A Neural Network Slab Quality Prediction Analysis Based on a Hybrid Intelligent Optimisation Algorithm

Yiran Li*, Chunna Zhang

Abstract—**To address the problem of inaccurate prediction of slab quality in continuous casting, an algorithm based on particle swarm optimisation and differential evolution is proposed. The algorithm combines BP neural network prediction method. Firstly, the factors affecting the quality of the slab are analysed and a sufficient number of sample sets are extracted for training; Secondly, the particle swarm optimisation algorithm is optimised, and then the BP neural network is optimised and the chaos mechanism is introduced to increase the continuity of motion of the particles. The population is partitioned hierarchically, and different types of particles are processed separately to improve the global search capability of the algorithm. The differential evolution algorithm is also integrated to optimise cross-selection and increase particle diversity. Finally, a prediction model is built to predict and analyse the sample data. In the experiment, three algorithms are tested for different benchmark functions, namely the comparison of convergence, stability and searchability. The optimisation time comparison is completed, the target curve fit test is performed and the slab quality test results are checked. The experimental results show that the PSO-EIDE algorithm has strong optimisation ability, fast convergence speed and obvious improvement in stability and accuracy.**

Index Terms—**Slab quality, BP neural network, Particle swarm division, Differential evolution, Optimisation capability**

I. Introduction

HOT delivery, hot charging and direct rolling of slabs are currently the established standards for continuous casting production. The primary requirement is to ensure slab quality[1]. Due to the complexity of on-site working conditions, slab quality problems are diverse, and the price of slab detection equipment in the hot state is relatively high. The traditional method of slab quality detection is no longer applicable[2]. In production, large iron and steel companies are gradually introducing sensors, electronic detectors, etc. to increase the degree of automation.Through automated professional equipment, technicians can obtain a large amount of high-quality production data. The effective and reasonable use of these data, the algorithm can greatly improve the prediction effect. The quality of the slab is mainly affected by uneven heat dissipation as it passes through the secondary cooling zone.The impact on other continuous production cannot be ignored, such as hot charging and discharging, continuous casting and rolling, etc. Slab quality defects include surface defects, internal defects and shape defects, which are manifested as surface cracks, bubbles, intermediate cracks, central cracks, corner cracks, off-square, bulging molten steel composition and so on. Therefore, in the continuous casting process, accurate prediction of slab quality defects is key to improving production quality[3].

With the great development of information technology, various new computer technologies are constantly emerging, providing a new solution for improving slab quality.In this paper, a neural network model constructed by a fusion of particle swarm optimisation and differential evolution algorithms is used for simulation. The samples are processed by a neural network, the relevant processing functions are determined and the weights of the neural network are optimised by an intelligent algorithm.

II. Analysis Of Slab Quality Defects

Surface cracking is one of the most common of all slab quality defects. Influencing factors include, but are not limited to, casting process, cooling rate, material properties, etc. Crystal growth forms a nucleus, which is a necessary condition for metal solidification, and the superheat and temperature drop gradient will directly affect the rate of crystal growth. At the same time, the melting point and moisture content of the mould flux will affect the thickness of the shell. As the superheat increases, the crystal grows too fast, which can easily cause the shell to be too thin, and the central segregation and central porosity will be higher. When the superheat reaches a certain level, there is even a risk of cracking and steel leakage. On the premise of ensuring production efficiency, drawing speed is particularly important. A high drawing speed will further increase the central segregation and the central porosity level. If the water ratio is not appropriate, serious sticking breakout will occur[4],[5].

Slab cracking is a gradual process influenced by many factors. Prior to solidification, the slab is subjected to tensile stress in the process of continuous movement. If it exceeds a certain critical value, internal cracks will occur. The external cause of cracking is mainly the action of various forces, and the internal cause is influenced by the sensitivity of the structure. Therefore, consideration of slab quality includes not only the endogenous structure, solidification conditions

Manuscript received May 1, 2024; revised January 19, 2025.

Yiran Li is an associate professor at the School of Applied Technology, University of Science and Technology Liaoning, Anshan 114051, Liaoning, China.(e-mail: lyr7879@163.com).

Chunna Zhang is an associate professor at the School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan 114051, Liaoning, China.(e-mail: zcn1979@163.com).

and high temperature mechanical properties, but also the operation of machinery and equipment.

III. Neural Network Model

The traditional way to build a slab quality prediction model is to use an expert system that relies on data and experience that is repeatedly verified. The system completes the logical judgement by setting various parameters. The system is highly dependent on the production process, steel grade and plant conditions, and is poorly adaptable to emergencies. In the case of incomplete data and high complexity, neural network can realise the reasonable mapping from input to output, and solve the problem of expert system, such as the problem of excessive dependence on data[6].

Before establishing the neural network model, a hypothesis is made in this paper, i.e. the operation of the continuous casting equipment is not considered. Eliminate dependence on specific steel grades, processes and equipment. It is assumed that the continuous casting plant is operating normally and that the formation of internal cracks is entirely determined by the process and chemical composition. The real-time data is derived from steel grade, chemical composition, casting speed, superheat, cooling water temperature, water volume, shell surface temperature, shell thickness and other process conditions and information.

In order to improve the performance of the neural network, the data obtained by manual verification is used as standard data, i.e. the expected value, in the learning process. Comparing the expected value with the measured data in production, if there is a large deviation, delete the value; the value with the smaller deviation is retained and merged with the previous data to adjust the weight of the neural network. If the actual measured value is the same as the manual check value, it does not need to be processed.

After data collection, the original data dimension is different. Some data values are very large, while others are very small. If they are applied without processing, it is easy to cause a large deviation in the output results.Therefore, it is necessary to normalise the samples and limit the values between $[0,1]$. The maximum and minimum normalisation functions are used in this paper as follows:

$$x_i' = \left\{ \begin{array}{ll} \dfrac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}}(x_{\max} - x_{\min}) & \le \theta \\ \dfrac{x_i - x_{\min}}{x_{\max} - x_{\min}} & > \theta \end{array} \right\} \qquad (1)$$

In the formula, $x = \{x_1, x_2, \cdots, x_n\}$ is the original sample data, n is the dimension, $x_{\max}$、$x_{\min}$ are the maximum and minimum values of the sample respectively, θ represents the stability threshold, and $\theta = 25$ ℃ in the text.

A. BP neural network structure

BP (back-propagation) neural network is an error back-propagation algorithm[7]. It can solve non-linear problems and is widely used in engineering, but it is essentially a descending gradient algorithm. This method has some unavoidable shortcomings, such as slow convergence speed, strong dependence on initial weights and thresholds, and inability to guarantee global optimality[8],[9].Particle Swarm Optimisation (PSO) is introduced in this paper. It is a global random search algorithm that can optimise the weights and thresholds and improve the convergence speed. When the particle swarm is initialised, the weights are assigned to each particle; in dimensional space, the particle represents the weight[10-12]. The fitness function of the algorithm is set to mean square error, and the minimum fitness value is the optimal connection of the neural network.The corresponding network topology structure is as follows:

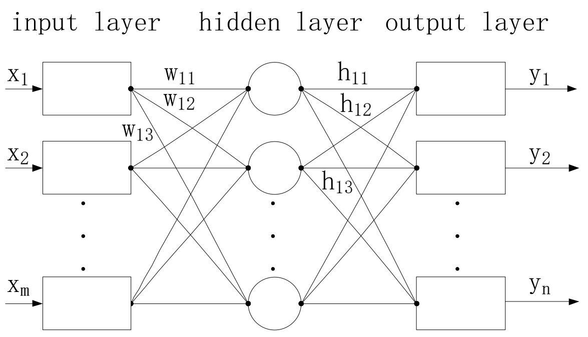


Fig. 1. Topological structure diagram of neural network

B. Parameter Design

In this paper, a three-layer network structure is established and the output vector is $y = [y_1, y_2 \cdots y_n]$, which is used to identify the defect level of the slab. The input vector is $x = [x_1, x_2 \cdots x_m]$ to identify the associated process parameters.

