Hadoop. In addition, two-level fusion model training is introduced to search for an optimal configuration for an application in MapReduce. However, it takes too much time to collect profiling data, and it is very difficult to build a model for the all applications in MapReduce.

The genetic algorithm [14] is a popular algorithm in parameter optimization. MSET [9] is a typical parameter optimization algorithm based on the genetic algorithm. A framework is first introduced to reduce the collection time of profiling data for the applications with a given configuration in Hadoop. In addition, the MT algorithm is leveraged to predict the running time of the applications with different configurations. Moreover, an MT-driven genetic algorithm is introduced to search the parameter space to optimize the configuration. DAC [10] also uses a genetic algorithm to search for suitable configuration parameters based on the running time of the applications. However, DAC can only deal with limited configuration parameters. BestConfig [12] uses a bound-and-search algorithm to optimize configurations. However, it is only suitable for fixed applications.

SMBSP [13] is a self-tuning method of parameters based on an artificial neural network. SMBSP consists of three layers, which are the input layer X, hidden layer H, and output layer Y. The main idea of the SMBSP is that different activation functions are used in different layer. The ReLU [13] activation function is introduced in the input and hidden layer. The softmax activation function is used in the output layer. The relationship of the three layers is defined as $H = X_1w_1 + X_2w_2 + b$. w *and* b represent the weights and bias, respectively. The self-tuning of parameters is based on the H. Obviously, H is a linear model. The relationship among the three layers is not expressed accurately because the relationship between the data is likely nonlinear [14].

HSim [11] is a typical MapReduce simulator. HSim makes improvements in the system state (processor, memory, buffer, disk, network and I/O state). It can accurately simulate the behavior of the Hadoop platform and reflect the parameters of system state dynamic transformation. In HSim, some benchmark programs for validation are first generated in the environment of optimizing the parameter configuration. In addition, jobs for simulation are read into HSim as data streams and the heartbeat mechanism is used as a comparison with the benchmark program in the Hadoop cluster. Finally, a decision regarding the efficiency of the parameter configuration can be made during the comparison process. However, for the convenience of simulation, a configuration parameter for HSim has been added to the simulator. As a result, parameters are added to MapReduce.

Shivnath *et al.* [15] introduce a competitive optimization [31] to run work with multiple instances, each instance running on the Hadoop platform with different parameter configurations. The parameter optimization of this method is directly implemented by the Hadoop platform, which reduces the workload of users. However, there are different stages on the Hadoop platform, and the operation efficiency is different in each stage. Therefore, there are some limitations to quickly judge the accuracy of a best example.

Bu *et al.* [16] adopt reinforcement learning (RL) method to automatically optimize the configuration parameters in the MapReduce model. To optimize the configuration parameters, a Markov decision process (MDP) is introduced. All parameter configuration situations are defined as a state space in MDP, and n parameters for the state space are considered to be a set of states consisting of n vectors, while the behavior in MDP is defined as three kinds: increase, decrease, and associate with other parameters. The least square method is introduced to optimize the parameter combination [8]. However, the combined parameters sometimes do not allow the job to run most efficiently.

As described in [1], [13], [16]–[18], many parameters will greatly increase the time of training, and the efficiency of the optimization process is also limited. Therefore, selecting some important parameters that can directly affect the efficiency of job execution is a more reliable method for parameter tuning in Hadoop MapReduce.

III. IMPORTANCE ANALYSIS OF MAPREDUCE CONFIGURATION PARAMETERS

There are more than 190 configuration parameters of different importance in MapReduce, and it is difficult for platform administrators to configure these configuration parameters. The main configuration parameters in MapReduce are listed in Table 1 [1], [6], [18]. These configuration parameters mainly include I/O management, slot resource allocation, memory management, concurrency, map and reduce configuration, and so on. The importance of related configuration parameters in [19]–[21], and [22] on concurrency, memory management, concurrency, map and reduce is analyzed carefully and verified by experiments. Liu *et al.* [11] uses the simulator method, in which the importance of the

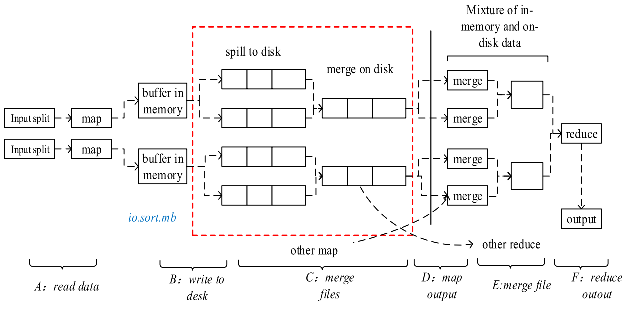


FIGURE 1. **The fundamental principle of MapReduce.**

TABLE 1. **The main parameter of Hadoop.**

parameters	Default
mapreduce.task.io.sort.mb	120
io.sort.record.percent	0.05
mapreduce.map.sort.spill.percent	0.8
mapreduce.task.io.sort.factor	10
io.file.buffer.size	4096
mapreduce.job.combine.class	NULL
mapreduce.map.combine.minspills	3
mapreduce.map.output.compress	FALSE
mapreduce.job.reduces	1
mapreduce.job.maps	2
mapreduce.output.fileoutputformat.compress	FALSE
mapreduce.job.reduce.slowstart.completedmaps	0.05
mapreduce.reduce.shuffle.input.buffer.percent	0.66
mapreduce.reduce.input.buffer.percent	0
mapreduce.map.maxattempts	4
mapreduce.tasktracker.map.tasks.maximum	2
mapreduce.tasktracker.reduce.tasks.maximum	2
mapred.child.java.opts	200
mapreduce.reduce.shuffle.parallelcopies	5
mapreduce.reduce.shuffle.merge.percent	0.66
dfs.blocksize	64
dfs.replication	3

above configuration parameters is illustrated by simulating the execution process of the map and reduce phases of the task in MapReduce. Moreover, the importance of I/O related configuration parameters is also illustrated by the simulator.

