IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 6, 2021, accepted November 23, 2021, date of publication November 25, 2021,
date of current version December 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130905

Ensemble Pruning of RF via Multi-Objective TLBO Algorithm and Its Parallelization on Spark

LANJUN WAN[1,2], KUN GONG[1,2], GEN ZHANG[1,2], CHANGYUN LI[1,2], ZHIBING WANG[1,2], AND XIAOJUN DENG[1]

[1]School of Computer Science, Hunan University of Technology, Zhuzhou 412007, China
[2]Hunan Key Laboratory of Intelligent Information Perception and Processing Technology, Hunan University of Technology, Zhuzhou 412007, China

Corresponding author: Changyun Li (licy469@163.com)

This work was supported in part by the Natural Science Foundation of Hunan Province, China, under Grant 2019JJ60048 and Grant 2019JJ60054; in part by the National Natural Science Foundation for Young Scientists of China under Grant 61702177; in part by the Open Platform Innovation Foundation of Hunan Provincial Education Department under Grant 18K077; in part by the Research Foundation of Education Bureau of Hunan Province, China, under Grant 19C0572; in part by the Key Research and Development Project of Hunan Province under Grant 2019GK2133; and in part by the National Key Research and Development Project under Grant 2018YFB1700204 and Grant 2019YFD1101305.

ABSTRACT Ensemble learning has been widely used in various fields. Still, too many base classifiers will affect the classification time of the ensemble classifier under the big data environment, while reducing base classifiers will affect the classification accuracy of the ensemble classifier. Therefore, the multi-objective teaching-learning-based optimization (MO-TLBO) algorithm is used to carry out ensemble pruning of random forest (RF) to improve the classification accuracy and speed of RF. MO-TLBO algorithm aims at maximizing classification accuracy and minimizing classification time, and it can find a sub-forest with higher classification accuracy and faster classification speed. In addition, considering the vast computational time of ensemble pruning of RF via MO-TLBO algorithm under the big data environment, a vote set is constructed to improve the fitness evaluation process. In the Spark platform, the RF improved by the MO-TLBO algorithm (MO-TLBO-RF) is parallelized based on data parallelism. The Shuffle optimization strategy is proposed to reduce the number of Shuffles in the execution of parallel MO-TLBO-RF. The proposed MO-TLBO-RF is applied to rolling bearing fault diagnosis. The experimental results show that the algorithm can obtain an RF with high fault diagnosis accuracy and fast fault diagnosis speed. The results also prove that the ensemble pruning time can be greatly reduced via the vote set and parallelization of MO-TLBO-RF.

INDEX TERMS Ensemble pruning, multi-objective TLBO algorithm, parallelization, random forest, Spark.

I. INTRODUCTION

Ensemble learning combines multiple base classifiers to form an ensemble classifier, which has been widely used in biology, transportation, energy, industry, medicine, and other fields [1]–[5]. The vibration signals collected from the rolling bearing are time series, and the ensemble learning is suitable for time series classification [6]–[8]. Moreover, due to the high classification accuracy and strong generalization ability, the researches on the fault diagnosis based on ensemble learning increase gradually [4], [9]–[11]. For example, Li *et al*. [9] firstly trained two different feature extractors to enhance feature representation, and then they adopted ensemble learning to obtain fault diagnosis results from different classifiers trained by different feature extractors. Yu and Zhao [10] proposed a probabilistic ensemble learning method based on Bayesian network, which can effectively improve the fault diagnosis accuracy. Ensemble learning can fully utilize multiple fault classifiers to diagnose faults more accurately, but the number of classifiers will restrict the diagnosis speed. With the expansion of industrial production scale and the popularization of intelligent manufacturing, the operations of production equipment will produce a lot of data, which poses a new challenge to ensure the diagnosis accuracy and the diagnosis speed of equipment fault diagnosis using ensemble learning.

Ensemble pruning is a method of selecting multiple base classifiers from the ensemble classifier to get a smaller but

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

better ensemble classifier [12], which mainly includes three pruning ways: pruning based on sorting [13]–[15], pruning based on clustering [16], [17], and pruning based on optimization [18]–[20]. Guo *et al*. [15] proposed a measurement method based on margin and diversity to evaluate the importance of base classifiers. Multiple base classifiers can be selected in descending order to form an ensemble classifier with better performance using the proposed measurement method. Cela and Suárez [17] used the clustering algorithm based on energy to cluster the base classifiers in the ensemble classifier. An ensemble classifier consisting of a representative classifier of each cluster is got. Zhou *et al*. [20] optimized the trained neural networks by genetic algorithm (GA) to obtain a small-scale but better generalization ensemble neural network. Moreover, some researchers [21]–[26] combined the three pruning ways for ensemble pruning. For example, Zhu *et al*. [25] first filtered base classifiers based on the minimization of margin distance and then used an artificial fish swarm algorithm to find the best ensemble classifier from the remaining base classifiers. Onan *et al*. [26] combined the clustering algorithm and multi-objective evolutionary algorithm to find candidate classifiers in each cluster, and a smaller-scale ensemble classifier with better performance is obtained. In recent ensemble pruning techniques [24], [27], a meta-heuristic algorithm is firstly used to generate multiple individuals, then the generated individuals are filtered according to an index such as reduce-error, and finally the rest of individuals is used as the initial population of the meta-heuristic algorithm to search the best ensemble model. The above studies show that ensemble pruning can reduce the size of the ensemble classifier and improve the generalization of the ensemble classifier.

It is a non-deterministic polynomial complete problem to find the best combination of base classifiers in the ensemble classifier [28]. In recent years, the single-objective meta-heuristic algorithms have been usually used for ensemble pruning to find a near-optimal solution in limited time, and the goal is generally the classification accuracy [29], [30]. Furthermore, multi-objective meta-heuristic algorithms can find a satisfactory solution in multiple performance criterias [31]–[33], and some researchers [34]–[38] have explored the application of multi-objective meta-heuristic algorithms in ensemble pruning and achieved good results. For example, Qian *et al*. [37] proposed a multi-objective particle swarm optimization algorithm to maximize the generalization and minimize the size of the ensemble classifier. Peimankar *et al*. [38] used the multi-objective evolutionary algorithm to maximize the classification accuracy and diversity. The existing researches use multi-objective meta-heuristic algorithms to effectively improve the classification accuracy and reduce the size of the ensemble classifier. However, they do not take the classification time of the ensemble classifier as one goal. Because the classification time of different base classifiers may be different, the ensemble classifiers which have the same number of base classifiers may have different classification time. Taking the minimization of the classification time as one goal, the ensemble classifier with the fastest classification speed can be found from these ensemble classifiers with the same number of base classifiers. Therefore, it is necessary to take the minimization of the classification time as one goal, which can further improve the classification speed. In addition, ensemble pruning via meta-heuristic algorithms usually has a huge computational cost under the big data environment.

In meta-heuristic algorithms, swarm intelligence optimization algorithms that solve combinatorial optimization problems by imitating biological activities have been widely favored [39]. In recent years, some new swarm optimization algorithms have been proposed, such as the TLBO algorithm [40], monarch butterfly optimization algorithm [41], slime mould algorithm [42], moth search algorithm [43], hunger games search algorithm [44], Runge Kutta method [45], and Harris hawks optimization algorithm [46]. In these algorithms, TLBO algorithm doesn't have the algorithm-specific parameters [47], i.e., it only needs to adjust the number of individuals and the number of iterations of the population. Therefore, TLBO algorithm is adopted in this paper.

Therefore, in order to find a sub-forest with higher classification accuracy and faster classification speed, the MO-TLBO algorithm, whose two goals are the maximization of the classification accuracy and the minimization of the classification time, is proposed for ensemble pruning of RF. Furthermore, in order to reduce the enormous computational time of ensemble pruning of RF via MO-TLBO algorithm under the big data environment, the RF improved by MO-TLBO algorithm is parallelized on Spark according to data parallelism, the Shuffle optimization strategy is proposed, and a vote set is constructed.

The main contributions of this paper are as follows.

- The MO-TLBO algorithm whose two goals are the maximization of classification accuracy and the minimization of classification time is proposed, and a crossover operator with an adaptive crossover rate is designed to better find the best combination of base classifiers.
- Considering the vast computational time of ensemble pruning of RF via MO-TLBO algorithm under the big data environment, a vote set is constructed to improve the fitness evaluation process.
- MO-TLBO-RF is parallelized on Spark according to data parallelism, which greatly reduces the training time, ensemble pruning time, and the classification time of the RF model.
- The Shuffle optimization strategy is proposed to reduce the number of Shuffles in the execution of parallel MO-TLBO-RF, which further reduces the ensemble pruning time.
- A large number of experiments verifies the effectiveness of MO-TLBO-RF. The results show that it not only has better fault diagnosis accuracy and generalization but also has faster training speed and fault diagnosis speed under the big data environment.

The rest of this paper is organized as follows. Section II introduces the classic TLBO algorithm. Section III discusses the proposed MO-TLBO-RF and its parallelization. Section IV presents the experimental results and analysis. Section V gives the conclusion of this paper.

II. THE CLASSIC TLBO ALGORITHM

The TLBO algorithm is a novel swarm intelligence optimization algorithm proposed by Rao *et al.* [40]. The algorithm is used to solve the optimization problem by simulating the knowledge transfer behaviors of teachers and students. The TLBO algorithm is mainly composed of the teaching stage and learning stage. In the teaching stage, teachers teach students the knowledge to improve students' average scores. In the learning stage, students learn from other students to improve their scores. Supposing that there is a population $P = \{x_1, x_2, \cdots, x_n\}$, x_i^{old} and x_i^{new} represent the i-th learner before and after learning, respectively. The $f(x)$ represents the objective function and the learner with the highest score of $f(x)$ in each iteration is the teacher. The detailed descriptions of teaching stage and learning stage are as follows.

In the teaching stage, a teacher teaches students knowledge according to their average scores, and students' positions are updated through knowledge transfer. The new position of student x_i is calculated by

$$x_i^{\text{new}} = x_i^{\text{old}} + r_i(x_{\text{teacher}} - (T_F x_{\text{mean}})), \tag{1}$$

where

$$T_F = \text{round}[1 + \text{rand}(0, 1)]. \tag{2}$$

In (1), r_i is a random floating-point number between 0 and 1, and x_{teacher} represents the teacher. T_F is the learning factor, which is used to determine the change of x_{mean}, and x_{mean} represents the average value of positions of all students in one iteration. Only when $f(x_i^{\text{new}}) > f(x_i^{\text{old}})$, x_i is updated to x_i^{new}.