In terms of convergence speed, learning speed is the key factor and setting too small will slow down the convergence speed. If the setting is too large, the optimum value cannot be obtained and convergence cannot be achieved. In this paper, considering the complexity of the real environment, the main consideration is the case of non-convergence, so some unnecessary parameters need to be eliminated.

The three nodes of the output layer are the corner crack level, the intermediate crack level and the central crack level, and the value interval is 0.5 units.The design of the input parameters reflects the practicality as much as possible, including chemical parameters, equipment parameters and so on.Parameters include C content, P content, S content, superheat, casting speed, shell thickness, surface temperature of each section of the slab, etc.Choosing the number of nodes in the hidden layer is more complicated and there is currently no theoretical guidance. If too few nodes are designed, it is easy to have a poor training effect and affect the overall performance of the algorithm. If there are too many nodes, the training time will be longer and fault tolerance will be compromised. According to the Komlogorov theorem, the relationship between the number of nodes in the input layer and the number of nodes in the hidden layer is as follows:

$$m \le 2n + 1 \qquad (2)$$

In the formula, m is the number of hidden layer nodes and n is the number of input layer nodes.

For the grade of slab defects, the number of nodes in the output layer is set to 3, the number of nodes in the input layer is 14, and the number of nodes in the hidden layer is 16.

When using BP neural network modelling, rational allocation of weight coefficients is particularly important, and the parameter adjustment method often used is trial and error.This method is applicable to the self-balancing controlled object, which is an extension of the critical proportionality method.The key to this method is to see if you can run the system and change the parameters at the same time until you get satisfactory results. However, this method relies on empirical values and has poor general applicability. Therefore, this paper uses an improved particle swarm optimisation algorithm to adjust the weighting coefficient[11],[12].

C. Particle Swarm Optimization Algorithm

PSO is an algorithm that uses inter-individual communication and cooperation to find the optimal solution.[13][14]. In Q -dimensional space, initialize n particles $\{x_1, x_2, \cdots, x_n\}$, where the position of the i particle in space is $x_i = \{x_{i1}, x_{i2}, \cdots, x_{iQ}\}$, the particle velocity is $v_i = \{v_{i1}, v_{i2}, \cdots, v_{iQ}\}$, $i = 1, 2, \cdots, n$, and direction of travel is random[15][16]. The trajectory of each particle follows two positions: the current optimal extremum Pb and the overall optimal extremum Gb .The fitness values of the particles are calculated in iterations and the velocity and position of the particles are updated, and Pb and Gb are updated after each iteration. After repeated iterations until the termination condition is satisfied, the optimal solution is obtained[17-19]. The equations related to the particle state are as follows:

$$v'_{iq} = \alpha v''_{iq} + \beta^0 (x_{Pb} - x_{iq}) + \beta^1 (x_{Gb} - x_{iq}) \tag{3}$$

$$x'_{iq} = x''_{iq} + v'_{iq} \tag{4}$$

In the formula, a is the inertia weight , it is used to balance the relationship between the velocities of different generations of particles. β^0、$\beta^1 \in [0,1]$, is randomly distributed.

The premature problem of PSO is mainly reflected in the convergence speed of the algorithm[20]. As far as the essence of the algorithm is concerned, in the process of evolution, the path of the particle follows the trajectory of the optimal particle. If there are particles with poor performance, their speed will be greatly reduced, and at the same time they will oscillate in a certain dimension. Finally, it is easy to fall into the local optimal constraint and premature phenomena occur [21-23]. To solve this problem, this paper starts from two aspects: first, the chaotic mechanism is introduced to improve the ergodic ability of the particles and increase the diversity of solutions; second, the particles with poor performance are mutated to get rid of oscillations.

D. Chaos Mechanism

Chaos is a widespread phenomenon in nature. It belongs to the non-linear category. It has the properties of ergodicity and regularity. It is sensitive to initial conditions and can search all internal states in a given domain according to the law without repetition[24]. In this way, the nature of chaos can be used to optimise the search. The search steps are as follows:

1) Define the initial region, set the N dimensional initial state vector $R_0 = (R_{01}, R_{02}, \cdots, R_{0N})$, where the values in R_0 are adjacent and have very small differences.

2) Using the logistics equation to calculate the initial vector R_0, a chaotic sequence $c_1, c_2, \cdots, c_n$ is generated. Here, after several iterations, the system will be completely in a chaotic state. The vector layer can be expressed as:

$$c_{i+1} = c_i (1 - c_{i-1}) \lambda \tag{5}$$

In the formula, λ is the iterative control parameter.

3) Assuming the spatial particle X_i, use equation (5) to obtain a better position for X_i, denote X_i'.

$$X_i' = r \cdot rnd \cdot c_j + X_i \tag{6}$$

In the formula, r is the active radius of particle X_i, $rnd \in [-1,1]$, $j \in [0, n]$.

The main idea of particle swarm optimization algorithm based on chaotic mechanism is as follows: on the one hand, the position and velocity of particles are initialised by chaotic sequence. Because of its ergodicity, it not only maintains the diversity of particles, but also improves the search ability of particles. Moreover, the chaotic state can make the motion of the particles persistent.

Particle chaos initialization : give initial values to X_i in Equation (6), and re-correct the velocity in particle swarm iteration:

$$\begin{aligned} v'_{i,j}(t+1) &= \alpha v''(t) + \beta^0(t)(x_{lb}(t) - x_{i,j}(t)) \\ &+ \beta^1(t)(x_{gb}(t) - x_{i,j}(t)) \end{aligned} \tag{7}$$

In the formula, α is a constant (0,1] , β is a random number with a normal distribution of $N[0,1]$, $i \in [1, n]$, $j \in [1, m]$, n is the number of particles, and m is the spatial dimension. About $v''(t)$:

$$v''(t) = \begin{cases} v_{i,j}(t) & \theta = 0 \\ N[0,1] \cdot \delta \cdot \tilde{v} & \theta = 1 \end{cases} \tag{8}$$

$$\theta = \begin{cases} 0 & f(x_{gb}(t-1)) > f(x_{gb}(t)) \\ 1 & f(x_{gb}(t)) = f(x_{gb}(t-1)) = \\ & \cdots = f(x_{gb}(t-5)) \end{cases} \tag{9}$$

In the formula, $\tilde{v} = v_{\max} \cdot c_i / 1.1$, $\delta = f(x_{gb}) - f(x_T)$, c_i is a new chaotic sequence, $f(x_{gb})$ is a satisfactory solution, $f(x_T)$ is the target solution.

IV. Differential Evolution Solution Based on Superior and Inferior Particles

The particle swarm algorithm has obvious advantages, such as strong global search ability and fast convergence speed. However, the algorithm is prone to premature convergence, and as the number of iterations increases, if the diversity of particles in the population does not improve, the particles will gradually converge to the global optimal position. However, if this position is a local extremum, it will be trapped in the local optimal constraint, reducing the accuracy of the algorithm. To solve this problem, this paper proposes a differential evolution algorithm based on good and bad particles.Introducing a strategy for handling the difference between good and bad particles, while conducting a global optimal search, utilizing the cross mutation operation in differential evolution algorithm to enhance the diversity of particles.