In addition, the importance of configuration parameters is also proved by a mathematical model [23]. First, the fundamental principles of MapReduce are analyzed, and the picture of fundamental principles is shown in Figure 1. As seen from Figure 1, the execution process of jobs in MapReduce is divided into six processes: reading data, writing to disk, merging files, map output, file merge and reduce output. Second, the influence of each parameter for job execution performance in different stages is analyzed and the role of I/O configuration parameters in these processes is explained. For example, the input split parameter in the map process, the mapreduce.task.io.sort.mb parameter and the mapreduce.task.io.sort.factor parameter in the file merge process, etc., are analyzed.

The above analysis shows that the configuration parameters can affect the efficiency of job execution in each process, which is the motivation of this paper.

IV. CLUSTERING FEATURE SELECTION BASED ON KERNEL FUNCTION PENALTY

In this section, a clustering feature selection algorithm (IK-means for short) based on the kernel function penalty is introduced, in which the configuration parameters with a large influence on the run time for jobs can be selected. The selected parameters can be used by administrators for tuning configuration in Hadoop, that is, it can avoid the situation in which administrators need to understand and configure more than 190 configuration parameters in Hadoop.

The IK-means algorithm is mainly to establish a clustering objective function. The function of this objective function is to delete the unimportant features when the clustering algorithm is running. The deletion features are the corresponding configuration parameters in Hadoop. To ensure the accuracy of feature selection, the objective function takes into account three aspects. The first is the choice of the kernel function. The second is the definition of the objective function. The third is the establishment of the feature penalty function. Assume that the input sample of the algorithm is n records. Each record records the job types and the configuration parameters when it executes in Hadoop. The job types can be divided into CPU-intensive, memory-intensive, I/O-intensive, and network-intensive. The values in all records are assumed to have been normalized. In the process of clustering, the features that have a great influence on the clustering of the job type will be selected, that is, the configuration parameters that have a great influence on clustering are considered as important parameters, because they can affect the parameters of the job type and the execution time of the job.

The three steps to establish the objective function are as follows.

A. THE SELECTION OF KERNEL FUNCTION

The traditional kernel function is controlled by a kernel width, when the features are mapped to higher dimensions, that is, the mapping of features to different dimensions is controlled by the same kernel width. The anisotropic Gaussian kernels (ANGKs) have a different kernel width in different dimensions, that is, a kernel width in a dimension corresponds to a feature. Therefore, anisotropic kernels can more accurately reflect the importance of different features [24]–[27]. The method has been proven in the feature selection of the support vector machine [24], and its effect is superior to the traditional Gaussian kernel-based feature selection. Therefore, anisotropic Gaussian kernel is introduced in the IK-means algorithm, in which each kernel width corresponds to a configuration parameter in Hadoop MapReduce.

The anisotropic Gaussian kernel is defined as:

$$K(x_i, x_s) = \exp(-\sum\nolimits_{j=1}^{n} \frac{(X_{ij}-X_{sj})^2}{2\sigma_j^2}) \tag{1}$$

where x_i and x_s are samples. The shape of the kernel function is controlled by the parameter σ. σ is defined as a vector containing n kernel width parameters, where n is the characteristic dimension of each sample. σ is defined as $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_n]$, where each reciprocal of σ_j can reflect the importance of corresponding features [24]. Therefore, in this paper, σ_j with a large value can be deleted, the large value corresponds to a configuration parameter in MapReduce. Therefore, the kernel width vector v in IK-means is defined as:

$$\mathrm{v} = [\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \frac{1}{\sigma_3}, \cdots, \frac{1}{\sigma_\mathrm{n}}] \tag{2}$$

Therefore, the kernel function is defined in (3), as shown at the bottom of this page, where $a * b = (a_1b_1, a_2b_2, \ldots\ldots, a_nb_n)$, v is also known as the kernel width vector of an anisotropic Gaussian kernel.

B. FEATURE PENALTY FUNCTION

The l_0 "norm" approximation can use some characteristics of the sample to express the characteristics of the original characteristics of the sample [104]. l_0 "norm" approximation has been validated in the feature selection algorithm based on the support vector machine (SVM) [24]. Assume that the l_0 "norm" is defined as $||w||_0$, and $||w||_0$ can be approximately expressed with a concave function, which is $||w||_0 \approx e^T$(e-exp(-$r|w|$). Suppose v is an n-dimensional sample; then $e^Tv < n$ [28], where the v_j is positive because it describes the kernel width in the anisotropic Gaussian kernel, and the value of the kernel width is always a non-negative number. Therefore, the following feature penalty function is defined as:

$$\mathrm{f}\,(\mathrm{v}) = e^T\,(e - \exp\,(-\gamma V)) = \sum\nolimits_{j=1}^{n} (1 - \exp(-\gamma V_j)) \tag{4}$$

$$\begin{aligned}
\mathrm{K}\,(X_i, X_s, v) &= \exp\left[-\sum_{j=1}^{n} \frac{(X_{ij}-X_{sj})^2}{2\sigma_j^2}\right] \\
&= \exp[-\sum_{j=1}^{n} \frac{(X_{ij}-X_{sj})^2}{2{}^1/{v_j}^2}] \\
&= \exp[-\frac{(V_1X_{i1}-V_1X_{s1})^2 + (V_2X_{i2}-V_2X_{s2})^2 \ldots + (V_nX_{in}-V_nX_{sn})^2}{2}] \\
&= \exp[-\frac{||V*X_i - V*X_s||^2}{2}]
\end{aligned} \tag{3}$$

According to the conclusion of [24], [27], [28], the value of γ is set to 5 so that the range of adaptation is the widest and the effect is better, so this paper also sets the value of γ to 5.