In the learning stage, student x_i randomly selects another student x_j to communicate, and the student who has more knowledge can learn something new. The new position of student x_i is calculated by

$$\begin{cases} x_i^{\text{new}} = x_i^{\text{old}} + r_i(x_i - x_j), \text{if} f(x_i) < f(x_j); \\ x_i^{\text{new}} = x_i^{\text{old}} + r_i(x_j - x_i), \text{if} f(x_i) \geq f(x_j). \end{cases} \tag{3}$$

III. THE PROPOSED ALGORITHM

This section mainly discusses the basic process of finding the optimal sub-forest (i.e., the best combination of base classifiers) and the MO-TLBO-RF algorithm.

A. THE BASIC PROCESS OF FINDING THE OPTIMAL SUB-FOREST

In order to further improve the classification accuracy and speed of RF under the big data environment, the ensemble pruning can be used in RF. In this paper, the MO-TLBO algorithm is proposed to combine the decision trees in the original RF model to get a sub-forest with high classification accuracy and fast classification speed, which is described as follows.

Step 1. Get the original RF model. The training set is used to train an RF model with k decision trees.

Step 2. Construct the vote set. The original RF model is used to classify samples in the validation set, and the vote results of all samples and the classification time of each decision tree are recorded.

Step 3. Evaluate the original RF model. Based on the vote results of each decision tree, the classification accuracy of the original RF model for each label in the validation set can be obtained.

Step 4. Reduce the size of the vote set. If the original RF model can achieve 100% accuracy for a certain label, the vote results of the label are removed from the vote set.

Step 5. Find the optimal sub-forest. The MO-TLBO-RF algorithm is used to arrange and combine the decision trees in the original RF model to find the optimal sub-forest, and the vote set is used to complete the fitness evaluation of the sub-forest quickly.

B. MO-TLBO-RF

This section describes the proposed MO-TLBO-RF algorithm, which is shown in Algorithm 1.

1) WEIGHTED MULTI-OBJECTIVE

The multi-objective problem is usually solved by transforming a multi-objective into a single-objective. The common methods are linear weighting method [48], hierarchical optimization method [49], and Pareto optimization method [50]. The two goals of the proposed MO-TLBO-RF algorithm are the maximization of classification accuracy and the minimization of classification time. Since the two goals can be transformed into a dimensionless single goal, the linear weighting method is adopted in this paper to avoid the excessive amount of calculation.

The classification accuracy is a dimensionless quantity, representing the proportion of the samples of correct classification to the total samples. The sum of the classification time of each decision tree in the original RF model is divided using the classification time of the sub-forest. The classification time can be converted into a dimensionless quantity, which represents the proportion of the classification time of the sub-forest to the classification time of the original RF model. Since the classification time is expected to be minimized, a new dimensionless quantity is obtained by subtracting the proportion from 1, which represents the proportion of the classification time of the unused decision trees to the classification time of the original RF model. Therefore, the fitness value of the sub-forest is calculated by

$$F_i = Acc_i + timeWeight \times (1 - t_i/t_{\text{all}}), \tag{4}$$

where F_i represents the fitness value of the sub-forest i, t_i is the classification time of the sub-forest i, and t_{all} is the sum of classification time of all decision trees in the original RF model. *timeWeight* is a decimal to constrain

Algorithm 1 The Serial MO-TLBO-RF Algorithm

Input: An original RF model, the number of individuals in the population w, the iteration number of the population p, *voteSet*, *timeRecord*, *timeWeight*, and the early-stop threshold *earlyStop*
Output: An optimal sub-forest

```
Initialize the variable counter, the key-value pair counterMap, and the population by a random number generator;
Evaluate the fitnesses of individuals in the population by (4);
Identify the teacher (i.e., the best solution);
for j = 1 to p do
  for k = 1 to w − 1 do
    Get the new position of student_k by (5) and (6);
    Get the classification accuracy of student_k by voteSet;
    Get the classification time of student_k by timeRecord;
    Evaluate the fitness of new position of student_k by (4);
    if the new fitness value > the current fitness value then
      The current position and fitness are replaced with the new position and fitness;
      counterMap(student_k) = 0;
    else
      counterMap(student_k) += 1;
    end if
    Get the new position of student_k by (6) and (7);
    Evaluate the fitness of new position of student_k by (4);
    if the new fitness value > the current fitness value then
      The current position and fitness are replaced with the new position and fitness;
      counterMap(student_k) = 0;
    else
      counterMap(student_k) += 1;
    end if
    if counterMap(student_k) ≥ 4 then
      Get the new position of student_k by mutation operator;
      Evaluate the fitness of new position by (4);
      The current position and fitness are replaced with the new position and fitness;
    end if
  end for
  Identify the new teacher;
  if the new teacher == the current teacher then
    counter += 1;
  else
    counter = 0;
  end if
  if counter > earlyStop then
    break;
  end if
end for
```

MO-TLBO-RF to find the sub-forest with high classification accuracy and less classification time. If the value of *timeWeight* is too large, MO-TLBO-RF will tend to find the sub-forest with less classification time but low classification accuracy. If the value of *timeWeight* is too small, MO-TLBO-RF will tend to find the sub-forest with high classification accuracy but more classification time.

2) ADAPTIVE CROSSOVER OPERATOR

TLBO algorithm is initially used to solve continuous optimization problems, but many studies show that TLBO algorithm is also suitable for discrete optimization problems. For example, Shao *et al.* [51] proposed a discrete TLBO algorithm based on a teaching-probabilistic learning mechanism to solve the job-shop scheduling problem. Sevinc and Dökeroğlu [52] combined TLBO algorithm and GA to solve discrete optimization problems.

To better find the optimal sub-forest, the TLBO algorithm and GA are combined. Therefore, binary coding is used to represent the position of an individual in the population, and 1 represents the decision tree is selected, as shown in Fig. 1. Considering that the knowledge learned by students from teachers has two possibilities of continuity and discontinuity, the crossover operator of bit crossover is used to simulate the knowledge transfer in TLBO algorithm, as shown in Fig. 2.

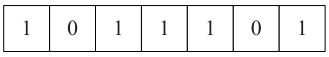


FIGURE 1. The position of an individual in the population.

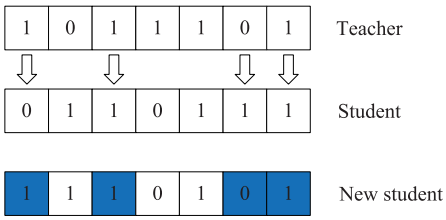


FIGURE 2. The diagram of the update of a student position.

However, the crossover operator brings the problem of parameter adjustment of crossover rate and mutation rate. Rao [47] pointed out that TLBO algorithm is a heuristic algorithm without algorithm-specific parameters, and there is no additional burden of parameters. Based on this point, a crossover operator with an adaptive crossover rate is designed. Note that when the crossover rate is greater than 1, it will be modified to 0.9 (i.e., a commonly used crossover rate).

In the teaching stage, each student's ability to receive knowledge is affected by the teachers' teaching methods and their learning enthusiasm, so that the crossover rate of each knowledge transfer is different. The crossover rate CR_{teaching} is calculated by

$$CR_{\text{teaching}} = T_F(f(x_{\text{teacher}}) - f(x)_{\text{mean}}), \tag{5}$$

where

$$T_F = 1 + \text{rand}[0, student\_num/2]. \tag{6}$$

In (5), $f(x)_{\text{mean}}$ is the average value of students' fitness values, and *student_num* represents the number of students. In (6), T_F is the learning factor which is used to control the fluctuation of crossover rate to express the effect of learning knowledge.

In the teaching stage, student x_i randomly selects another student x_j to communicate, and the student can learn something new from the gap between two communicated students.

The crossover operator is also used to update the positions of students, and the crossover rate CR_{learning} is calculated by

$$CR_{\text{learning}} = T_F |f(x_i) - f(x_j)|, \tag{7}$$

where $f(x_i)$ is the fitness value of student i, and $f(x_j)$ is the fitness value of student j.

3) MUTATION OPERATOR

In TLBO algorithm, whether in the teaching or learning stage, only when the fitness value of the new individual is higher than the fitness value of the old individual, the old individual will be updated to the new individual. In other words, ideally, the individual can be updated twice in one iteration. Therefore, a counting table is used to record the number of times each individual has not been updated continuously. When the teaching or learning stage is completed, the count will be increased by one if an individual is not updated. When the count reaches four, it is considered that the individual has fallen into the local optimal solution, and the mutation operator is carried out. The mutation operator is help to jump out of the local optimal solution and reduce the risk of premature convergence. In this paper, a mutation operator of bitwise negation with a 50% mutation rate is used to get a new individual, which directly replaces the old individual to maintain the diversity of the population.

4) VOTE SET

When a swarm intelligence optimization algorithm is used in ensemble pruning, it is necessary to evaluate the fitness values of generated sub-forests. The traditional fitness evaluation process is as follows. Firstly, the binary coding is used to represent the selections of decision trees in the sub-forest. Secondly, the binary coding of an individual is decoded to get an actual sub-forest, i.e., an ensemble classifier. Thirdly, the validation set is used to evaluate the classification accuracy and classification time of the sub-forest. Finally, the fitness value of the sub-forest is calculated according to (4). In experiments, it is found that the above process is time-consuming when the size of validation set is large so that the computational time of using swarm intelligence optimization algorithm for ensemble pruning will be very large under the big data environment. In order to quickly evaluate the fitness of the sub-forest, a vote set is constructed, and the new fitness evaluation process is as follows.

Step 1. The original RF model is used to classify samples in the validation set, and the vote results and classification time of each decision tree are recorded. Thus, the *voteSet* and *timeRecord* as shown below are obtained.

$$\begin{bmatrix} label_1 & vote_{1,1} & \cdots & vote_{n,1} \\ label_2 & vote_{1,2} & \cdots & vote_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ label_m & vote_{1,m} & \cdots & vote_{n,m} \end{bmatrix}$$
$$\begin{bmatrix} t_{\text{all}} & t_1 & t_2 & \cdots & t_n \end{bmatrix}$$

In the *voteSet*, $label_i$ represents the label of sample i, $vote_{j,i}$ is the result of decision tree j voting on sample i, m is the number of samples in the validation set, and n is the number of decision trees. In the *timeRecord*, t_i is the classification time of decision tree i for the validation set, and t_{all} represents the sum of classification time of each decision tree.

Step 2. According to the binary coding of the individual, the vote results and classification time of the corresponding decision trees are extracted from the *voteSet* and *timeRecord*, respectively. The vote results of the sub-forest are obtained according to the majority voting method, and the classification time of the sub-forest is obtained by summing the classification time extracted from the *timeRecord*.

Step 3. According to the sample labels in the *voteSet*, the classification accuracy of the sub-forest can be got, and the fitness value of the sub-forest is calculated by (4).

Compared with the traditional fitness evaluation process, the new process eliminates the decoding stage and avoids the repeated classification of decision trees for the validation set. It is found that the vote set can greatly reduce the time of evaluating the fitness of an individual. Note that the vote set can replace a validation set with m samples of d-dimension features and 1-dimension label with an $m \times (n+1)$ matrix. If d is less than n, the size of the *voteSet* will be larger than the size of the validation set, and therefore the computational time will be reduced by increasing the memory space. If d is larger than n, the size of the *voteSet* is smaller than the size of the validation set, which means that the vote set can reduce not only the computational time, but also reduce the memory space.