A. Division of superior and inferior particles

In the operation of the algorithm, the performance of the particles after each iteration is different, and the performance of the search ability is different. Fitness values are commonly used to measure the search ability and accuracy of algorithms. Particles with good fitness are more likely to reach the optimal position and lead other particles in the population to move to that position; and particles with poor fitness are far from the optimal position, making it easy for them to break out of local optimal constraints.

$$N_1 = \frac{1}{6}n + n(0.45 - \frac{k}{K})^3$$
$$N_2 = \frac{5}{6}n + n(0.45 - \frac{k}{K})^3 \qquad (10)$$

In the formula, N_1 is the dividing line between superior and inferior swarms, N_2 is the dividing line between inferior and random swarms, n denotes the total number of particles, and k and K denote the number of current iterations and maximum iterations, respectively.

After adopting the stratification strategy, $\{x_1, x_2, \cdots, x_n\}$ in the population completes the three-part division into $\{x_1, x_2, \cdots, x_{N1}\}, \{x_{N1+1}, x_{N1+2}, \cdots, x_{N2}\}, \{x_{N2+1}, x_{N2+2}, \cdots, x_n\}$. The particles in the superior group are updated by a differential evolution algorithm, which is beneficial to reduce the risk of particles falling into local optimal constraints. In the inferior group, the optimised particle swarm optimisation algorithm achieves fast convergence; the particles in the random group have poor activity and need to be eliminated, and then randomly generated particles are used to increase population diversity.

B. *Differential Evolution*

The basic idea of the Differential Evolution Algorithm (DE) is similar to that of the Genetic Algorithm, i.e. using the mutation operation to generate new individuals and then implementing crossover and selection operations. After continuous iterative evolution, the global optimal solution is searched in the solution space by vector, but the two are defined differently[25],[26]. DE implements mutation operation through differential strategy, which is different from genetic algorithm. It uses the characteristics of the population to improve the optimisation ability of the algorithm[27]. In the optimisation problem, the non-linear minimisation problem is solved as an example. The basic mathematics is as follows:

$$\begin{cases} \min f(x_1, x_2, \cdots x_Q) \\ x_j^L \le x_j \le x_j^U, j = 1, 2, \cdots, Q \end{cases} \qquad (11)$$

In the formula， Q is the solution space dimension, and U、L are the upper and lower bounds on the values of the component x_j , respectively.

The basic steps of the differential evolutionary algorithm are as follows:

1) Population Initialization

If the population size is set to N_p , then the k-th generation individual i can be expressed as $\{x_{j,i} \mid x_{j,i}^L \le x_{j,i}^L x_{j,i} \le x_{j,i}^U$, $i = 1, 2, \cdots, N_p$, $j = 1, 2, \cdots, Q\}$. In the algorithm, each generation population is composed of N_p dimensions and the vector of Q, then the initial population $x_{j,i}(0)$ can be expressed as :The initial generation of the population $x_{j,i}(0)$ can be denoted as:

$$x_{j,i}(0) = x_{j,i}^L + rand(0,1) \cdot (x_{j,i}^U - x_{j,i}^L) \qquad (12)$$

In the formula, $rand(0,1)$ is a random number, $\in (0,1)$, uniformly distributed.

2) Mutation operation

For each target individual $x_{j,i}$ of the population, DE generally adopts a difference strategy to implement the mutation operation[28], that is, two individuals in the population are randomly selected to be vectorially fused with the to-be-mutated individuals, which in turn produces a new mutated individual $V_{j,i}(k+1)$ .The scaling factor F is set in the algorithm. This parameter affects the updating of the population in the iteration. It is set as an adaptive method, i.e. the initial search area is large and the F value is large. As the algorithm progresses, the F value should gradually decrease from large to small searches.

$$F = F_{\max} - \frac{k(F_{\max} - F_{\min})}{K} \qquad (13)$$

$$V_{j,i}(k+1) = x_{r1}(k) + F \cdot (x_{r2}(k) - x_{r3}(k)) \qquad (14)$$

In the formula, $V_{j,i}(k+1)$ is the new mutant individual; F is the scaling factor, $\in (0,1)$; r_1、r_2、$r_3 \in [1, N_p]$ is used to control the vector difference of the randomly selected individuals , which is also different from the individuals to be mutated, and $r_1 \ne r_2 \ne r_3 \ne j$.

In the iteration of the algorithm, whether it is a randomly selected individual or a newly generated individual, both must be effective and conform to the established constraints, $\{V_{j,i}(k+1) \mid x_{j,i}^L \le V_{j,i}(k+1) \le x_{j,i}^U\}$ [29],[30]. After the DE mutation operation, the particles will change in a certain order, and the ascending mutation operation is taken for a real number of individuals, for example, $X = \{3.2, 2.8, 1.3, 1.9, 7.5, 5.3\}$ is $X = \{1.3, 1.9, 2.8, 3.2, 5.3, 7.5\}$ after the mutation operation, and the related encoding can be expressed as $(4, 3, 1, 2, 6, 5)$.

3) Cross Operation

The individual to be tested $V_{j,i}^{'}(k+1)$, is obtained through the cross transformation of the individual, then the corresponding $V_{j,i}^{'}(k+1) = \{V_{1,i}^{'}(k+1), V_{2,i}^{'}(k+1), \cdots, V_{Q,i}^{'}(k+1)\}$. When the intermediate is cross-transformed, the individual to be tested contains at least one vector by a random strategy and is generated by $x_{j,i}$. The cross must follow the following formula:

$$V_{j,i}^{'}(k+1) = \begin{cases} V_{j,i}(k+1) & rand(j) \le CR \, or \; j \in [1, Q] \\ x_{j,i}(k) \end{cases} \qquad (15)$$

In the formula， j is a bounded random number to ensure that the individual to be tested will not completely detach from the target individual $x_{j,i}$; $CR \in [0,1]$, called the crossover probability, is used to regulate the difference between the newly generated individual and the original individual, and is a random number.

In order to eliminate the repeated particles in the n-th generation population, the mapping operation is performed by local crossover. The particles of the n-th generation population are crossed, and the local positions to be swapped are selected, and then the positions of the two elements are swapped. The particles in the local area are mapped, and then the repeated elements in the offspring are eliminated to maintain the diversity of the offspring. The corresponding figures are as follows.

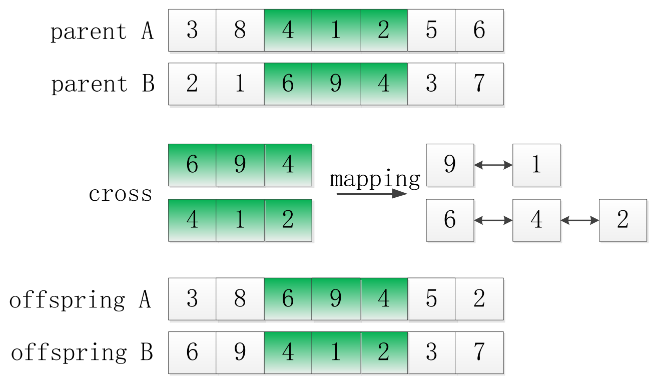


Fig. 2. Particle Cross Mapping

4) Selection Operation

By comparing the fitness values of the test individual and the target individual, the smaller one is retained to become the new individual of the next generation.The greedy algorithm is used to determine the new individual:

$$x_{j,i}(k+1)=\begin{cases}V_{j,i}^{'}(k+1) & f(V_{j,i}^{'}(k+1))\le f(x_{j,i}(k))\\ x_{j,i}(k) & f(V_{j,i}^{'}(k+1))> f(x_{j,i}(k))\end{cases} \tag{16}$$

In the DE algorithm iteration, after the individual is initialised, the mutation and crossover operations are completed. According to the fitness value, the eligible individuals are selected to enter the next iteration and the inferior individuals are eliminated. This operation is repeated in each iteration process and finally the optimal individual is obtained, which is the optimal solution of the algorithm.