C. THE ESTABLISHMENT OF THE OBJECTIVE FUNCTION

In the kernel K-means algorithm, assuming that there are K class samples, selecting the best class for the sample x_i, x_i satisfies formula 5:

$$\| \Phi(X_i) - U_W) \|^2 \leq \| \Phi(X_i) - U_{W'})\|^2, \quad 1 \leq \mathrm{i} \leq N, 1 \leq w' \leq K, w \neq w' \qquad (5)$$

which describes the distance from $\Phi(x_i)$ to class u_w less than $u_{w'}$ in any other cluster center, and N is the number of samples. x_i represents the i sample, and is consistent with the definition of the nuclear K-means.

Effective clustering algorithms need to make the sample that is closest to the center point belong to the same class, and the distance between different classes is far. Moreover, the clustering algorithm needs to reduce the repeated selection consumption in different classes [29], that is, sample x_i belongs to the c_i class in the T iteration, however it belongs to the $c_i + 1$ class in $t + 1$ iteration. Therefore, the algorithm needs to make the distance from sample x_i to the cluster center as small as possible at the t iteration. Although Maldonado *et al.* [27] proposes the objective function based on the kernel function, it ignores the case in which the sample is selected back and forth between the two classes. Therefore, this paper improves the algorithm of [27] and introduces the following cost formula:

$$\begin{aligned}\text{Cost}\,(\text{c}) &= \sum\nolimits_{w=1}^{k} \sum\nolimits_{i\in C_w} \sum\nolimits_{w\neq w'} \frac{\| \Phi(X_i - u_w) \|^2}{\| \Phi(X_i - u_{w'}) \|^2} \\ &+ \sum\nolimits_{t=1}^{T} \sum\nolimits_{X_j\in X} \| u_{x_j}^{t} - u_{x_j}^{t-1} \|^2 \end{aligned} \qquad (6)$$

where $X = \{x_1, x_2, x_3, \ldots, x_n\}$ is a sample set, the clustering set of all sample points is C, that is, the clusters are divided into $C = \{C_1, C_2, C_3, \ldots, C_k\}$, the number of clusters is k, u_w is the cluster center of the C_w class, and t is the number of iterations. u_{xj} is the clustering center of the t-*th* iteration of the sample x_j, and u_{xj} is the clustering center of the (t-1)-*th* iteration of the sample x_j. According to the definition of kernel K-means and formula (6) the cost function is expressed as:

$$\text{Cost}\,(C) = \sum\nolimits_{W=1}^{K} \sum\nolimits_{i\in C_W} \sum\nolimits_{W\neq W'} \frac{Q_{v,W}}{Q_{v,W'}} + Q \qquad (7)$$

where

$$\begin{aligned} Q_{v,W} &= K\,(X_i, X_i, v) - \frac{2}{N_{C_W}} \sum\nolimits_{j\in C_W} K\left(X_i, X_j, v\right) \\ &+ \frac{1}{N_{C_W}^2} \sum\nolimits_{j\in C_W p\in C_W} K\left(X_j, X_p, v\right) \\ \text{Q} &= \sum_{t=1}^{T} \sum_{X_j\in X} \| u_{x_j}^{t} - u_{x_j}^{t-1} \|^2 \end{aligned} \qquad (8)$$

Q is the cost of the sample being repeatedly selected in different classes. The target function minimizes this cost, that is, the distance between the cluster center of sample x_j at t-time and $t - 1$ time is as small as possible.

Therefore, the target function minimizes formula (6) by punishing the kernel width vector v, and deletes the unimportant kernel width (corresponding features) of the kernel width vector v by the penalty function, such that the selected features are as close as possible to the clustering effect of the original features. In addition, the target function follows the principle that the sample in the class is closest to the sample, and the sample is far away between the classes, while minimizing the cost of the sample being repeatedly selected in different classes. Therefore, the following minimization objective functions is introduced:

$$\min_{v} F\,(v, C) = \sum_{W=1}^{K} \sum_{i\in C_W} \sum_{W\neq W'} \frac{Q_{v,W}}{Q_{v,W'}} + Q + \lambda f\,(v), \quad v_i \geq 0, \forall i \in \{1, \ldots, N\} \qquad (9)$$

The parameter λ is a predefined parameter used to penalize formula $f\,(v)$ and the cost function.

Algorithm 1 is an algorithm for feature selection in the IK-means process, in which feature deletion and updating of the kernel width are required. The algorithm uses a gradient descent algorithm to minimize the kernel width vector in the iterative process, $v = \{1/\sigma_1, 1/\sigma_2, 1/\sigma_3, \ldots, 1/\sigma_n\}$ updating the kernel width vector v in each iteration, and deleting the unimportant feature v_j.