5) TIME COMPLEXITY ANALYSIS OF THE SERIAL MO-TLBO-RF

In the serial MO-TLBO-RF described in Algorithm 1, the outer for-loop repeats p times (see line 4), the inner for-loop repeats $w-1$ times (see line 5), multiple fitness calculations are performed during each execution of the inner for-loop (see lines 9, 17, and 26), and the time complexity of the fitness calculation is $O(w)$. In addition, the time complexity of RF is $O(n(md\log m))$ [53]. Therefore, the time complexity of the serial MO-TLBO-RF algorithm is $O(n(md\log m)+pw^2)$. Although the time complexity of the fitness calculation is small, it becomes the most time-consuming part of the serial MO-TLBO-RF algorithm due to the large amount of data processed and many times of operations.

C. THE PARALLELIZATION OF MO-TLBO-RF

This section discusses the parallel design, Shuffle optimization, and parallel implementation of MO-TLBO-RF.

1) THE PARALLEL DESIGN

Currently, the two main ways of parallelization are task parallelism and data parallelism. The parallelization process of MO-TLBO-RF based on task parallelism is shown in Fig. 3. Each element of the *populationRDD* includes a vote set, a teacher, and two students, where rStudent i represents a student randomly selected from all students except for the

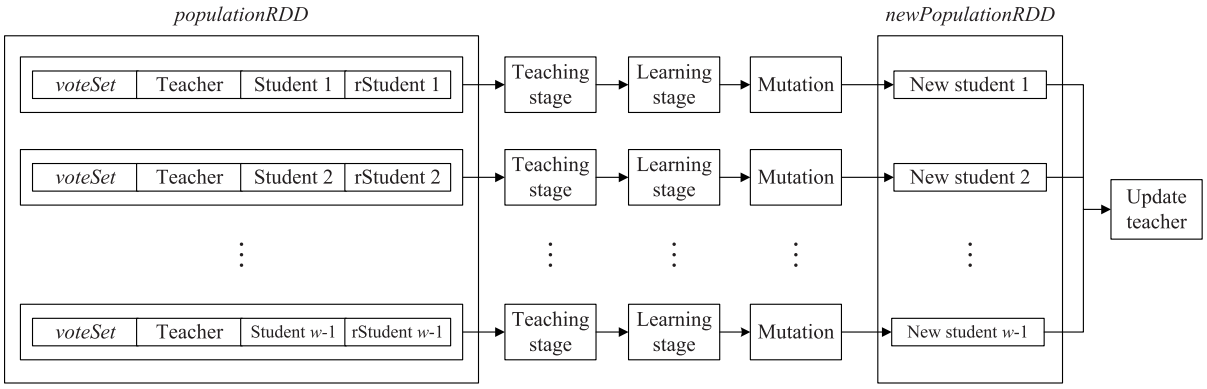


FIGURE 3. The parallelization process of MO-TLBO-RF based on task parallelism.

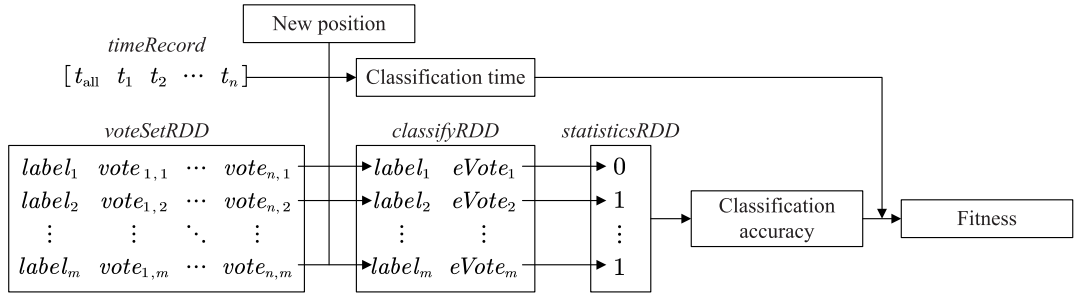


FIGURE 4. The parallelization process of fitness evaluation based on data parallelism.

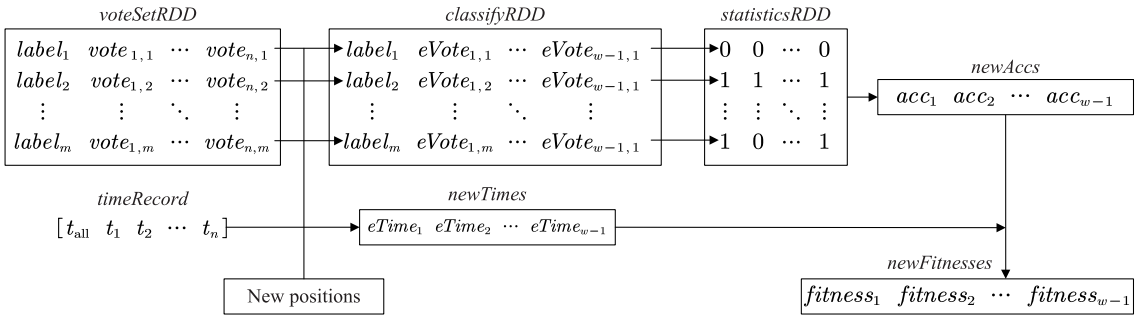


FIGURE 5. The parallelization process of fitness evaluation using the Shuffle optimization strategy.

i-th student. The *populationRDD* includes $w-1$ elements, thus one iteration of the population is divided into $w-1$ computational tasks which can be executed in parallel. Each task needs to complete the teaching stage, learning stage, and mutation. After $w-1$ tasks have been completed, and the teacher is updated according to the fitness values of individuals in the population.

The parallelization process of fitness evaluation based on data parallelism is shown in Fig. 4. The *voteSet* is transformed into an RDD *voteSetRDD*, and an element of *voteSetRDD* is a row of *voteSet*. After the update of student position and statistics of classification time have been completed on Spark driver. The parallelization process of fitness evaluation is as follows. Firstly, the ensemble vote results of the student for each sample are obtained according to its position and the majority voting method. Secondly, the ensemble vote results are compared with the sample labels, 0 means the wrong classification, and 1 means the correct classification. Thirdly, the classification results obtained by each Spark worker are summed and divided by the total number of samples to get the classification accuracy. Finally, the fitness value of the new student position is calculated by (4).

If the task parallelism is adopted, each element of *populationRDD* needs to be regarded as a partition, and each partition needs to contain a *voteSet*. Because students and the teacher will be updated in each iteration, *populationRDD* needs to be recreated according to the new population in each iteration, which will cause $w-1$ *voteSet* to be uploaded to the Spark cluster frequently, resulting in large additional data transfer overhead. If the data parallelism is adopted, the

voteSet used in Algorithm 1 can remain unchanged. In other words, the *voteSetRDD* can remain unchanged, which means that the *voteSet* only needs to be uploaded to the Spark cluster once so that the data transfer overhead is small. Therefore, MO-TLBO-RF is parallelized based on data parallelism.

2) THE SHUFFLE OPTIMIZATION STRATEGY

Spark driver is responsible for executing the serial parts of the program, and all worker nodes are responsible for parallelly executing RDD calculation. Spark programs need to avoid too many Shuffles, because Shuffle usually will produce a lot of data transfer overhead [54]. Therefore, the fewer Shuffles are performed in a Spark program, the shorter the parallel execution time.

The MO-TLBO-RF algorithm needs to perform $2(w-1)p$ fitness evaluations without considering early stop and mutation. As shown in Fig. 4, the process of using the reduce operator to sum each element of *statisticsRDD* to obtain the classification accuracy will produce a Shuffle, so that the parallel MO-TLBO-RF algorithm needs to perform $2(w-1)p$ Shuffles. The computational time of the parallel MO-TLBO-RF algorithm can be significantly reduced by reducing the number of Shuffles. Therefore, the parallelizations of the teaching and learning stages in the MO-TLBO-RF algorithm are improved, which can be described as follows. At first all positions of students are updated and all classification time of all students is calculated on Spark driver, then the fitness values of all students are calculated in parallel on multiple worker nodes, and finally each student whose fitness value is lower than the new fitness value is updated. In the parallel MO-TLBO-RF algorithm using the Shuffle optimization strategy, an RDD can be used to calculate the fitness values of all students. As shown in Fig. 5, each element of *classifyRDD* includes a sample label and the ensemble vote results of all students for the label, the *statisticsRDD* can be obtained by comparing sample labels and the ensemble vote results of all students, and the process of using the reduce operator to sum each element of *statisticsRDD* to obtain the classification accuracies of all students will produce a Shuffle. The parallel MO-TLBO-RF algorithm using the Shuffle optimization strategy needs to perform $2p$ fitness evaluations so that it only needs to perform $2p$ Shuffles.

3) THE PARALLEL IMPLEMENTATION

The parallel MO-TLBO-RF using the Shuffle optimization strategy based on Spark platform is shown in Algorithm 2, which can be described as follows.

Step 1. Initialize the population. The population is initialized, and the fitness of each individual in the population is evaluated. The individual with the highest fitness value is teacher, and the rest of the individuals are students.

Step 2. Update students in the teaching stage. All students carry out the teaching stage, the fitness values of all students

Algorithm 2 The Parallel MO-TLBO-RF Algorithm

Input: An original RF model, the number of individuals in the population w, the iteration number of the population p, *voteSetRDD*, *timeRecord*, *timeWeight*, and the early-stop threshold *earlyStop*

Output: An optimal sub-forest

Initialize the variable *counter*, the key-value pair *counterMap*, and the population by a random number generator;
Evaluate the fitnesses of individuals in the population by (4);
Identify the teacher;
for $j = 1$ **to** p **do**
  for $k = 1$ **to** $w - 1$ **do**
    Get the new position of $student_k$ by (5) and (6);
    Get the classification time of $student_k$ by *timeRecord*;
  end for
  Parallelly calculate the fitness values of all students by (4);
  for $k = 1$ **to** $w - 1$ **do**
    if the new fitness value $>$ the current fitness value **then**
      The current position and fitness of $student_k$ are replaced with the new position and fitness;
      $counterMap(student_k) = 0$;
    else
      $counterMap(student_k) += 1$;
    end if
  end for
  for $k = 1$ **to** $w - 1$ **do**
    Get the new position of $student_k$ by (6) and (7);
    Get the classification time of $student_k$ by *timeRecord*;
  end for
  Parallelly calculate the fitness values of all students by (4);
  for $k = 1$ **to** $w - 1$ **do**
    if the new fitness value $>$ the current fitness value **then**
      The current position and fitness of $student_k$ are replaced with the new position and fitness;
      $counterMap(student_k) = 0$;
    else
      $counterMap(student_k) += 1$;
    end if
  end for
  for $k = 1$ **to** $w - 1$ **do**
    if $counterMap(student_k) \geq 4$ **then**
      Get the new position of $student_k$ by mutation operator;
      Get the classification time of $student_k$ by *timeRecord*;
    end if
  end for
  if the number of mutated students > 0 **then**
    Parallelly evaluate fitnesses of mutated students by (4);
    for $r = 1$ **to** the number of mutated students **do**
      The current position and fitness of $student_r$ are replaced with the new position and fitness;
      $counterMap(student_r) = 0$;
    end for
  end if
  Identify the new teacher;
  if the new teacher == the current teacher **then**
    $counter += 1$;
  else
    $counter = 0$;
  end if
  if $counter > earlyStop$ **then**
    break;
  end if
end for

are calculated in parallel, and each student whose fitness is lower than the new fitness is updated.