C. *PSO-EIDE algorithm process*

PSO is combined with DE algorithm and named PSO-EIDE after optimisation. After initialising the position and velocity of the particles, the algorithm calculates the fitness of the particles. According to the fitness value of the particles, the population is divided into three levels: superior group, inferior group and random group. The three populations are processed separately, and the velocity and position of the particles are updated once by iteration, and the fitness value is recalculated to divide the population until the optimal solution is obtained or the maximum number of iterations is satisfied.

Input: steel grade, chemical composition, drawing speed, superheat, cooling water temperature, water volume, shell surface temperature, shell thickness.

Output: n randomly initialised particles meeting the requirements

```
1. Initialization Parameter List
// The sample data is sorted, the particles are initialised and the clustering centre vector is determined for each particle.
2. for(int i=0;i<=n;i++){
// The initial fitness value is calculated and the fitness value of each particle is calculated according to the fitness function. The best position of the particle is selected as the initial iteration position.}
3.Selection of locally and globally optimal individuals.
4. while( Current number of iterations is less than the maximum number of iterations){
// Calculate the fitness value and divide the population according to equation (10).
for(int i=0;i<=n;i++){
// Each particle is treated separately.
if(The particles are in the superior group)
//Update particle velocity and position using DE algorithm.
else if(The particles are in the inferior group)
//Update particle velocity and position using PSO algorithm.
else
//Correct the velocity and position of the particle using equation (5).
//Calculate the fitness value }
5.Generate a new generation of individual $x_i^t$ .
6. If(Comparison of the current optimal solution and the temporary optimal solution)
// Update the current optimal solution.
7. if( Reference Particle Swarm Optimization to Set Iteration Termination Conditions)
    break;
}
```

D. *BP Neural Network Optimization*

In this paper, the global search performance of the PSO-EIDE algorithm is used to optimise the training of the neural network, and the speed and position of the particles are redesigned. Setting A^1, A^2 as the weight matrices between the input layer and the hidden layer, respectively, The i particle can be defined as $\{A_i^1, A_i^2\}$, so that the particle velocity and position equations can be redefined as:

$$\begin{aligned}v_{i,j}(t+1)=\alpha v_{i,j}(t)+\beta^0(t)(x_{lb}(t)-A_i^1(t))\\ +\beta^1(t)(x_{gb}(t)-A_i^2(t))\end{aligned} \tag{17}$$

$$X_{i,j}(t+1)=X_{i,j}(t)+cv_{i,j}(t+1) \tag{18}$$

In the formula, α is a constant of $(0,1]$, β is a random number of normal distribution $N[0,1]$, $x_{lb}(t)$ is a local optimum, $x_{gb}(t)$ is a global optimum, c is a speed factor, $\in[0,1]$. $i\in[1,n]$, $j\in[1,m]$, n is the number of particles, and m is the spatial dimension.

According to the characteristics of slab quality determination, the structure of the neural network is determined, the corresponding weights of the neurons are coded, and the PSO-EIDE algorithm is used to iterate. The value of the fitness function is the mean square error, and it is continuously iterated to finally meet the accuracy requirements. The specific steps are as follows:

Step 1: The samples are collected and pre-processed, the threshold of the BP neural network is initialised, and the weights and thresholds are encoded to obtain the initial population;

Step 2: The velocity and position of the particles are initialised using formulae (17) and (18) and the fitness value is calculated;

Step 3: Update the extreme value and update the speed and position again to see if the conditions are met;;

Step 4: The weights and thresholds are obtained and the mean square error of the BP neural network is calculated to determine if the conditions are met. If not, proceed to step 3, otherwise exit.

V. Experimental Analysis

Combined with the field data of a steel plant, BP neural network: 14 equipment and process parameters are introduced, that is, 14 dimensional vectors, 3000 groups of samples are extracted, the output vector dimension is 3, and the data normalization algorithm reference formula (1). PSO-EIDE algorithm uses five classic test functions for analysis.Set the population size is 40, the speed factor $c = 0.6$, the inertia weight α decreases from the maximum value of 0.85 to 0.35, the maximum iteration number is 400, the scaling factor $F = 0.5$, and the crossover probability $CR = 0.4$ .The selected samples are used to analyse PSO, PSO-DE combined with differential evolution, and PSO-EIDE algorithm using the pros and cons division strategy.The definitions and value ranges of the five benchmark functions are given below.

1) The value range of function $Schwefel$ is $[-10,10]^D$, and the formula is as follows:

$$Schwefel(x) = \sum_{i=1}^{D}(x_i \sin\sqrt{|x_i|})$$

2) The value range of function $Rosenbrock$ is $[-10,10]^D$, and the formula is as follows:

$$Rosenbrock(x) = \sum_{i=1}^{D}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$

3) The value range of function $Rastrigin$ is $[-5.12, 5.12]^D$, and the formula is as follows:

$$Rastrigin = \sum_{i=1}^{D}[x_i^2 - 10\cos 2\pi x_i + 10]$$

$$x_i = \begin{cases} k_i & |k_i| < 0.5 \\ round(2k_i)/2 & |k_i| \geq 0.5 \end{cases}$$

4) The value range of function $Griewank$ is $[-600, 600]^D$, and the formula is as follows:

$$Griewank = \frac{1}{4000}\sum_{i=1}^{D}(x_i)^2 - \prod_{i=1}^{D}\cos(x_i/\sqrt{i}) + 1$$

5) The value range of function $Patho\log ical$ is $[-100,100]$, and the formula is as follows:

$$Patho\log ical = \sum_{i=1}^{n-1}(0.5 + \frac{\sin\sqrt{(100x_i^2 + x_{i+1}^2)} - 0.5}{1 + 0.001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2})$$

6) The value range of function $schaffer$ is $[-5.12, 5.12]^D$, and the formula is as follows:

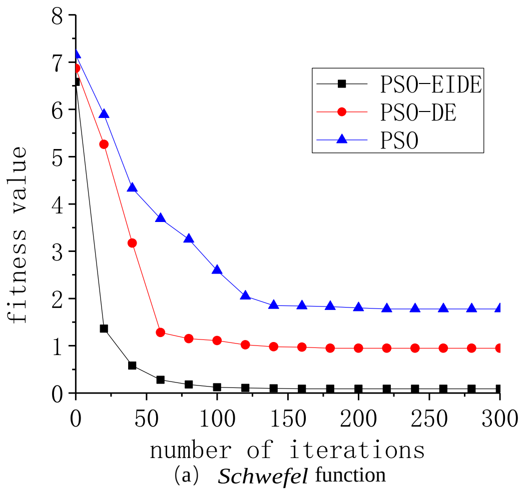
$$schaffer = \frac{\sin^2\sqrt{\sum_{i=1}^{n}x_i^2 - 0.5}}{(1 + 0.001(\sum_{i=1}^{n}x_i^2))^2} + 0.5$$