Algorithm 1 Feature Delection and Kernel Width Updates

```
1. Initialization $v = v_0 e$
2. EndFlag=false; t=0;
3. repeat
4.    C = kernel k-means clusting
5.    $v^t = v^{t-1} - \gamma \nabla F(v^{t-1}, C)$
6.    for each $v_l \in v$ do
7.        if ($v_l < M$)
8.          $v_l = 0$;
9.        end if
10.   end for
11.   if ($v^t = v^{t-1}$)
12.       EndFlag=True
13.   end if
14. until (True==EndFlag)
```

The kernel width vector of an anisotropic Gaussian kernel is adjusted by the gradient descent algorithm in Algorithm 1. For a clustering C (which can be generated by a kernel clustering algorithm through training sample data), for features l, the gradient descent function is as follows:

$$\nabla_l F\,(v, C) = \sum_{w=1}^{k} \sum_{i\in Cw} \sum_{w\neq w'} \nabla_l \frac{Q_{v,w}}{Q_{v,w'}} + \nabla_l Q + \lambda \nabla_l f\,(v) \qquad (10)$$

where

$$\nabla_l \frac{Q_{v,w}}{Q_{v,w'}} = \frac{\nabla_l Q_{v,w} \cdot Q_{v,w'} - Q_{v,w} \cdot \nabla_l Q_{v,w'}}{Q_{v,w'}^2}$$

V. EXPERIMENT AND ANALYSIS

To investigate the effectiveness of the proposed IK-means algorithm, it is compared with the existing typical feature selection algorithms, which are KPKM [27], IRFE [22], and uEFS [32] algorithms in the Hadoop platform.

The Hadoop platform version used in the experiment is Hadoop 2.7.3, which has more than 190 configuration parameters and many types of parameters. Before the experiment, all test parameters were standardized as floating point numbers. The sample for the test is composed of 2000 job records. Each record includes numeric values for the job type and, different configuration parameters, and all records are generated by different types of jobs under the same Hadoop platform by changing the numeric values of each configuration parameter. In these 2000 records, the types of jobs are divided into CPU-intensive, I/O-intensive, memory-intensive, and network-intensive types. In the process of IK-means clustering, the features that have an important influence on the job type are selected, that is, the configuration parameters that have a great influence on the clustering are considered valid parameters in this paper because the parameters that can have an influence on the work category can also have an influence on the execution time of the work.

The experimental method executes feature selection experiments four times employing the four feature selection algorithms, which are IK-means, KPKM, IRFE and uEFS. The first experiment is to select 10 configuration parameters from Hadoop's configuration parameters. The second experiment is to select 8 configuration parameters, the third experiment is to select 5 configuration parameters, and the fourth experiment is to select 3 configuration parameters. Tables 2, 3, 4 and 5 are the results of four sets of experiments, respectively.

The importance of some parameters in Table 2, Table 3, Table 4, and Table 5 is illustrated in [10], [11], [21], which are as follows: mapreduce.task.io.sort.mb, dfs.blocksize, mapreduce.map.sort.spill.percent, mapreduce.task.io.sort. fact or, mapreduce.map.output.compress, dfs.namenode. handle.co unt,and mapreduce.map.combine.minspills. Therefore, these configuration parameters will not be validated in this paper. This paper will verify the importance of the remaining parameters, which are: mapreduce.job. maps, fs.df.interval, mapreduce.job.reduces, mapred.child. java.opts, mapreduc e.job.reduce.slowstart.completedmaps, mapreduce. tasktracker.healthchecker.script.timeout, mapreduce.tasktracker.map.ta sks.maximum,mapreduce.reduce. shuffle. parallelcopies, mapreduce.task.timeout,mapreduce. tasktracker.reduce.tasks.maximum, dfs.datanode.failed. volumes. tolerated, dfs.namenode. checkpoint.period, and dfs.namenode.replication.interval.

The test method is to manually modify the value of the parameters to determine whether it will influence the execution time of the jobs. The value of the test parameter is modified when a job is running. If the change of the value has a great impact on the execution time of the job, the selected configuration parameters are of high importance.

The experimental method is that the sorting programs, of which the data sizes are of 1 GB, 2 GB, and 4 GB, are executed on a Hadoop platform with 10 nodes, and the values of the tested configuration parameters are modified. If the

TABLE 2. Select 10 parameters from Hadoop.

Algorithm	Selected parameters
IK-means	*dfs.namenode.handler.count*; *mapred.child.java.opts*; *mapreduce.job.maps*; *mapreduce.job.reduces*; *dfs.blocksize*; *mapreduce.map.output.compress*; *mapreduce.tasktracker.reduce.tasks.maximum;* *mapreduce.map.sort.spill.percent*; *mapreduce.task.io.sort.factor*; ***fs.df.interval***
IFRE	*mapreduce.job.reduces*; *dfs.blocksize*; *mapreduce.map.output.compress*; *mapreduce.task.io.sort.mb*; ***mapreduce.task.timeout***; *mapreduce.map.sort.spill.percent*; *mapreduce.task.io.sort.factor*; *io.file.buffer.size*; ***fs.df.interval*; *dfs.datanode.failed.volumes.tolerated*;**
uEFS	*dfs.namenode.handler.count; mapred.child.java.opts; dfs.blocksize; mapreduce.job.maps; mapreduce.job.reduces;mapreduce.task.io.sort.mb; mapreduce.map.output.compress; mapreduce.map.sort.spill.percent; mapreduce.task.io.sort.factor; dfs.namenode.replication.interval;*
KPKM	*mapreduce.job.maps*; *mapreduce.job.reduces*; *mapreduce.map.output.compress*; *mapreduce.task.io.sort.mb*; *mapreduce.map.sort.spill.percent*; *mapreduce.task.io.sort.factor*; *mapreduce.map.combine.minspills*; *mapreduce.job.reduce.slowstart.completedmaps*; ***mapreduce.tasktracker.healthchecker.script.timeout; dfs.namenode.checkpoint.period***

TABLE 3. Select 8 parameters from Hadoop.