Step 3. Update students in the learning stage. All students carry out the learning stage, the fitness values of all students are calculated in parallel, and each student whose fitness is lower than the new fitness is updated.

Step 4. Carry out the mutation. The student who doesn't continuously update its position four times carries out the mutation operator, the fitness values of the mutated students are calculated, and the old students are replaced with the mutated students.

Step 5. Update the teacher. The optimal solution of the population is regarded as the new teacher. If the fitness value of the new teacher is larger than the fitness value of the current teacher, the teacher is updated, and the rest of the individuals are students.

Step 6. If the number of times the teacher has not been updated continuously exceeds the *earlyStop*, or the iteration number of the population exceeds p, the algorithm is stopped, and the current teacher as the optimal solution is output; otherwise, return to Step 2.

4) TIME COMPLEXITY ANALYSIS OF THE PARALLEL MO-TLBO-RF

In the parallel MO-TLBO-RF described in Algorithm 2, the outer for-loop repeats p times (see line 4), multiple fitness calculations of the population are performed in parallel during each execution of the outer for-loop (see lines 9, 22, and 38). and the time complexity of the parallel fitness calculation of the population is $O(w^2/uv)$, where u is the number of worker nodes and v is the number of CPU cores within a worker node. Because the computational time of six inner for-loops (see lines 5, 10, 18, 23, 31, and 39) is very small, their time complexities need not be considered. In addition, the time complexity of the parallel RF algorithm based on Spark is $O(n(md \log m)/uv)$. Therefore, the time complexity of the parallel MO-TLBO-RF algorithm is $O((n(md \log m) + pw^2)/uv)$.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL DATASET

In order to evaluate the effectiveness of the proposed MO-TLBO-RF algorithm, the rolling bearing dataset provided by Paderborn University in Germany [55] is used to carry out experiments. The dataset includes normal state data, outer race fault data, and inner race fault data, which are current signals and vibration signals collected at the frequency of 64 kHz. The fault data include artificial fault data and real fault data. The artificial damages of rolling bearings are made by drilling, electric engraver, and electrical discharge machining. The real damages of rolling bearings are made by the test rig of accelerated life.

Paderborn University [55] pointed out the vibration signals can more accurately reflect the state of rolling bearings than the current signals. In view of this, the vibration signals are selected to conduct experiments. In the data preprocessing, firstly, the vibration signals are divided into many samples, and each sample includes 4096 sampling data. Secondly, the wavelet packet decomposition [56] is used to decompose each sample at three levels. The data obtained from the third level decomposition is used to calculate the wavelet energy to get eight time-frequency features. Thirdly, 10 time-domain features are extracted from the time-domain signals of each sample, which include the mean value, variance, standard deviation, root mean square, skewness, kurtosis, waveform factor, peak factor, pulse factor, and margin factor. Finally, 18-dimension feature vectors are obtained by connecting time-frequency features with time-domain features.

In order to evaluate the rolling bearing fault diagnosis accuracy of the proposed MO-TLBO-RF algorithm under the real environment, the normal state data and real fault data are selected. The categorization of data files selected from the dataset provided by Paderborn University is shown in Table 1. These data are preprocessed to get the dataset DATA A. Moreover, in order to better evaluate the performance of the proposed algorithm under the big data environment, the sliding window [57] is used to enhance these data listed in Table 1, and the enhanced data are preprocessed to get the dataset DATA B whose data size reaches 35 GB. DATA A and DATA B are divided into a training set, validation set, and test set according to the ratio of 7:1:2, respectively. Specifically, the feature dimensions of the training set, validation set, and test set are 18, the dimensions of vote set is 100, and the sizes of the training set, validation set, and test sets from DATA B are 24.5 GB, 3.5 GB, and 7.0 GB, respectively.

TABLE 1. Categorization of data files selected from the dataset provided by Paderborn University.

Normal State	Outer Race Fault	Inner Race Fault
K001	KA04	KI04
K002	KA15	KI14
K003	KA16	KI16
K004	KA22	KI18
K005	KA30	KI21

B. EXPERIMENTAL SETTINGS

The Spark cluster used in experiments includes one master node and four worker nodes. The hardware configuration of the master node includes one quad-core Intel Xeon E3-1225 V5 CPU at 3.3 GHz and 32 GB main memory. The hardware configuration of each worker node includes one eight-core Intel Core i7-9700k CPU at 3.6 GHz and 64 GB main memory. The software configuration of the Spark cluster is as follows: CentOS 8.1, Hadoop 3.2, and Spark 3.0.

The parameter settings of RF are shown in Table 2. The detailed explanations of parameters can be found in [58], where *numTrees* represents the number of decision trees in RF. Increasing the value of *numTrees* can significantly improve the classification accuracy of RF. Still, when it reaches a certain value, the classification accuracy of RF will

TABLE 2. Parameter settings of RF.

Parameter Name	Parameter Value
numTrees	100
maxDepth	18
maxBins	70
impurity	Gini
minInstancePerNode	1 : DATA A 50 : DATA B

TABLE 3. Parameter settings of MO-TLBO-RF.

Parameter Name	Parameter Value
timeWeight	0.01
populationSize	30
iterations	100
earlyStop	20

not continue to be improved. It will significantly increase the training time and classification time of the RF model.

The parameters of MO-TLBO-RF are listed in Table 3. *timeWeight* is used to constrain MO-TLBO-RF to find the sub-forest with high classification accuracy and less classification time. The consequences of improper selection of *timeWeight* are described in Section III-B1. *populationSize* is the number of individuals in the population. The choice of the parameter will affect the quality of the sub-forest and the ensemble pruning time. *iterations* denotes the iteration number of the population. *earlyStop* is the threshold of early stopping the algorithm. In order to avoid too long execution time of the algorithm, when the number of times the teacher has not been updated continuously exceeds the value of *earlyStop*, it is considered that the algorithm has found the optimal sub-forest. Note that the parameters used in this paper are selected by the grid-search method, and the fitness calculated by (4) is used as the evaluation index of the grid-search method.

In this paper, to accurately evaluate the performance of a fault diagnosis model, each experiment is repeated 30 times. The measurement results are expressed in the form of the mean and standard deviation (std).

C. MODEL TRAINING AND VERIFICATION

1) COMPARISON OF RF, IRF, gcForest, AND MO-TLBO-RF

To evaluate the effectiveness of the proposed MO-TLBO-RF, four different RF algorithms, i.e., Spark-RF [59], Spark-IRF [58], gcForest [60], and MO-TLBO-RF are used for fault diagnosis model training with DATA A, respectively. Spark-RF is an RF algorithm provided by Spark MLlib, Spark-IRF is an improved RF algorithm based on sub-forest optimization and it is implemented with Spark, and gcForest is a latest available open-source ensemble learning model. In this experiment, the parameter settings of Spark-RF are listed in Table 2, the parameter settings of Spark-IRF can be found in [58], and the key parameter settings of gcForest are as follows: the number of estimators in each cascade layer is set to 4, the number of decision trees in each estimator is set to 100, and the type of the predictor concatenated to the deep forest is specified as "forest".

TABLE 4. Fault diagnosis accuracies obtained by different RF algorithms using real fault data.

Algorithm	Fault Diagnosis Accuracy	Number of Decision Trees
RF	98.30%	Unknown
Spark-RF	99.39±0.21%	100
Spark-IRF	99.42±0.25%	39±6
gcForest	98.68±0.03%	1945±455
MO-TLBO-RF	99.49±0.13%	27±4

Table 4 shows the average fault diagnosis accuracies obtained by different RF algorithms using real fault data. Compared with the fault diagnosis accuracy of RF from Paderborn University [55], the diagnosis accuracies of Spark-RF, Spark-IRF, and MO-TLBO-RF are improved by 1.09%, 1.12%, and 1.19%, respectively. Note that the experimental dataset used in [55] is consistent with DATA A. The results show that the ensemble pruning of RF via MO-TLBO algorithm can improve the fault diagnosis accuracy to a certain extent and the data preprocessing method used in this paper is suitable for real fault data. The average fault diagnosis accuracy of MO-TLBO-RF is 0.07% higher than the average fault diagnosis accuracy of Spark-IRF, which means that the proposed MO-TLBO algorithm is more effective than the sub-forest optimization method based on similarity proposed in [58]. The average fault diagnosis accuracy of MO-TLBO-RF is 0.81% higher than the average fault diagnosis accuracy of gcForest, which indicates that the proposed MO-TLBO algorithm is more suitable for fault diagnosis of rolling bearing than gcForest.

As shown in Table 4, compared with Spark-RF, the number of decision trees of Spark-IRF is reduced by 61%, and the number of decision trees of MO-TLBO-RF is reduced by 73%, which indicates that the ensemble pruning can effectively reduce the number of decision trees. Compared with Spark-IRF, the number of decision trees of MO-TLBO-RF is reduced by 31%, which means that the sub-forest with higher fault diagnosis accuracy and fewer decision trees can be found by using the MO-TLBO algorithm. The number of decision trees of MO-TLBO-RF is 1.39% of that of gcForest, which indicates that the structure of MO-TLBO-RF is much simpler than gcForest.

To further validate the effectiveness of the proposed MO-TLBO-RF in various scenarios, a series of experiments are conducted on 28 different datasets from UCI machine learning repository [61]. Table 5 shows the comparison of classification accuracies of Spark-RF, gcForest, and MO-TLBO-RF for different datasets. As shown in Table 5, compared with Spark-RF and gcForest, MO-TLBO-RF has better classification accuracies for 22 different datasets. The experimental results show that the ensemble pruning via MO-TLBO algorithm is effective in most datasets, and it has a minor negative effect on the classification accuracy of RF in a few datasets. Moreover, compared with Spark-RF and

TABLE 5. Comparison of classification accuracies of Spark-RF, gcForest, and MO-TLBO-RF for different datasets.