TABLE 1 COMPARISON OF BENCHMARK FUNCTION TEST VALUES

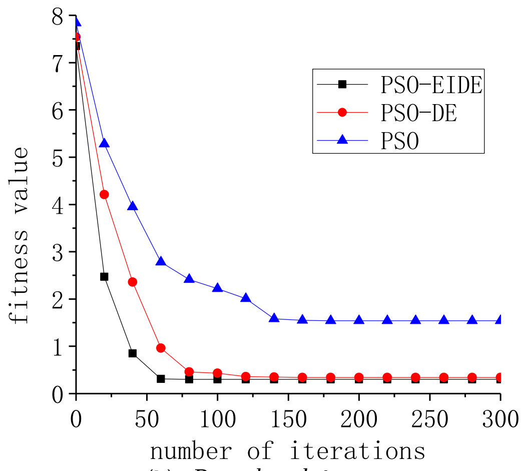
benchmark function	algorithm	optimal value	Worst value	variance
$Schwefel$	PSO	1.644	18.452	13.217
	PSO-DE	0	2.415	0.328
	PSO-EIDE	0	0	0
$Rosenbrock$	PSO	0	4.242	1.308
	PSO-DE	0	3.581	0.875
	PSO-EIDE	0	7.2×10^{-24}	6.2×10^{-51}
$Rastrigin$	PSO	2.391	12.842	7.331
	PSO-DE	0	0.894	1.241
	PSO-EIDE	0	0	0
$Griewank$	PSO	2.895	16.995	10.472
	PSO-DE	0.113	3.224	4.847
	PSO-EIDE	0	0	0
$Patho\log ical$	PSO	0	3.368	5.127
	PSO-DE	0	1.114	2.752
	PSO-EIDE	0	0	0
$schaffer$	PSO	0	2.368	4.128
	PSO-DE	0	0.351	1.173
	PSO-EIDE	0	0	0

From the analysis of Table 1, for six benchmark functions, five of them find the global optimal solution and only $Rosenbrock$ falls into the local optimal solution. From the analysis of its experimental results, the average optimisation accuracy reaches 10-25,which is much higher than the accuracy of other algorithms.The specific analysis shows that the average value of the PSO algorithm in the five benchmark functions is greater than 0, which easily falls into the local optimum, and the global optimum solution cannot be obtained. At the same time, the variance results are too large, indicating that the stability of the algorithm is poor. After the introduction of the differential evolution algorithm, the optimisation capability of the PSO-DE algorithm has been enhanced, and all aspects of the indicators have been significantly improved. For most functions, better test results can be obtained, but the optimisation effect of $Griewank$ function does not meet the requirements, and the convergence speed is not optimal. In the test, in the improved PSO-EIDE algorithm, in terms of average value, the results of three benchmark functions are 0, and two are minimum values, indicating that the algorithm has good robustness, and the test optimal value of the algorithm is 0, and the variance value reaches 0 twice, indicating that the algorithm has good stability. At the same time, a comprehensive comparison of several algorithms can also conclude that the optimal value of the PSO-EIDE algorithm is not much different from the worst value, and the variance is much smaller than other algorithms, which also shows that the optimised algorithm has stronger stability and optimisation ability.

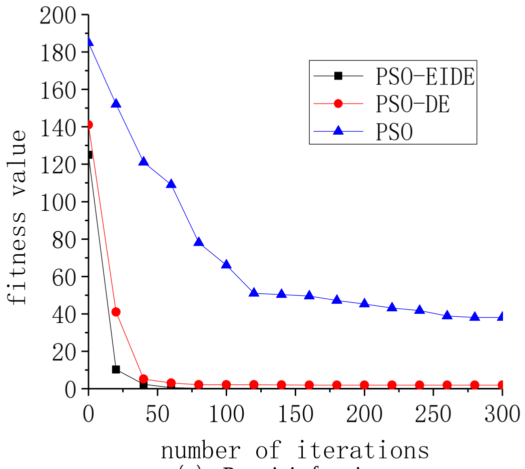
The optimisation time of the three algorithms on different use cases is compared.Each test is run 20 times independently and the time after execution is recorded and averaged. The normalised information processing method is used to test the three algorithms,and the correlation coefficients of different samples are obtained, and the target curve is fitted. The correlation coefficient is linear with the target curve, and the closer the value, the better the effect. Figure 3 is the normalised target curve comparison.



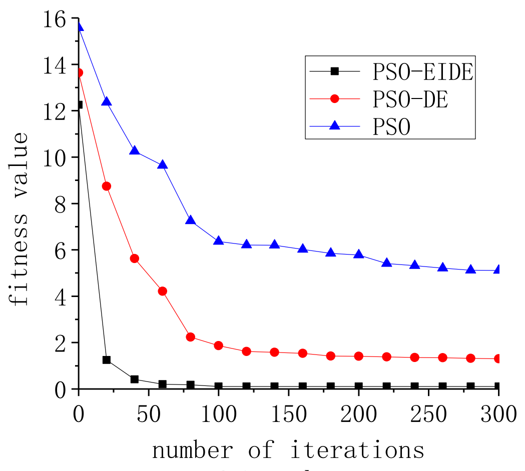
(a) *Schwefel* function



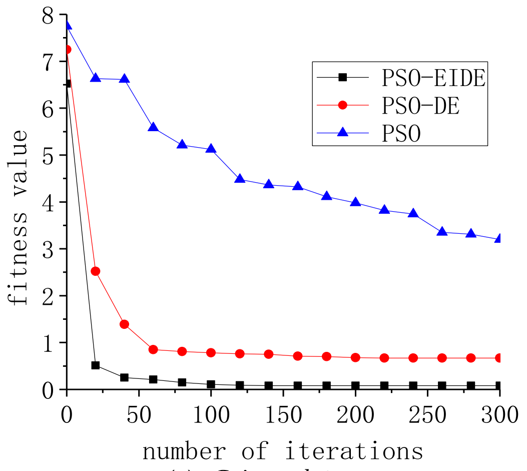
(b) *Rosenbrock* function



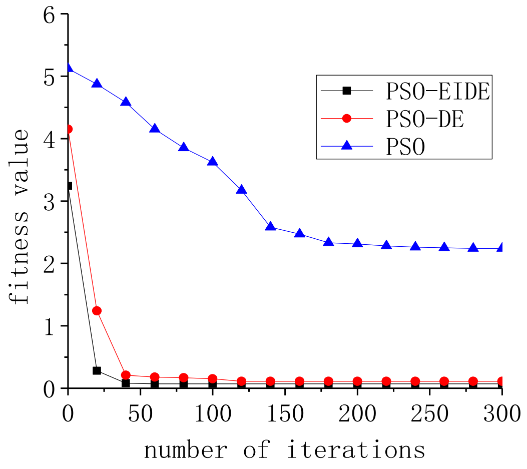
(c) *Rastrigin* function



(d) *Griewank* function



(e) *Griewank* function



(f) *Griewank* function

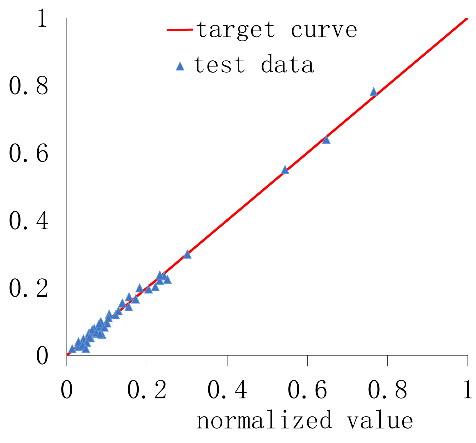
Fig. 3. Convergence of benchmark function

From the analysis of Figure 3, it can be seen that PSO-EIDE has the fastest convergence speed and stable performance among the three algorithms in the test of six benchmark functions. In the early stages of iteration, PSO-EIDE converges faster than the other two algorithms. When the number of iterations is 20, the PSO-EIDE basis reaches a stable value and gradually tends to be stable. In contrast to the other two algorithms, PSO-DE, due to the integration of DE algorithm in the algorithm, the particle diversity is guaranteed and it is easier to find the optimal solution. In addition to the two benchmark functions of A and B, the convergence of other functions is close to that of PSO-EIDE, while the initial convergence performance of the PSO algorithm is poor;In the middle of the iteration, the PSO algorithm is still in the process of optimisation. The PSO-DE algorithm completes fast convergence, but is not stable and does not fully converge. In the later stage of the iteration, the improved algorithm can remain stable and quickly find the global optimal solution, while the optimisation results of the other two algorithms are not ideal. Especially in the later stage of the PSO, there are still some function values that do not converge, indicating that the algorithm has fallen into the local optimum bounds. The improved algorithm can converge quickly, which shows that in the process of algorithm advancement, because the DE algorithm adopts the local crossover strategy in the cross selection, the algorithm avoids the repeated samples and the population diversity is guaranteed. The selection of the superior and inferior particles also makes the particle vitality continue, and the global search ability is strong, which can prevent the algorithm from falling into the local optimum prematurely.