Algorithm	Selected parameters
IK-means	*dfs.namenode.handler.count*; *mapred.child.java.opts*; *mapreduce.job.maps*; *mapreduce.task.io.sort.mb*; *mapreduce.job.reduces*; *dfs.blocksize*; *mapreduce.map.output.compress*; *mapreduce.map.sort.spill.percent*
IFRE	*mapreduce.job.reduces*; *dfs.blocksize*; *mapreduce.map.output.compress*; *mapreduce.task.io.sort.mb*; *mapreduce.job.maps* *mapreduce.map.sort.spill.percent*; ***dfs.datanode.failed.volumes.tolerated; mapreduce.task.timeout***
uEFS	*mapreduce.job.maps*; *mapreduce.task.io.sort.mb*; *mapreduce.tasktracker.map.tasks.maximum*; *mapreduce.job.reduces;mapreduce.task.io.sort.factor; mapreduce.map.output.compress;* *mapreduce.map.sort.spill.percent*; ***dfs.namenode.replication.interval***
KPKM	*mapreduce.job.maps*; *mapreduce.job.reduces*; *mapreduce.map.output.compress;mapreduce.task.io.sor t.mb*; *mapreduce.map.sort.spill.percent*; *mapreduce.task.io.sort.factor*; *dfs.blocksize* *mapreduce.map.combine.minspills*; ***fs.df.interval***

TABLE 4. Select 5 parameters from Hadoop.

Algorithm	Selected parameters
IK-means	*mapred.child.java.opts*; *mapreduce.job.maps*; *mapreduce.job.reduces*; *dfs.blocksize*; *mapreduce.task.io.sort.mb*
IFRE	*mapreduce.job.reduces*; *dfs.blocksize*; *mapreduce.map.output.compress*; *mapreduce.task.io.sort.mb*; ***dfs.namenode.replication.interval***
uEFS	*mapreduce.job.maps*; *mapreduce.job.reduces*; *mapred.tasktracker.map.tasks.maximum*; *mapreduce.task.io.sort.mb*; *mapreduce.task.io.sort.factor*
KPKM	*mapreduce.job.maps*; *mapreduce.job.reduces*; *mapreduce.map.output.compress*; *mapreduce.task.io.sort.factor*; *mapreduce.map.combine.minspills*

TABLE 5. Select 3 parameters from Hadoop.

Algorithm	Selected parameters
IK-means	*mapred.child.java.opts*; *mapreduce.job.maps*; *mapreduce.job.reduces*
IFRE	*mapreduce.job.reduces*; *mapreduce.map.output.compress*; *mapreduce.task.io.sort.mb*
uEFS	*mapreduce.job.maps*; *mapreduce.job.reduces*; *mapreduce.task.io.sort.mb*
KPKM	*mapreduce.job.maps*; *mapreduce.job.reduces*; *mapreduce.map.combine.minspills*

execution time of the job has a great change, the parameter is considered to be an important configuration parameter, and the configuration parameter is also considered to be a valid feature selected by the test algorithm. In contrast, if the execution time of the job has little effect, the parameter is considered to be an unimportant configuration parameter, and the configuration parameter is also considered to be an invalid feature selected by the test algorithm. Figure 2 and Figure 3 show the test results of the above unverified configuration parameters. As can be seen from Figure 2, as the value of the configuration parameters changes, the execution time of the job changes significantly, that is, the curve in the graph has obvious fluctuations, and each configuration parameter has a value that minimizes the running time of the work, so the configuration parameters in the figure are considered to be valid features. In Figure 3, it can be seen that changing the value of each configuration parameter has little effect on the execution time of the work, and the curve in the figure has no significant fluctuation, so the configuration parameters in the figure are considered to be invalid features. The ordinates in both figures represent the execution time of the work (in seconds). The abscissa indicates the actual value of each configuration parameter. The abscissa of the first figure in Figure 2 indicates that the configuration parameter mapreduce.job.maps has specific values of 2, 4, 6, ..., 30 when the work execution time changes. The abscissa of the first graph in Figure 3 indicates that the configuration parameter fs.df.interval has a specific value of 1, 2, 3, ..., 10 seconds (in seconds) when the work execution time changes.

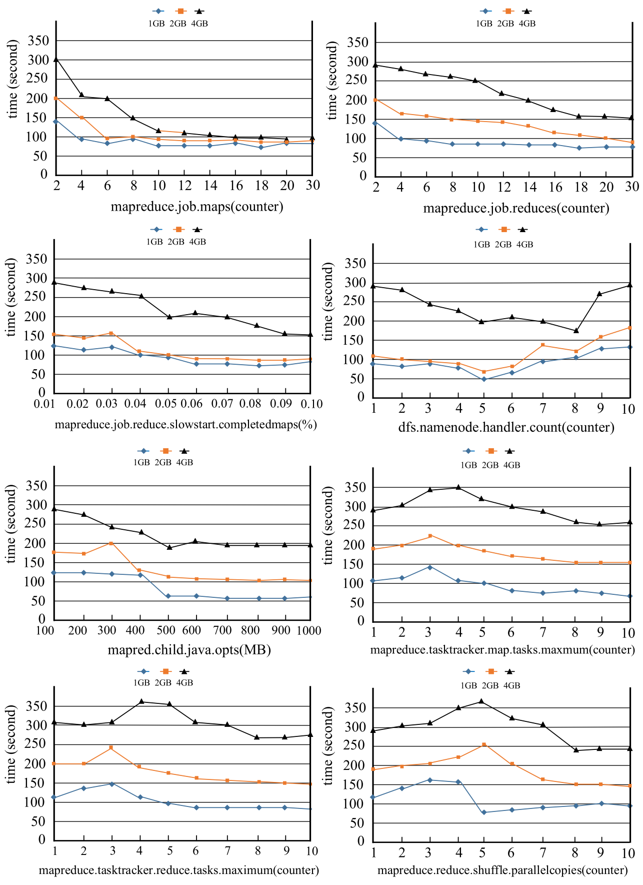


FIGURE 2. The influence of the job running time of the selection parameters (effective features).