Name	Dataset			Spark-RF	gcForest	MO-TLBO-RF
	The number of feature dimensions	The number of class labels	The number of samples	Average classification accuracy	Average classification accuracy	Average classification accuracy
Acute inflammations	6	2	120	99.54±1.47%	**100±0%**	99.64±0.72%
Balance scale	4	3	625	94.77±1.91%	91.83±1.63%	**94.90±0.70%**
Congressional voting records	16	2	435	98.33±1.24%	95.99±**0.51**%	**98.46**±0.55%
Diagnostic Wisconsin breast cancer	32	2	569	98.75±0.91%	94.24±0.73%	**98.95±0.43%**
Ecoli	8	8	336	94.66±3.47%	85.87±2.33%	**95.46±1.31%**
EEG eye state	15	2	14980	**97.26**±3.42%	95.40±**0.21**%	97.24±1.98%
Fertility	10	2	100	95.01±4.33%	87.13±2.29%	**95.70±1.21%**
Hayes-roth	5	3	160	94.27±4.51%	84.44±2.35%	**94.42±2.31%**
Ionosphere	34	2	351	97.49±1.51%	95.98±0.65%	**98.05±0.64%**
Iris	4	3	150	97.89±1.85%	91.67±1.00%	**98.29±0.93%**
Letter	16	26	20000	**98.29**±0.27%	96.96±**0.11**%	98.19±0.15%
Musk – 1	168	2	476	95.49±3.26%	95.23±1.73%	**96.61±0.94%**
Musk – 2	168	2	6598	95.24%±0.39	94.72±1.24%	**96.30±0.11%**
Mammographic mass	6	2	961	**90.31**±1.93%	81.67±1.16%	90.05±**0.51**%
Polish companies bankruptcy – 1st year	64	2	7027	**98.99**±0.21%	97.82±**0.10**%	98.95±0.14%
Polish companies bankruptcy – 2nd year	64	2	10173	**98.79**±0.29%	96.90±**0.08**%	98.78±0.16%
Polish companies bankruptcy – 3rd year	64	2	8207	98.63±0.23%	95.59±0.14%	**98.69±0.11%**
Polish companies bankruptcy – 4th year	64	2	9792	98.56±0.20%	96.39±0.13%	**98.57±0.11%**
Polish companies bankruptcy – 5th year	64	2	5910	98.43±0.37%	96.45±0.20%	**98.47±0.12%**
QSAR biodegradation	41	2	1055	95.82±1.86%	85.92±1.12%	**95.99±0.64%**
Semeion	256	10	1593	98.94±0.57%	94.47±0.47%	**98.97±0.34%**
Seismic-bumps	19	2	2584	97.05±0.63%	93.35±**0.15**%	**97.22**±0.31%
Sonar	60	2	208	94.68±3.34%	84.81±2.49%	**95.11±1.79%**
Spambase	57	2	4601	97.86±0.51%	95.87±**0.28**%	**97.90**±0.29%
Statlog(heart)	13	2	270	94.12±3.87%	86.37±2.73%	**94.95±1.39%**
Vowel	10	11	528	97.75±0.98%	96.38±0.80%	**97.84±0.66%**
Waveform	21	3	5000	95.30±0.63%	84.52±0.39%	**95.47±0.10%**
Waveform – noise	40	3	5000	95.41±0.79%	85.26±0.42%	**95.57±0.31%**

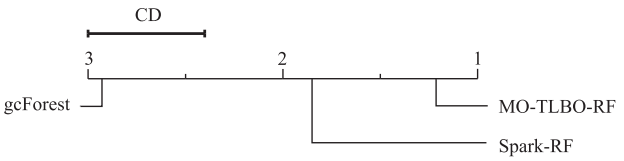


FIGURE 6. Critical difference diagram for the Bonferroni-Dunn test. Comparison of RF algorithms against each other on the basis of classification accuracy ($\alpha = 0.05$).

gcForest, MO-TLBO-RF has the lowest standard deviations in 20 different datasets, which means that the classification accuracies obtained by MO-TLBO-RF on different datasets are relatively stable. In addition, gcForest has lower classification accuracies in some datasets (such as ecoli, hayes-roth, and mammographic mass), and the reason may be that gcForest is not suitable for these datasets with low feature dimensions and few samples.

To demonstrate that the experimental results are statistically significant, the Bonferroni-Dunn test is performed in the data from Table 5, and the test details can be seen in [62]. In Fig. 6, the critical difference whose value is 0.60 is clearly marked as CD, and the numbers on the axis represent the average rankings of the three different RF algorithms used in the experimental comparison. As can be seen form Fig. 6, the average rankings of Spark-RF, gcForest, and MO-TLBO-RF are 1.86, 2.93, and 1.21, respectively. The difference of average ranking between Spark-RF and MO-TLBO-RF is larger than the critical difference, and the difference of average ranking between gcForest and MO-TLBO-RF is also larger than the critical difference. Fig. 6 shows the MO-TLBO-RF is better in classification accuracy.

2) COMPARISON OF DIFFERENT SWARM INTELLIGENCE OPTIMIZATION ALGORITHMS

To evaluate the effectiveness of the MO-TLBO algorithm, three different swarm intelligence optimization algorithms are used for ensemble pruning of RF, i.e., RF improved by multi-objective genetic algorithm (MO-GA-RF), RF improved by multi-objective whale optimization algorithm (MO-WOA-RF), and MO-TLBO-RF. In MO-WOA-RF, the

sigmoid function is adopted, which can enable WOA to solve discrete optimization problems [63]. In this experiment, DATA A is used to evaluate the fault diagnosis accuracies of three different algorithms, and DATA B is used to evaluate the fault diagnosis time of three different algorithms on the Spark cluster. For the sake of fairness, the three algorithms use the same RF model trained by the same training set. It is worth noting that when evaluating the fault diagnosis time of the three algorithms, the fault diagnosis model is used to diagnose all data of DATA B. The parameter settings of MO-GA and MO-WOA are shown in Table 6.

TABLE 6. The parameter settings of MO-GA and MO-WOA.

Algorithm	Parameter Name	Parameter Value
MO-GA	*dnaNum*	60
	iterations	100
	earlyStop	20
	crossRate	0.9
	mutateRate	0.05
	timeWeight	0.3
MO-WOA	*whaleNum*	40
	iterations	100
	earlyStop	20
	a	2
	b	1
	timeWeight	0.05

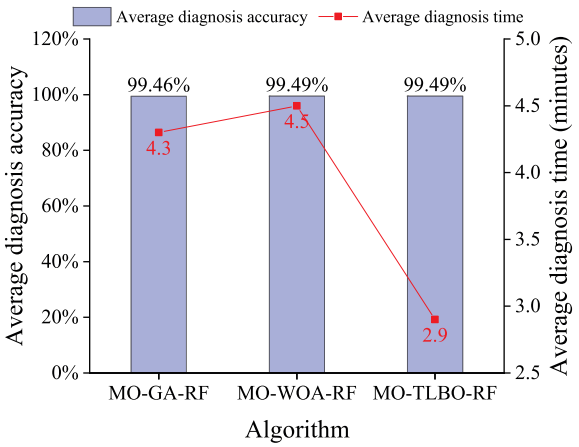


FIGURE 7. Effectiveness comparison of ensemble pruning of RF via different multi-objective swarm intelligence optimization algorithms.

Fig. 7 shows the effectiveness comparison of the ensemble pruning of RF via different multi-objective swarm intelligence optimization algorithms. As can be seen from Fig. 7, the average fault diagnosis accuracies of MO-GA-RF, MO-WOA-RF, and MO-TLBO-RF are 99.46%, 99.49%, and 99.49% respectively, and the standard deviations of MO-GA-RF, MO-WOA-RF, and MO-TLBO-RF are 0.22, 0.17, and 0.13 respectively. The average fault diagnosis accuracies of MO-WOA-RF and MO-TLBO-RF are the same and 0.03% higher than the average fault diagnosis accuracy of MO-GA-RF, which indicates that MO-WOA-RF and MO-TLBO-RF have found the sub-forest with the highest fault diagnosis accuracy. The average fault diagnosis time of MO-GA-RF, MO-WOA-RF, and MO-TLBO-RF is 4.3 minutes, 4.5 minutes, and 2.9 minutes respectively, and the standard deviations of MO-GA-RF, MO-WOA-RF, and MO-TLBO-RF are 0.6, 0.3, and 0.2 respectively. Compared with MO-GA-RF and MO-WOA-RF, the fault diagnosis time of MO-TLBO-RF is reduced by 32.6% and 35.6%, respectively. The results show that MO-TLBO-RF can achieve the two goals of the maximization of fault diagnosis accuracy and minimization of fault diagnosis time.

3) VALIDATION OF MODEL GENERALIZATION

To evaluate the generalization of MO-TLBO-RF, the artificial fault data are used to train fault diagnosis models by Spark-RF, Spark-IRF, and MO-TLBO-RF, and the trained models are used to diagnose the real fault data. The artificial fault data used for training and the real fault data used for testing are shown in Table 6, and they are preprocessed according to the data preprocessing process mentioned in Section IV-A, where the test data are divided into validation set and test set according to the ratio of 3:7. The same generalization experiment has been carried out by Paderborn University [55] using RF, and the experimental data used in [55] are consistent with the data listed in Table 7.

TABLE 7. Categorization of data files selected from the dataset provided by Paderborn University for the generalization experiment.

Fault Type	Training Data (Artificial Fault Data)	Test Data (Real Fault Data)
Normal state	K002	K001
Outer race fault	KA01	KA04
	KA05	KA15
	KA07	KA16
	-	KA30
Inner race fault	KI01	KI14
	KI05	KI16
	KI07	KI17
	-	KI18
	-	KI21

Fig. 8 presents the comparison of fault diagnosis accuracies of different RF models trained with artificial fault data for real fault data. As shown in Fig. 8, the average fault diagnosis accuracy of Spark-RF is 7.6% lower than the average fault diagnosis accuracy of the RF model, which indicates that the data preprocessing method adopted in [55] is more able to enhance the generalization of the RF model. The average fault diagnosis accuracies of Spark-IRF and MO-TLBO-RF are 2.6% and 13.2% higher than the average fault diagnosis accuracy of Spark-RF, respectively, which means that ensemble pruning can enhance the generalization of the RF model. The reason for the low fault diagnosis accuracy of Spark-RF is that it contains a large number of decision trees that cannot accurately diagnose real bearing faults. The average fault diagnosis accuracy of MO-TLBO-RF is 10.6% higher than the average fault diagnosis accuracy of Spark-IRF, which

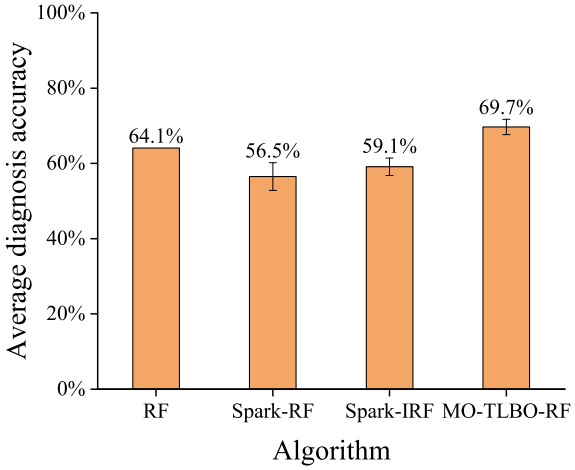


FIGURE 8. Comparison of fault diagnosis accuracies of different RF models trained with artificial fault data for real fault data.

shows that ensemble pruning via MO-TLBO algorithm can better enhance the generalization of the RF model than the sub-forest optimization based on similarity. The average fault diagnosis accuracy of MO-TLBO-RF is 5.6% higher than the average fault diagnosis accuracy of RF, which shows that the proposed MO-TLBO-RF algorithm is more effective.