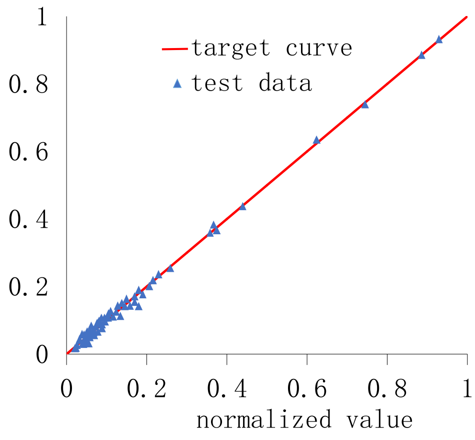
The algorithm is used to test the optimisation time and three algorithms of PSO, PSO-DE and PSO-EIDE are selected for comparison. The process is set up as a single thread, and the population and iteration times are consistent with the benchmark function test. Each test is run 10 times independently, the time after execution is recorded and the average is taken.

TABLE 2 COMPARISON OF OPTIMIZATION TIMES (UNIT: S)

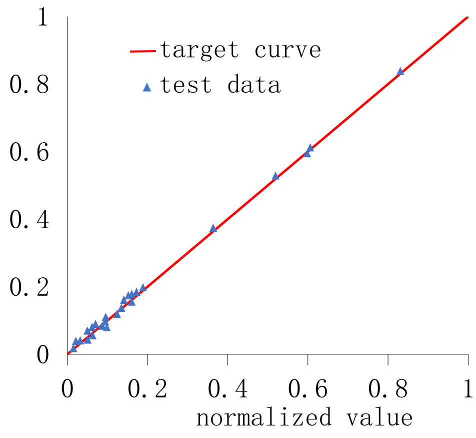
Example	PSO	PSO-DE	PSO-EIDE
E1	52	1214	714
E2	23	419	223
E3	11	390	198
E4	45	982	473
E5	19	457	231
E6	39	816	384



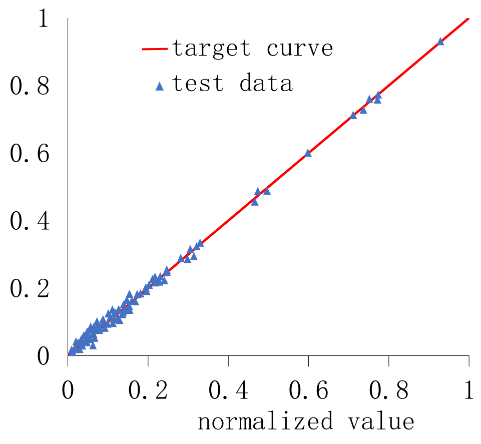
(a) Test data 1



(b) Test data 2



(c) Test data 3



(d) Test data 4

Fig. 4 Comparison of normalized target curves

From Table 2 it can be seen that the PSO algorithm takes the shortest time of the six examples and has significant advantages over the other two algorithms, while the PSO-DE algorithm takes the longest time and the PSO-EIDE algorithm is in between the two. The PSO takes the shortest time because there is no fusion DE algorithm and the particles are not optimised and it is easy to fall into a local optimum. Although the PSO algorithm saves time, the accuracy of its optimal solution is lacking; Figure 4 shows the normalisation of the test data and the fitting of the target curve. The closer it is to the target curve, the more accurate the prediction result. The PSO-EIDE algorithm is tested repeatedly, and the normalised correlation coefficients are all close to the diagonal, further indicating the consistency of the test results on different data. Combined with the overall performance of the algorithm, the PSO-EIDE algorithm is selected as the best.

After the algorithm is improved, the optimisation time is slightly worse, but the algorithm performs best in terms of accuracy. The reason is that the algorithm adopts the strategy of separating good and bad particles, which has greatly improved the optimisation ability.The introduction of the differential evolution algorithm increases the diversity of the population and more particles can be selected, which makes up for the deficiency of the particle swarm optimization algorithm. Particles have stronger adaptability and can achieve better global search ability.

The PSO-EIDE modifies the weights of the BP neural network and the correlation pairs are shown in Figure 5:

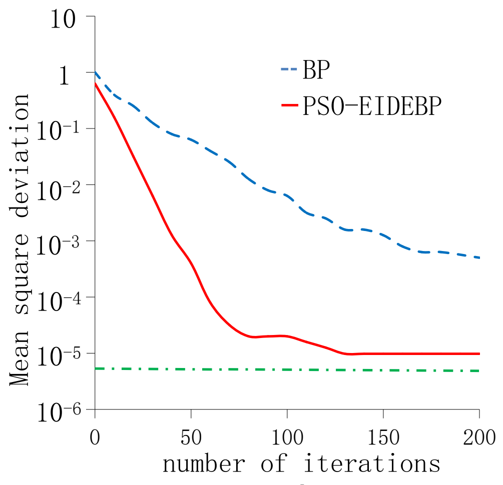
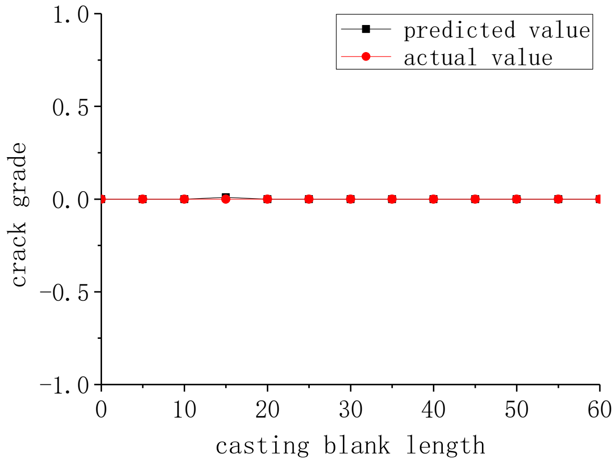


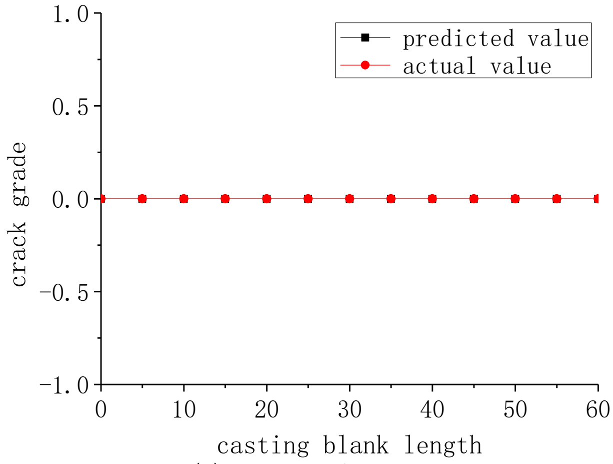
Fig. 5 Mean square deviation

From Figure 5 it can be seen that after optimising the PSO-EIDE algorithm, the iteration curve completes rapid convergence at the initial stage of the algorithm. When the number of iterations of the BP neural network reaches 75, it has initially converged. When the number of iterations reaches 130, training accuracy is achieved. Compared with the traditional BP neural network, the convergence ability of the improved algorithm is significantly improved, and the mean square error curve of the improved neural network decreases faster and the stability is stronger.