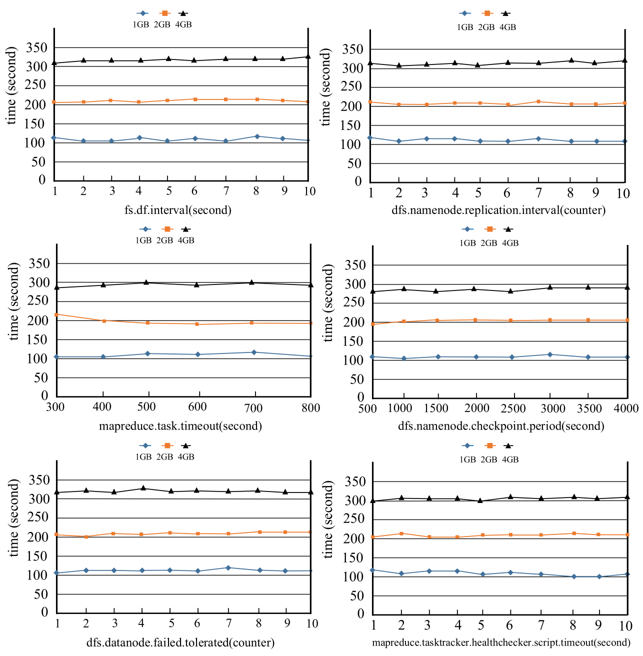


FIGURE 3. The influence of the job running time of the selection parameters (invalid features).

Therefore, through the above analysis, it is considered that invalid features are obtained when the configuration parameters in Figure 3 have a low impact on the job execution time, (corresponding to the bold italic configuration parameters in Table 2, Table 3, Table 4, and Table 5).

In the following, the effectiveness of each algorithm is analyzed by judging the number of invalid features of the four feature selection algorithms: IK-means, KPKM, IRFE, and uEFS.

The results of the four feature selection experiments from Table 2, Table 3, Table 4 and Table 5 include the four feature selection algorithm: IK-means, KPKM, IRFE, and uEFS. The bold italic configuration parameter in the Table (that is, the configuration parameter in Figure 3) is an invalid configuration parameter, and it means that the configuration parameter has a lower impact on the execution time of the MapReduce job. The parameters shown in normal font are considered to be valid configuration parameters, which means the modification of the values of these configuration parameters have a strong impact on the execution time of the MapReduce job.

The four algorithms, namely, IK-means, IFRE, uEFS, and KPKM, have all selected invalid configuration parameters in the experiment of selecting 10 configuration parameters (Table 2). The IK-means and uEFS algorithms select one invalid configuration parameter. The KPKM algorithm selects two invalid configuration parameters. The IFRE algorithm selects three invalid configuration parameters. In the experiment of selecting 8 configuration parameters (Table 3), the number of selected invalid configuration parameters by each algorithm is lower than the case in which 10 configuration parameters were selected. When the selected configuration parameter is 3 (Table 5), all of the algorithms did not select invalid configuration parameters. It can be seen that the effectiveness of the configuration parameters selected by the IK-means algorithm proposed in this paper is better than that of the three feature selection algorithm comparisons. Among them, the principle of the IRFE feature selection algorithm is to use the Gaussian kernel to select features, which is based on the SVM feature selection algorithm. The IK-means feature selection algorithm proposed in this paper uses the anisotropic Gaussian kernel instead of traditional Gaussian nucleus. Comparing the methods illustrates that with the anisotropic Gaussian kernel, the importance of different features is reflected by different kernels width, while the Gaussian nucleus uses the same nucleus width to reflect the importance of different features. In the paper, the kernel width vector is defined as $\sigma \ =[\sigma_1, \sigma_2, \ldots, \sigma_n]$. σ_i can reflect the importance of corresponding configuration parameter. Each vector in the σ set is different. In Gaussian kernel, Each vector in the σ set is same. Therefore, the accuracy achieved by the anisotropic Gaussian kernel in judging the importance of different features is better than that of the Gaussian nucleus.

Although the KPKM algorithm is also a kernel method, this method does not take into account the features that are selected back and forth among different classes. This situation will reduce the accuracy of important parameter judgments. The uEFS algorithm uses the unified features scoring (UFS) algorithm to generate a final ranked list of features following a comprehensive evaluation of a feature set. Then a threshold value selection algorithm is introduced to select a subset of features. However, the selection of the threshold value can affect the accuracy of the subset of the features.

Compared with the uEFS algorithm, IK-means introduces the gradient descent algorithm to minimize the kernel width vector in the anisotropic Gaussian kernel, in which feature deletion and updating of the kernel width are introduced to minimize the kernel width vector in the iterative process. In each iterative process, kernel width vector is updated, and the unimportant configuration parameter is deleted. So that the selected configuration parameters can represent the original configuration parameters. This is the reason that the IK-means algorithm achieves a better selection on important parameters.