D. PERFORMANCE ANALYSIS OF MODEL TRAINING AND FAULT DIAGNOSIS

1) VALIDATION OF PARALLELIZATION EFFECTIVENESS

To analyze the influence of parallelization on the training time and fault diagnosis time, the serial MO-TLBO-RF and parallel MO-TLBO-RF are used to train the rolling bearing fault diagnosis model with DATA B, respectively, where the serial MO-TLBO-RF is performed using one CPU core of a single worker node and the parallel MO-TLBO-RF is performed on the cluster with four worker nodes. In addition, the rolling bearing fault diagnosis models are used to diagnose all data of DATA B.

Fig. 9(a) shows the comparison of the model training time of the serial and parallel MO-TLBO-RF. As seen in Fig. 9(a), the serial MO-TLBO-RF takes 2505.7 minutes to train the original RF model and 243.1 minutes to find the optimal sub-forest, while the parallel MO-TLBO-RF only takes 173.6 minutes to train the original RF model and 6.7 minutes to carry out ensemble pruning on the Spark cluster with four worker nodes. Compared with the serial MO-TLBO-RF, the training speed of the original RF model of the parallel MO-TLBO-RF is increased by 13.4 times on average, and the ensemble pruning time is reduced by 97.2% on average. The reason is that the parallel MO-TLBO-RF can fully exploit the computational resources of multiple CPU cores and multiple worker nodes. In addition, the parallel MO-TLBO-RF can effectively utilize all memory resources of a cluster, while the serial MO-TLBO-RF can only use the memory resource of a single worker node. When large-scale data are used to train the original RF model on a single worker node, a large

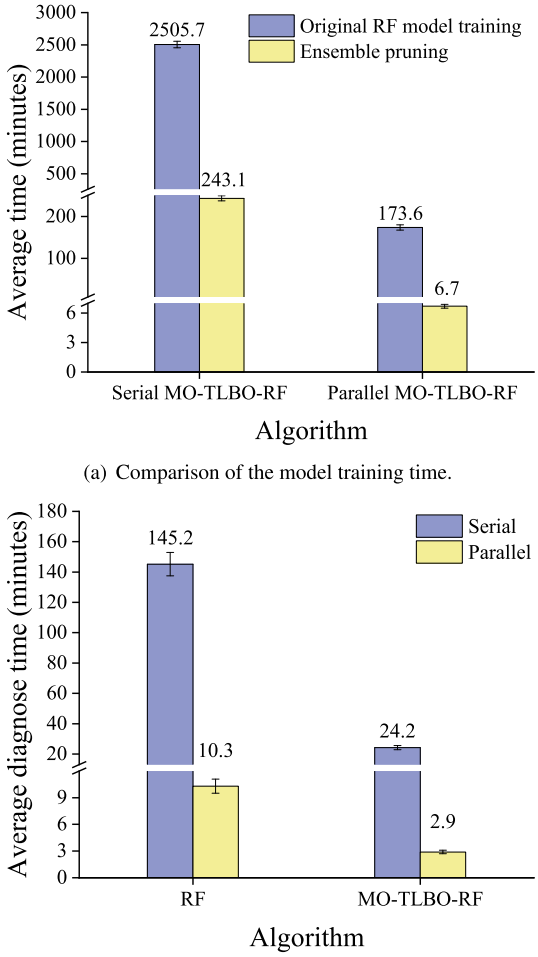


FIGURE 9. Comparison of the serial and parallel fault diagnosis models.

amount of intermediate data will spill onto the disk, which will greatly affect the efficiency of model training.

Fig. 9(b) presents the comparison of fault diagnosis time of the serial and parallel fault diagnosis models. As shown in Fig. 9(b), the fault diagnosis time of the serial RF and MO-TLBO-RF is 145.2 minutes and 24.2 minutes, respectively, and the fault diagnosis time of the parallel RF and MO-TLBO-RF is 10.3 minutes and 2.9 minutes, respectively. The fault diagnosis time of the parallel RF is reduced by 92.9% than the fault diagnosis time of the serial RF, and the fault diagnosis time of the parallel MO-TLBO-RF is reduced by 88.0% than the fault diagnosis time of the serial MO-TLBO-RF. The results show that parallelization can significantly improve the diagnosis speed of rolling bearing fault diagnosis models.

2) INFLUENCE OF THE VOTE SET ON ENSEMBLE PRUNING

To analyze the influence of the vote set on the ensemble pruning time, the parallel MO-TLBO-RF with the vote set and that without the vote set are used to train the rolling bearing fault diagnosis model with DATA B on the Spark cluster,

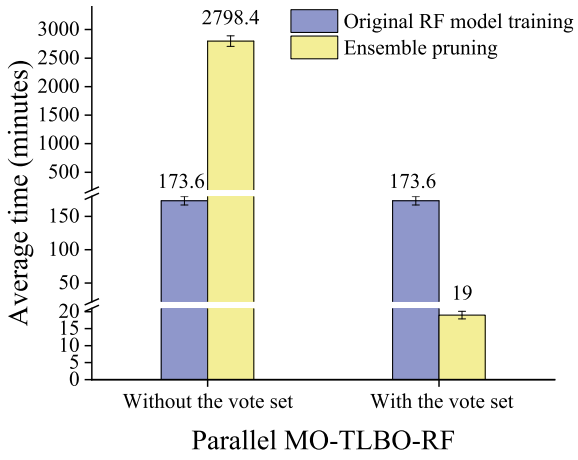


FIGURE 10. Influence of the vote set on the ensemble pruning time.

respectively. For the sake of fairness, the two algorithms use the same validation set to carry out ensemble pruning for the same original RF model.

Fig. 10 shows the influence of the vote set on the ensemble pruning time. As shown in Fig. 10, the parallel MO-TLBO-RF takes 173.6 minutes to train an original RF model, while it takes 2798.4 minutes to carry out ensemble pruning without the vote set for the original RF model. The results show that the computational time of ensemble pruning of RF via MO-TLBO algorithm under the big data environment is very large. However, the parallel MO-TLBO-RF only takes 19 minutes to carry out ensemble pruning with the vote set, and the ensemble pruning time of MO-TLBO-RF with the vote set is reduced by 99.3% than the ensemble pruning time of MO-TLBO-RF without the vote set, which shows that the vote set can greatly reduce the ensemble pruning time. Therefore, the vote set makes it feasible to use multi-objective swarm intelligence optimization algorithms for ensemble pruning under the big data environment.

3) INFLUENCE OF THE SHUFFLE OPTIMIZATION STRATEGY ON ENSEMBLE PRUNING

To analyze the influence of the Shuffle optimization strategy on the ensemble pruning time in the case of using the vote set, the parallel MO-TLBO-RF with the Shuffle optimization strategy and that without the Shuffle optimization strategy are used to train the rolling bearing fault diagnosis model with DATA B on the Spark cluster, respectively.

As seen in Fig. 11, the average ensemble pruning time of the parallel MO-TLBO-RF without the Shuffle optimization strategy and that with the Shuffle optimization strategy are 19 minutes and 6.7 minutes, respectively, i.e., the Shuffle optimization strategy reduces the ensemble pruning time by 64.7% on average. The results show that the Shuffle optimization strategy can significantly reduce the ensemble pruning time.

4) COMPARISON WITH SPARK-IRF

To compare the ensemble pruning time and fault diagnosis time of Spark-IRF and that of the parallel MO-TLBO-RF,

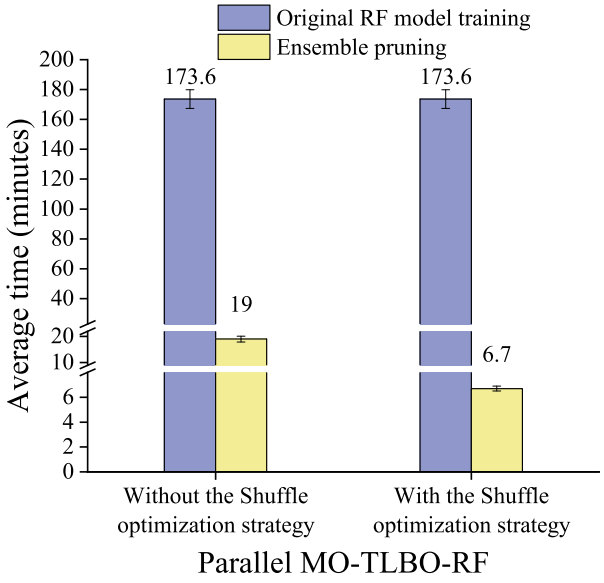
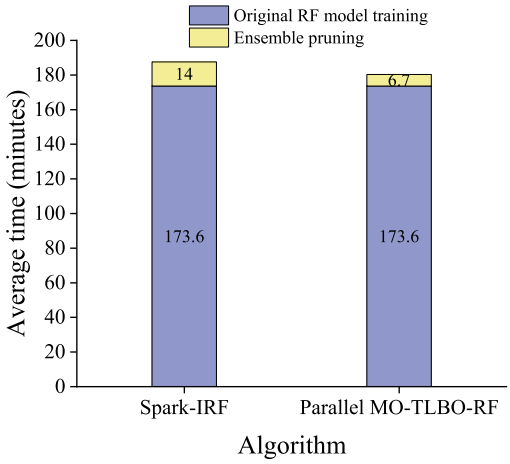


FIGURE 11. Influence of the Shuffle optimization strategy on the ensemble pruning time.

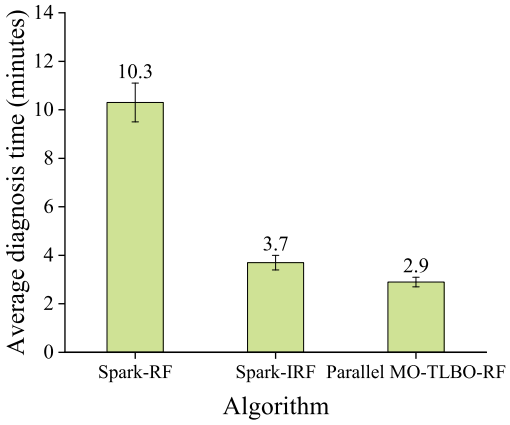
the two algorithms are used to train the rolling bearing fault diagnosis models with DATA B on the Spark cluster.