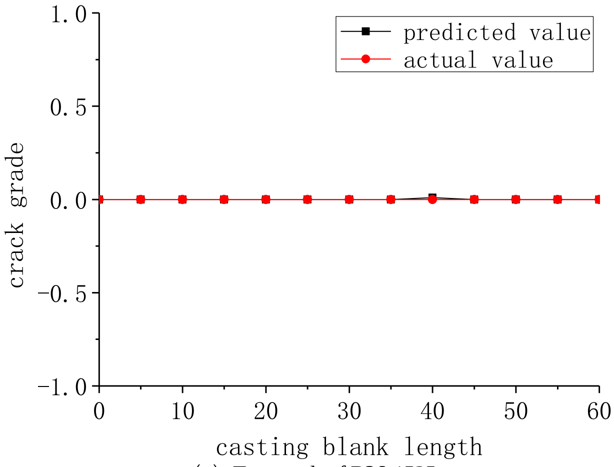
In terms of slab quality, two types of qualified and unqualified slabs were selected for full surface longitudinal crack testing. A total of 7 groups of 56 tests were performed. The data were normalised using equation (1). The predicted results are shown in Figure 6 and Figure 7.



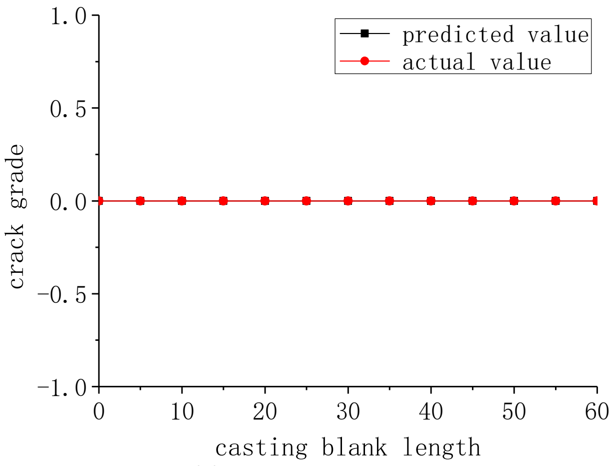
(a) Test result of B36-1510



(b) Test result of B36-1520

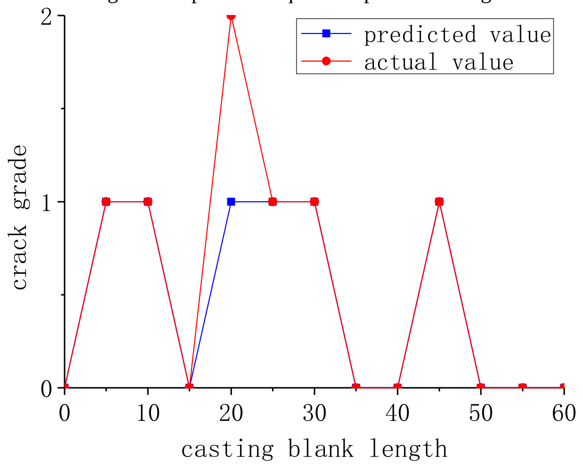


(c) Test result of B36-1525

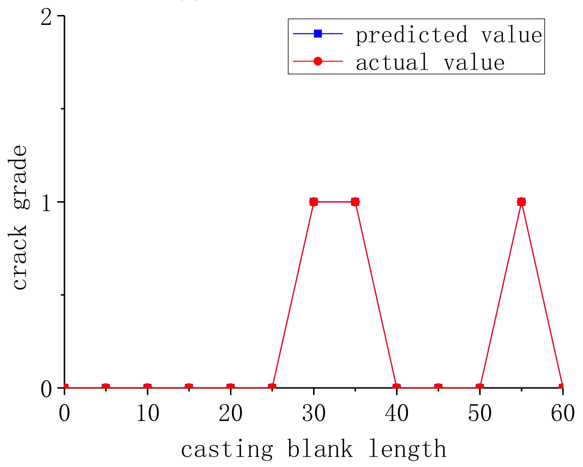


(d) Test result of B36-1530

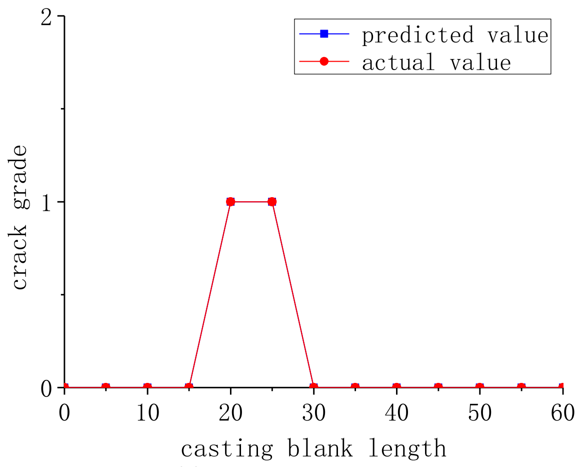
Fig. 6. Comparison of qualified product testing



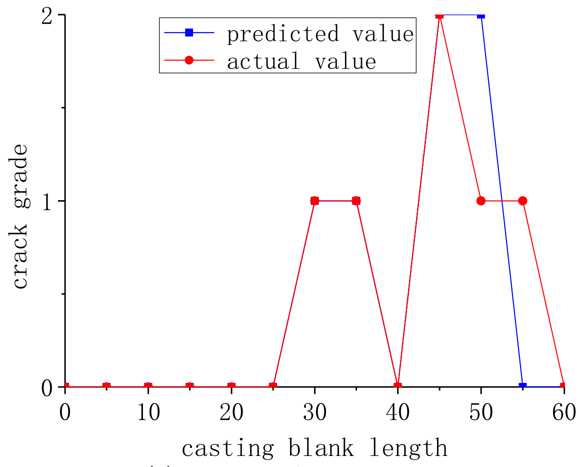
(a) Test result of B70-2965



(b) Test result of B70-2975



(c) Test result of B70-2985



(d) Test result of B70-2990

Fig. 7. Comparison of Unqualified Product Testing

After the test, some results are selected. From the test comparison of qualified and unqualified products in Figure 6 and Figure 7, the accuracy rate of qualified products is 96.15%, the accuracy rate of unqualified products is 94.23%, and the overall accuracy rate is 95.19%. For the qualified products without cracks, the prediction accuracy is high. Compared with the slab without cracks, the prediction value of the slab with cracks is slightly deficient, which basically meets the technical requirements, indicating that the prediction model can effectively predict the quality of the slab.

VI. Conclusion

Aiming at the problem of slab quality prediction under complex working conditions, a model combining hybrid optimisation algorithm and BP neural network is established in this paper. Based on the particle swarm optimization algorithm, the model proposes a strategy of dividing the superior and inferior particles and repositioning the three types of particles in the iteration, which solves the problem of the weak global search ability of particles in the later stage of the algorithm. Combined with the differential evolution algorithm, the local crossover strategy is used to increase the diversity of particles, thereby improving the overall activity of the population and speeding up convergence. From the analysis of the experimental results, the optimised algorithm has undergone a lot of tests in terms of accuracy, convergence, stability and target fitting, and the effect has been significantly improved.