VI. CONCLUSION

Because of the massive configuration parameters, parameter tuning is challenging work in Hadoop MapReduce. In this paper, a novel parameter tuning method based on kernel K-means clustering (IK-means for short) has been put forward to select the important parameters that can directly impact the running time of the jobs in Hadoop rather than tuning all of the parameters. To tune the parameters, the IK-means algorithm process has been presented. The selection of the kernel function, feature penalty function and the establishment of the objective function have also been studied for IK-means. Moreover, to accurately judge the importance of each feature, anisotropic Gaussian kernel is introduced instead of traditional Gaussian kernel. The importance of each feature is reflected by the parameters of anisotropic Gaussian kernel in different directions. At the same time, the method of gradient descent is introduced to select a group of features that are as close to the original features as possible. A series of experiments are conducted and encouraging results are obtained.

REFERENCES

[1] A. Khaleel and H. Al-Raweshidy, "Optimization of computing and networking resources of a Hadoop cluster based on software defined network," *IEEE Access*, vol. 6, pp. 61351–61365, 2018.

[2] M. R. Karim, M. Cochez, O. D. Beyan, C. F. Ahmed, and S. Decker, "Mining maximal frequent patterns in transactional databases and dynamic data streams: A spark-based approach," *Inf. Sci.*, vol. 432, pp. 278–300, Mar. 2018.

[3] M. Hajeer and D. Dasgupta, "Handling big data using a data-aware HDFS and evolutionary clustering technique," *IEEE Trans. Big Data*, vol. 5, no. 2, pp. 134–147, Jun. 2019.

[4] K. Chen, J. Powers, S. Guo, and F. Tian, "CRESP: Towards optimal resource provisioning for MapReduce computing in public clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1403–1412, Jun. 2014.

[5] R. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, and F. Martínez-Álvarez, "Big data time series forecasting based on nearest neighbours distributed computing with spark," *Knowl.-Based Syst.*, vol. 161, pp. 12–25, Dec. 2018.

[6] X. Hua, M. C. Huang, and P. Liu, "Hadoop configuration tuning with ensemble modeling and Metaheuristic optimization," *IEEE Access*, vol. 6, pp. 44161–44174, 2018.

[7] W. Premchaiswadi and W. Romsaiyud, "Optimizing and tuning MapReduce jobs to improve the large-scale data analysis process," *Int. J. Intell. Syst.*, vol. 28, no. 2, pp. 185–200, Feb. 2013.

[8] J. Liu, R. Liu, N. Li, B. Zhu, L. Jiang, and R. Huang, "An novel parameter tuning alogrithm for MapReduce model," in *Proc. 8th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, Beijing, China, Jun. 2018, pp. 266–269.

[9] Z. Bei, Z. Yu, Q. Liu, C. Xu, S. Feng, and S. Song, "MEST: A model-driven efficient searching approach for MapReduce self-tuning," *IEEE Access*, vol. 5, pp. 3580–3593, 2017.

[10] Z. Yu, Z. Bei, and X. Qian, "Datasize-aware high dimensional configurations auto-tuning of in-memory cluster computing," *Proc. 23d Int. Conf. Architectural Support Program. Lang. Operating Syst.*, New York, NY, USA, 2018, pp. 564–577.

[11] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, "HSim: A MapReduce simulator in enabling cloud computing," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 300–308, Jan. 2013.

[12] Y. Zhu, J. Liu, M. Guo, Y. Bao, W. Ma, Z. Liu, K. Song, and Y. Yang, "BestConfig: Tapping the performance potential of systems via automatic configuration tuning," in *Proc. Symp. Cloud Comput. SoCC*, New York, NY, USA, 2017, pp. 338–350.

[13] M. A. Rahman, J. Hossen, and C. Venkataseshaiah, "SMBSP: A self-tuning approach using machine learning to improve performance of spark in big data processing," in *Proc. 7th Int. Conf. Comput. Commun. Eng. (ICCCE)*, Kuala Lumpur, Malaysia, Sep. 2018, pp. 274–278.

[14] M. A. Rahman, J. Hossen, C. Venkataseshaiah, C. K. Ho, K. G. Tan, A. Sultana, M. Z. H. Jesmeen, and F. Hossain, "A survey of machine learning techniques for self-tuning Hadoop performance," *Int. J. Elect. Comput. Eng.*, vol. 8, no. 3, pp. 1854–1862, Jun. 2018.

[15] S. Babu, "Towards automatic optimization of MapReduce programs," in *Proc. 1st ACM Symp. Cloud Comput. SoCC*, Indianapolis, IN, USA, 2010, pp. 137–142.

[16] X. Bu, *Autonomic Management and Performance Optimization for Cloud Computing Services*. Wayne State Univ., 2013.

[17] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, May 2017.

[18] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

[19] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.

[20] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz, "Energy efficiency for large-scale MapReduce workloads with significant interactive analysis," in *Proc. 7th ACM Eur. Conf. Comput. Syst. EuroSys*, Bern, Switzerland, 2012, pp. 43–45.

[21] H. Herodotos, L. Harold, L. Gang, and B. Nedyalko, "Starfish: A self-tuning system for big data Analytics," in *Proc. 5th Biennial Conf. Innov. Data Syst. Res.*, 2011, pp. 261–272.

[22] G. P. Wang, "Research on Key technologies of dependability oriented anomaly detection of virtual machines under cloud environment," Ph.D. dissertation, Chongqing Univ., Chongqing, China, 2015.