Fig. 12(a) shows the time spent on original RF model training and ensemble pruning in the process of training fault diagnosis models with Spark-IRF and the parallel MO-TLBO-RF. As shown in Fig. 12(a), the training time of the average original RF model is 173.6 minutes, while the average ensemble pruning time of Spark-IRF and that of the parallel MO-TLBO-RF is 14 minutes and 6.7 minutes, respectively, accounting for 7.5% and 3.7% of the total model training time. The results show that the average ensemble pruning time of Spark-IRF and the parallel MO-TLBO-RF is short. Compared with Spark-IRF, the parallel MO-TLBO-RF has less ensemble pruning time, because the time complexity of Spark-IRF is higher, and the computational time of Spark-IRF is related to the complexity of the decision tree. The complexity of the decision tree is increased with the increase of dataset size. However, the computational time of the parallel MO-TLBO-RF is not related to the complexity of the decision tree, and it is only related to the number of individuals and the size of the vote set. Therefore, the parallel MO-TLBO-RF is more suitable for training rolling bearing fault diagnosis models under the big data environment.

As seen in Fig. 12(b), the average fault diagnosis time of Spark-RF, Spark-IRF, and the parallel MO-TLBO-RF is 10.3 minutes, 3.7 minutes, and 2.9 minutes, respectively. Compared with Spark-RF and Spark-IRF, the fault diagnosis time of the parallel MO-TLBO-RF is reduced by 71.8% and 21.6%, respectively. The results show that the parallel MO-TLBO-RF has a faster fault diagnosis speed than Spark-IRF. This is because the proposed MO-TLBO-RF can find the sub-forest with fewer decision trees, and these decision trees have a shorter classification time. Therefore, the parallel



(a) Comparison of the ensemble pruning time.



(b) Comparison of the fault diagnosis time.

FIGURE 12. Comparison of Spark-IRF and parallel MO-TLBO-RF.

TABLE 8. Comparison of gcForest and parallel MO-TLBO-RF.

Algorithm	Model training time	Fault diagnosis time
gcForest	1835.7 minutes	72.8 minutes
Parallel MO-TLBO-RF	180.3 minutes	2.9 minutes

MO-TLBO-RF is more suitable for diagnosing the rolling bearing faults under the big data environment.

5) COMPARISON WITH gcForest

To further validate the effectiveness of the parallel MO-TLBO-RF, gcForest and parallel MO-TLBO-RF are used to train the rolling bearing fault diagnosis models with DATA B. Note that gcForest only can use all CPU cores of a worker node to train a fault diagnosis model.

Table 8 presents the comparison of the model training time and fault diagnosis time of gcForest and parallel MO-TLBO-RF. As shown in Table 8, the model training time of parallel MO-TLBO-RF is lower 90.18% than the model training time of gcForest, and the fault diagnosis time of parallel MO-TLBO-RF is lower 96.02% than the fault diagnosis time of gcForest. Table 8 demonstrates that the parallel MO-TLBO-RF has better performance than gcForest in terms of the model training time and fault diagnosis time. The reason is that gcForest needs more computational time to build multiple forests, and its diagnosis result is obtained after synthesizing the votes of the multiple forests. However, the parallel MO-TLBO-RF can utilize all worker nodes of the Spark cluster to perform model training and fault diagnosis, it only needs to train a forest, and its diagnosis result is obtained after synthesizing the votes of fewer decision trees.

6) COMPUTATIONAL COMPLEXITY

Table 9 presents the summary of the computational complexity of RF, Spark-RF, Spark-IRF, MO-TLBO-RF, and parallel MO-TLBO-RF. In Table 9, n is the number of decision trees, m is the number of samples, d is the number of feature dimensions, u is the number of worker nodes, v is the number of CPU cores within a worker node, and the specific meaning of x, l, t, and φ can be found in [58]. As can be seen from Table 9, the time complexity of Spark-RF is the lowest, and the time complexity of parallel MO-TLBO-RF is the second lowest, which demonstrates that the proposed parallel MO-TLBO-RF doesn't increase too much complexity while carrying out ensemble pruning of RF.

TABLE 9. Summary of the computational complexity of RF algorithms.

Algorithm	Complexity
RF	$O(n(md\log m))$
Spark-RF	$O(n(md\log m)/uv)$
Spark-IRF	$O((n(md\log m)+x^2lt\varphi)/uv)$
MO-TLBO-RF	$O(n(md\log m)+pw^2)$
Parallel MO-TLBO-RF	$O((n(md\log m)+pw^2)/uv)$

V. CONCLUSION

This paper proposes an RF improved by MO-TLBO algorithm to find the sub-forest with high classification accuracy and less classification time. The proposed MO-TLBO-RF is applied in the rolling bearing fault diagnosis. The diagnosis accuracy is 99.49% using the fault diagnosis model trained with real fault data. In view of the huge computational time of ensemble pruning of RF via MO-TLBO algorithm under the big data environment, a vote set is constructed to improve the fitness evaluation process, which reduces the ensemble pruning time by 99.3%. In addition, MO-TLBO-RF is parallelized on Spark, which reduces the ensemble pruning time by 97.2%, increases the training speed of the fault diagnosis model by 13.4 times, and reduces the fault diagnosis time by 92.9%. In order to effectively reduce the number of Shuffles of MO-TLBO-RF, the Shuffle optimization strategy is proposed, which further reduces the ensemble pruning time by 64.7% in the case of using the vote set.

The size of the validation set will affect the performance of ensemble pruning. In the future, we will try to generate a smaller validation set containing more information for ensemble pruning, which is helpful for finding the sub-forest with better generalization performance in less pruning time.

REFERENCES

[1] B. Liu, K. Li, D.-S. Huang, and K.-C. Chou, "iEnhancer-EL: Identifying enhancers and their strength with ensemble learning approach," *Bioinformatics*, vol. 34, no. 22, pp. 3835–3842, Nov. 2018.

[2] X. M. Chen, M. Zahiri, and S. Zhang, "Understanding ridesplitting behavior of on-demand ride services: An ensemble learning approach," *Transp. Res. C, Emerg. Technol.*, vol. 76, pp. 51–70, Mar. 2017.

[3] Z. Wang, Y. Wang, and R. S. Srinivasan, "A novel ensemble learning approach to support building energy use prediction," *Energy Buildings*, vol. 159, pp. 109–122, Jan. 2018.

[4] G. Xu, M. Liu, Z. Jiang, D. Söffker, and W. Shen, "Bearing fault diagnosis method based on deep convolutional neural network and random forest ensemble learning," *Sensors*, vol. 19, no. 5, p. 1088, 2019.

[5] H.-I. Suk, S.-W. Lee, and D. Shen, "Deep ensemble learning of sparse regression models for brain disease diagnosis," *Med. Image Anal.*, vol. 37, pp. 101–113, Apr. 2017.

[6] B. Bai, G. Li, S. Wang, Z. Wu, and W. Yan, "Time series classification based on multi-feature dictionary representation and ensemble learning," *Expert Syst. Appl.*, vol. 169, May 2021, Art. no. 114162.

[7] W. Yan, G. Li, Z. Wu, S. Wang, and P. S. Yu, "Extracting diverse-shapelets for early classification on time series," *World Wide Web*, vol. 23, no. 6, pp. 3055–3081, May 2020.

[8] G. Li, W. Yan, and Z. Wu, "Discovering shapelets with key points in time series classification," *Expert Syst. Appl.*, vol. 132, pp. 76–86, Oct. 2019.

[9] Y. Li, Y. Song, L. Jia, S. Gao, Q. Li, and M. Qiu, "Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2833–2841, Apr. 2021.

[10] W. Yu and C. Zhao, "Online fault diagnosis for industrial processes with Bayesian network-based probabilistic ensemble learning strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1922–1932, Oct. 2019.

[11] Y. Wang, K. Zhu, M. Sun, and Y. Deng, "An ensemble learning approach for fault diagnosis in self-organizing heterogeneous networks," *IEEE Access*, vol. 7, pp. 125662–125675, 2019.

[12] G. Tsoumakas, I. Partalas, and I. Vlahavas, "An ensemble pruning primer," in *Applications of Supervised and Unsupervised Ensemble Methods*. Berlin, Germany: Springer, 2009, pp. 1–13.

[13] J. Cao, W. Li, C. Ma, and Z. Tao, "Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition," *Inf. Fusion*, vol. 41, pp. 68–79, May 2018.

[14] K. Yu, L. Wang, and Y. Yu, "Ordering-based Kalman filter selective ensemble for classification," *IEEE Access*, vol. 8, pp. 9715–9727, 2020.

[15] H. Guo, H. Liu, R. Li, C. Wu, Y. Guo, and M. Xu, "Margin & diversity based ordering ensemble pruning," *Neurocomputing*, vol. 275, pp. 237–246, Jan. 2018.

[16] P. Zyblewski and M. Woźniak, "Novel clustering-based pruning algorithms," *Pattern Anal. Appl.*, vol. 23, no. 3, pp. 1049–1058, Aug. 2020.

[17] J. Cela and A. Suárez, "Energy-based clustering for pruning heterogeneous ensembles," in *Proc. Int. Conf. Artif. Neural Netw.*, Rhodes, Greece, 2018, pp. 346–351.

[18] Z. Li, K. Goebel, and D. Wu, "Degradation modeling and remaining useful life prediction of aircraft engines using ensemble learning," *J. Eng. Gas Turbines Power*, vol. 141, no. 4, Apr. 2019, Art. no. 041008.

[19] Y. Hayashi and H. Iiduka, "Optimality and convergence for convex ensemble learning with sparsity and diversity based on fixed point optimization," *Neurocomputing*, vol. 273, pp. 367–372, Jan. 2018.

[20] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, no. 1, pp. 239–263, 2002.

[21] D. Li, G. Wen, Z. Hou, E. Huan, Y. Hu, and H. Li, "RTCRelief-F: An effective clustering and ordering-based ensemble pruning algorithm for facial expression recognition," *Knowl. Inf. Syst.*, vol. 59, no. 1, pp. 219–250, Apr. 2019.

[22] J. Valente de Oliveira, A. Szabo, and L. N. de Castro, "Particle swarm clustering in clustering ensembles: Exploiting pruning and alignment free consensus," *Appl. Soft Comput.*, vol. 55, pp. 141–153, Jun. 2017.

[23] X. Zhu, Z. Ni, M. Cheng, F. Jin, J. Li, and G. Weckman, "Selective ensemble based on extreme learning machine and improved discrete artificial fish swarm algorithm for haze forecast," *Appl. Intell.*, vol. 48, no. 7, pp. 1757–1775, Jul. 2018.