References

[1] I. A. Varfolomeev, E. V. Ershov, and L. N. Vinogradova, “Statistical Control of Defects in a Continuously Cast Billet Based on Machine Learning and Data Analysis Methods,” Automation and Remote Control, vol. 79, no. 8, pp1450-1457, 2018

[2] K. N. Anisimov, I. A. Krasnyanskaya, O. V. Bakulin, and V. S. Kulikov, “Thermophysical Characteristics of Mold Flux Bulk Layer for Continuous Casting of Steel,” Steel in Translation, vol. 52,no.12, pp1149-1153, 2022

[3] Aditya Narayan Shiv Shankar Swain, Suvankar Ganguly, Arunava Sengupta, Elanjickal Zachariah Chacko, Swapnil Dhakate, and Pankaj Kumar Pandey, “Investigation of Corner Cracks in Continuous Casting Billet Using Thermomechanical Model and Plant Measurements,” Metals and Materials International, vol. 28, no.10, pp2434-2447, 2022

[4] Jiong-Ming Zhang, Qing-Hai Zhou, Yan-Bi Yin, Xing-Xing Wu, and Hua-Yang Liu, “Research and application of three-dimensional dynamic secondary cooling and accurate soft reduction for continuous casting slab,” Gongcheng Kexue Xuebao/Chinese Journal of Engineering, vol. 43, no.12, pp1666-1678, 2021

[5] Bendjama Hocine, Bouhouche Salah, Aouabdi Salim, and Bast Jürgen, “Monitoring of casting quality using principal component analysis and self-organizing map,” International Journal of Advanced Manufacturing Technology, vol. 120, no.5-6, pp3599-3607, 2022

[6] XinShun Yang, JiaJia Zhou, and DaoQun Wen, “An optimized BP neural network model for teaching management evaluation,” Journal of Intelligent and Fuzzy Systems, vol. 40, no.2, pp3215-3221, 2021

[7] Shuhui Cheng, Youxi Wu, Yan Li, Fang Yao, and Fan Min, “TWD-SFNN: Three-way decisions with a single hidden layer feedforward neural network,” Information Sciences, vol.579, pp15-32, 2021

[8] Qiang Fu, Dongdong Zhao, Shaolei Zhang, and Yuan Song, “Magnetic Localization Algorithm of Capsule Robot Based on BP Neural Network,” IEEE Transactions on Instrumentation and Measurement, vol. 73, pp1-9, 2024

[9] D. V. Yavna, V. V. Babenko, O. A. Gorbenkova, I. V. Plavelsky, V. D. Voronaya, and A. S. Stoletniy, “Classification of objects and scenes by a neural network with pretrained input modules to decode spatial texture inhomogeneities,” Journal of Optical Technology, vol. 90, no.1, pp20-25, 2023

[10] N. Yuvaraj, R. Arshath Raja, and N. V. Kousik, “Privacy Preservation Between Privacy and Utility Using ECC-based PSO Algorithm,” Advances in Intelligent Systems and Computing, Ghaziabad, India, 2021, pp567-573.

[11] M. P. Anbarasi, and S. Kanthalakshmi, “Power maximization in standalone photovoltaic system: an adaptive PSO approach,” Soft Computing, vol. 27, no.12, pp8223-8232, 2023

[12] Shruti Gupta, Rajani Kumari, and Rishi Pal Singh, “Lunar cycle inspired PSO for single machine total weighted tardiness scheduling problem,” Evolutionary Intelligence, vol. 14, no.3, pp1355-1366, 2021

[13] Fahad Wallam, “A dynamical convolutional neural network–based adaptive observer for information-poor systems,” Transactions of the Institute of Measurement and Control, vol. 45, no.16, pp3159-3172, 2023

[14] Subhajit Chaudhury, and Toshihiko Yamasaki, “Robustness of Adaptive Neural Network Optimization under Training Noise,” IEEE Access, vol.9, pp37039-37053, 2021

[15] Haifeng Li, Cong Xu, Lin Ma, Hongjian Bo, and David Zhang, “MODENN: A Shallow Broad Neural Network Model Based on Multi-Order Descartes Expansion,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no.12, pp9417-9433, 2022

[16] Lanfen Liu, Xinfeng Yang, and Ke Yang, “Research on Multiple Products Aggregate Production Planning under Random Environment,” IAENG International Journal of Applied Mathematics, vol. 54, no.3, pp562-575, 2024

[17] W. S. Kiran, S. Smys, and V. Bindhu, “Clustering of WSN Based on PSO with Fault Tolerance and Efficient Multidirectional Routing,” Wireless Personal Communications, vol. 121, no.1, pp31-47, 2021

[18] Ch. Amarendra, and K. Harinadha Reddy, “Performance analysis intelligent-based advanced PSO algorithm and testing of real-time matrix converter electrical system,” Soft Computing, vol. 24, no.18, pp 14209-14220, 2020

[19] Yu Peng, Jin-fang Cheng, and Run-xiang Jiang, “Inversion of UEP signatures induced by ships based on PSO method,” Defence Technology, vol. 16, pp172-177, 2020

[20] Ashok Singh Bhandari, Akshay Kumar, and Mangey Ram, “Grey wolf optimizer and hybrid PSO-GWO for reliability optimization and redundancy allocation problem,” Quality and Reliability Engineering International, vol. 39, nol.3, pp905-921, 2023

[21] Shokoufeh Naderi, Maude J. Blondin, and Behrooz Rezaie, “Optimizing an adaptive fuzzy logic controller of a 3-DOF helicopter with a modified PSO algorithm,” International Journal of Dynamics and Control, vol. 11, no.4, pp1895-1913, 2023

[22] Mohammad Zare, and Manfred Koch, “Hybrid signal processing/machine learning and PSO optimization model for conjunctive management of surface–groundwater resources,” Neural Computing and Applications, vol. 33, no.13, pp8067-8088, 2021

[23] Mohammad Zubair Khan, R. Mangayarkarasi, C. Vanmathi, and M. Angulakshmi, “Bio-Inspired PSO for Improving Neural Based Diabetes Prediction System,” Journal of ICT Standardization, vol. 10, nol.2, pp179-200, 2022

[24] Derya Sekman, and Vatan Karakaya, “On Chaos Controlling Mechanism for Ishikawa Iteration and Its Traffic Flow Model in Discrete Dynamical Systems,” Journal of Dynamical and Control Systems, vol. 29, no.4, pp1547-1570, 2023

[25] Alejandro Silva-Juárez, Carlos Javier Morales-Pérez, Luis Gerardo de la Fraga, Esteban Tlelo-Cuautle, and José de Jesús Rangel-Magdaleno,

"On maximizing the positive Lyapunov exponent of chaotic oscillators applying DE and PSO," International Journal of Dynamics and Control, vol. 7, no.4, pp1157-1172, 2019

[26] Lingyu Wu, Zixu Li, Wanzhen Ge, and Xinchao Zhao, "An adaptive differential evolution algorithm with elite gaussian mutation and bare-bones strategy," Mathematical Biosciences and Engineering, vol. 19, nol.8, pp8537-8553, 2022

[27] Lifeng Chen, "Parameter Tuning of a PID Controller Based on the Cellular Genetic Algorithm," Engineering Letters, vol.32, no.4, pp828-834, 2024

[28] Ismail M. Ali, Daryl Essam, and Kathryn Kasmarik, "Novel binary differential evolution algorithm for knapsack problems," Information Sciences, vol. 542, pp177-194, 2021

[29] Divya Asija, and Pallavi Choudekar, "Congestion management using multi-objective hybrid DE-PSO optimization with solar-ess based distributed generation in deregulated power Market," Renewable Energy Focus, vol. 36, pp32-42, 2021

[30] Jing Wu, Yan-Ming Cheng, Cheng Liu, Il-Kyoo Lee, Jae-Sang Cha, and Wen-Lin Huang, "A BP Neural Network Based on Improved PSO for Increasing Current Efficiency of Copper Electrowinning," Journal of Electrical Engineering and Technology, vol.16, nol.3, pp1297-1304, 2021