[23] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "SCALLA: A platform for scalable one-pass analytics using MapReduce," *ACM Trans. Database Syst.*, vol. 37, no. 4, pp. 1–43, Dec. 2012.

[24] S. Maldonado, R. Weber, and J. Basak, "Simultaneous feature selection and classification using kernel-penalized support vector machines," *Inf. Sci.*, vol. 181, no. 1, pp. 115–128, Jan. 2011.

[25] C. Lopez-Molina, G. Vidal-Diez de Ulzurrun, J. M. Baetens, J. Van den Bulcke, and B. De Baets, "Unsupervised ridge detection using second order anisotropic Gaussian kernels," *Signal Process.*, vol. 116, pp. 55–67, Nov. 2015.

[26] P.-L. Shui and W.-C. Zhang, "Noise-robust edge detector combining isotropic and anisotropic Gaussian kernels," *Pattern Recognit.*, vol. 45, no. 2, pp. 806–820, Feb. 2012.

[27] S. Maldonado, E. Carrizosa, and R. Weber, "Kernel penalized K-means: A feature selection method based on kernel K-means," *Inf. Sci.*, vol. 322, pp. 150–160, Nov. 2015.

[28] P. S. Bradley and O. L. Mangasarian, "Feature selection via concave minimization and support vector machines," in *Proc. 15th Int. Conf. (ICML)*, San Francisco, CA, USA, Morgan Kaufmann, Jul. 1998, pp. 82–90.

[29] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proc. 24th Int. Conf. Mach. Learn. ICML*, New York, NY, USA, 2007, pp. 1151–1157.

[30] M. Lin and Z. Yao, "Dynamic garbage collection scheme based on past update times for NAND flash-based consumer electronics," *IEEE Trans. Consum. Electron.*, vol. 61, no. 4, pp. 478–483, Nov. 2015.

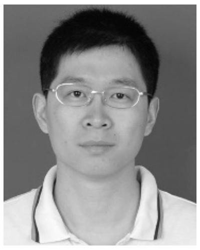
[31] R. Avnur and J. M. Hellerstein, "Eddies: Continuously adaptive query processing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data SIGMOD*, Dallas, TX, USA, 2000, pp. 261–272.

[32] M. Ali, S. I. Ali, D. Kim, T. Hur, J. Bang, S. Lee, B. H. Kang, and M. Hussain, "UEFS: An efficient and comprehensive ensemble-based feature selection methodology to select informative features," *PLoS ONE*, vol. 13, no. 8, 2018, Art. no. e0202705.

[33] Q. Zou, X.-B. Li, W.-R. Jiang, Z.-Y. Lin, G.-L. Li, and K. Chen, "Survey of MapReduce frame operation in bioinformatics," *Briefings Bioinf.*, vol. 15, no. 4, pp. 637–647, Jul. 2014.

[34] Q. Zou, J. Zeng, L. Cao, and R. Ji, "A novel features ranking metric with application to scalable visual and bioinformatics data classification," *Neurocomputing*, vol. 173, pp. 346–354, Jan. 2016.

[35] J. Zeng, D. Li, Y. Wu, Q. Zou, and X. Liu, "An empirical study of features fusion techniques for protein-protein interaction prediction," *Current Bioinf.*, vol. 11, no. 1, pp. 4–12, Mar. 2016.



JUN LIU received the B.S. degree in software engineering and the Ph.D. degree in computer science and technology from Chongqing University, China, in July 2008 and June 2016, respectively. Since 2016, he has been with the School of Software Engineering, Chongqing University of Posts and Telecommunications, China. His research interests include big data analytics, anomaly detection, machine learning, NAND flash memory, information security, and cloud computing.

SULE TANG received the B.S. degree in software engineering from Hebei GEO University, China, in June 2019. She is currently pursuing the M.S. degree with the College of Software Engineering, Chongqing University of Posts and Telecommunications. Her research interests include big data analytics, complex networks, machine learning, deep learning, and large- scale data mining.

GUANGXIA XU received the M.S. and Ph.D. degrees in computer science from Chongqing University, China. She is currently a Professor with the Chongqing University of Posts and Telecommunications. She is also the Research Vice Director of the Network and Information Security Engineering Center, Chongqing, China. Her research interests include block-chain, big data analytics, and security AI. She is also a Committee Member of the Fault Tolerant Computing of China Computer Federation and the Vice Chairman of the Information Security Association in Chongqing, China.

CHUANG MA received the B.S. degree in computer science and technology and the Ph.D. degree in computer science and technology from Jilin University, China, in 2016. Since 2017, he has been with the School of Software Engineering, Chongqing University of Posts and Telecommunications, China. His research interests include delay tolerant networks, social networks, body area networks, and machine learning.

MINGWEI LIN received the B.S. degree in software engineering and the Ph.D. degree in computer science and technology from Chongqing University, Chongqing, China, in 2009 and 2014, respectively. He is currently an Associate Professor with the College of Mathematics and Informatics, Fujian Normal University, China. He has published more than 50 high-quality research articles in international journals and conference proceedings, such as *Nonlinear Dynamics*, *Complexity*, the IEEE Internet of Things Journal, the *International Journal of Intelligent Systems*, IEEE Access, *Sustainable Cities and Society*, *Artificial Intelligence Review*, the IEEE Transactions on Consumer Electronics, and the *Journal of the Operational Research Society*. He has published three ESI highly cited articles. His research interests include decision making and aggregation operator. He got the CSC-IBM Chinese Excellent Student Scholarship, in 2012.