[24] X. Zhu, Z. Ni, P. Xia, and L. Ni, "Hybrid ensemble pruning using coevolution binary glowworm swarm optimization and reduce-error," *Complexity*, vol. 2020, Oct. 2020, Art. no. 1329692.

[25] X. Zhu, Z. Ni, L. Ni, F. Jin, M. Cheng, and J. Li, "Improved discrete artificial fish swarm algorithm combined with margin distance minimization for ensemble pruning," *Comput. Ind. Eng.*, vol. 128, pp. 32–46, Feb. 2019.

[26] A. Onan, S. Korukoğlu, and H. Bulut, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Inf. Process. Manage.*, vol. 53, no. 4, pp. 814–833, Jul. 2017.

[27] Z. Ni, P. Xia, X. Zhu, Y. Ding, and L. Ni, "A novel ensemble pruning approach based on information exchange glowworm swarm optimization and complementarity measure," *J. Intell. Fuzzy Syst.*, vol. 39, no. 6, pp. 8299–8313, Dec. 2020.

[28] G. Martinez-Munoz, D. Hernandez-Lobato, and A. Suarez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 245–259, Feb. 2009.

[29] S. E. N. Fernandes, A. N. D. Souza, D. S. Gastaldello, D. R. Pereira, and J. P. Papa, "Pruning optimum-path forest ensembles using Metaheuristic optimization for land-cover classification," *Int. J. Remote Sens.*, vol. 38, no. 20, pp. 5736–5762, Jul. 2017.

[30] X. Zhu, Z. Ni, L. Ni, F. Jin, M. Cheng, and Z. Wu, "Ensemble pruning of ELM via migratory binary glowworm swarm optimization and margin distance minimization," *Neural Process. Lett.*, vol. 52, no. 3, pp. 2043–2067, Dec. 2020.

[31] X. Wen, K. Wang, H. Li, H. Sun, H. Wang, and L. Jin, "A two-stage solution method based on NSGA-II for green multi-objective integrated process planning and scheduling in a battery packaging machinery workshop," *Swarm Evol. Comput.*, vol. 61, Mar. 2021, Art. no. 100820.

[32] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, and H. Zhou, "Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey," *Swarm Evol. Comput.*, vol. 44, pp. 365–387, Feb. 2019.

[33] H. Ma, M. Fei, Z. Jiang, L. Li, H. Zhou, and D. Crookes, "A multipopulation-based multiobjective evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 689–702, Feb. 2020.

[34] S. Zhang, Y. Chen, W. Zhang, and R. Feng, "A novel ensemble deep learning model with dynamic error correction and multi-objective ensemble pruning for time series forecasting," *Inf. Sci.*, vol. 544, pp. 427–445, Jan. 2021.

[35] T.-C. Yang, P.-S. Yu, K.-H. Lin, C.-M. Kuo, and H.-W. Tseng, "Predictor selection method for the construction of support vector machine (SVM)-based typhoon rainfall forecasting models using a non-dominated sorting genetic algorithm," *Meteorological Appl.*, vol. 25, no. 4, pp. 510–522, Oct. 2018.

[36] X. Li, S. Zhang, and K.-C. Wong, "Single-cell RNA-seq interpretations using evolutionary multiobjective ensemble pruning," *Bioinformatics*, vol. 35, no. 16, pp. 2809–2817, Aug. 2019.

[37] C. Qian, Y. Yu, and Z.-H. Zhou, "Pareto ensemble pruning," in *Proc. 29th AAAI Conf. Artif. Intell.*, Austin, TX, USA, 2015, pp. 2935–2941.

[38] A. Peimankar, S. J. Weddell, T. Jalal, and A. C. Lapthorn, "Multi-objective ensemble forecasting with an application to power transformers," *Appl. Soft Comput.*, vol. 68, pp. 233–248, Jul. 2018.

[39] H. Ma, S. Ye, D. Simon, and M. Fei, "Conceptual and numerical comparisons of swarm intelligence optimization algorithms," *Soft Comput.*, vol. 21, no. 11, pp. 3081–3100, Jun. 2017.

[40] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, Mar. 2011.

[41] G. G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 1995–2014, 2019.

[42] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300–323, Oct. 2020.

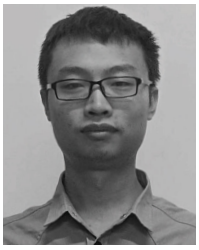
[43] G.-G. Wang, "Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Comput.*, vol. 10, no. 2, pp. 151–164, Jun. 2018.

[44] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114864.

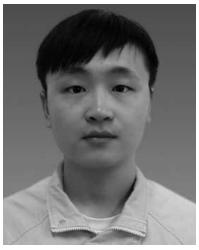
[45] I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, and H. Chen, "RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method," *Expert Syst. Appl.*, vol. 181, Nov. 2021, Art. no. 115079.

[46] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[47] R. V. Rao, "Teaching-learning-based optimization algorithm," in *Teaching Learning Based Optimization Algorithm*. Berlin, Germany: Springer, 2016, pp. 9–39.
[48] J. Huang, Y. Liu, M. Liu, M. Cao, and Q. Yan, "Multi-objective optimization control of distributed electric drive vehicles based on optimal torque distribution," *IEEE Access*, vol. 7, pp. 16377–16394, 2019.
[49] Y. Liu, Q. Cheng, Y. Gan, Y. Wang, Z. Li, and J. Zhao, "Multi-objective optimization of energy consumption in crude oil pipeline transportation system operation based on exergy loss analysis," *Neurocomputing*, vol. 332, pp. 100–110, Mar. 2019.
[50] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[51] W. Shao, D. Pi, and Z. Shao, "A hybrid discrete optimization algorithm based on teaching-probabilistic learning mechanism for no-wait flow shop scheduling," *Knowl.-Based Syst.*, vol. 107, pp. 219–234, Sep. 2016.
[52] E. Sevinc and T. Dökeroğlu, "A novel hybrid teaching-learning-based optimization algorithm for the classification of data by using extreme learning machines," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 27, no. 2, pp. 1523–1533, 2019.
[53] G. Biau, "Analysis of a random forests model," *J. Mach. Learn. Res.*, vol. 13, pp. 1063–1095, Apr. 2012.
[54] S. Raj, D. Ramesh, and K. K. Sethi, "A spark-based apriori algorithm with reduced shuffle overhead," *J. Supercomput.*, vol. 77, no. 1, pp. 133–151, Jan. 2021.
[55] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro, "Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification," in *Proc. Eur. Conf. Prognostics Health Manage. Soc.*, 2016, vol. 3, no. 1, Denver, CO, USA, pp. 1–17.
[56] Z. Wang, Q. Zhang, J. Xiong, M. Xiao, G. Sun, and J. He, "Fault diagnosis of a rolling bearing using wavelet packet denoising and random forests," *IEEE Sensors J.*, vol. 17, no. 17, pp. 5581–5588, Sep. 2017.
[57] S. Ma, W. Cai, W. Liu, Z. Shang, and G. Liu, "A lighted deep convolutional neural network based fault diagnosis of rotating machinery," *Sensors*, vol. 19, no. 10, p. 2381, May 2019.
[58] L. Wan, K. Gong, G. Zhang, X. Yuan, C. Li, and X. Deng, "An efficient rolling bearing fault diagnosis method based on spark and improved random forest algorithm," *IEEE Access*, vol. 9, pp. 37866–37882, 2021.
[59] X. Meng, J. Bradley, B. Yavuz, and E. Sparks, "MLlib: Machine learning in Apache spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, 2016.
[60] Z.-H. Zhou and J. Feng, "Deep forest," *Nat. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2019.
[61] D. Dua and C. Graff. *UCI Machine Learning Repository*. Accessed: Oct. 1, 2021. [Online]. Available: http://archive.ics.uci.edu/ml
[62] R. Razavi-Far, M. Farajzadeh-Zanajni, B. Wang, M. Saif, and S. Chakrabarti, "Imputation-based ensemble techniques for class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 1988–2001, May 2021.
[63] M. Abdel-Basset, D. El-Shahat, I. El-henawy, A. K. Sangaiah, and S. H. Ahmed, "A novel whale optimization algorithm for cryptanalysis in Merkle-Hellman cryptosystem," *Mobile Netw. Appl.*, vol. 23, no. 4, pp. 723–733, Aug. 2018.

LANJUN WAN was born in Hunan, China, in 1982. He received the B.S. and M.S. degrees in computer science and technology from the Hunan University of Technology, Zhuzhou, China, in 2005 and 2009, respectively, and the Ph.D. degree in circuits and systems from Hunan University, Changsha, China, in 2016. He is currently an Assistant Professor with the School of Computer Science, Hunan University of Technology. He has published many research articles in international conferences and journals, such as *JPDC*, *CCPE*, *ParCo*, and *Sensors*. His research interests include industrial big data analysis, industrial equipment fault diagnosis, high-performance computing, and parallel computing. He serves as a Reviewer for the *JPDC*, *CCPE*, *Sensors*, and IEEE Access.



KUN GONG was born in Hunan, China, in 1996. He received the B.S. degree in mechanical engineering from the Hunan University of Technology, Zhuzhou, China, in 2019, where he is currently pursuing the M.S. degree in computer science and technology. His research interests include industrial big data analysis and industrial equipment fault diagnosis.



GEN ZHANG was born in Anhui, China, in 1995. He received the B.S. degree in network engineering from West Anhui University, Lu'an, China, in 2019. He is currently pursuing the M.S. degree in computer science and technology with the Hunan University of Technology, Zhuzhou, China. His research interests include industrial big data analysis and industrial equipment fault diagnosis.

CHANGYUN LI was born in Hunan, China, in 1972. He received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2007. He is currently a Full Professor of computer science and the Dean of the School of Computer Science, Hunan University of Technology, Zhuzhou, China. He has published many research articles in international conferences and journals, such as *JICT*, *JSW*, and *JCP*. His major research interests include industrial big data analysis, industrial equipment fault diagnosis, intelligent information perception and processing technology, the Internet of Things, and software methodology.

ZHIBING WANG was born in Hunan, China, in 1974. He received the M.S. degree in computer science and technology from the Hunan University of Technology, Zhuzhou, China, in 2010. He is currently an Associate Professor with the School of Computer Science, Hunan University of Technology. His research interests include the Industrial Internet of Things, trusted software, and knowledge graph.

XIAOJUN DENG was born in Hunan, China, in 1974. He received the M.S. degree in computer science and technology from the National University of Defense Technology, Changsha, China, in 2004. He is currently a Full Professor with the School of Computer Science, Hunan University of Technology, Zhuzhou, China. He has published many research articles in international conferences and journals, such as *IJSN*, *JCSE*, and IEEE Access. His major research interests include industrial big data analysis, industry equipment health management, the Internet of Things, and image processing.