Mathematical Biosciences and Engineering

MBE, 19(9): 8866–8891.
DOI: 10.3934/mbe.2022411
Received: 19 February 2022
Revised: 30 May 2022
Accepted: 01 June 2022
Published: 17 June 2022

http://www.aimspress.com/journal/MBE

Research article

A multi-strategy firefly algorithm based on rough data reasoning for power economic dispatch

Ning Zhou*, Chen Zhang and Songlin Zhang

School of Electronics and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China

* **Correspondence:** Email: zhouning@lzjtu.edu.cn.

Abstract: Dynamic economic dispatch (DED) is a multi constraint and nonlinear complex problem, which is embodied in the dynamic decision-making coupled with each other in time and space. It is generally transformed into a high-dimensional multi constraint optimization problem. In this paper, a multi Strategy firefly algorithm (MSRFA) is proposed to solve the DED problem. MSRFA puts forward three strategies through the idea of opposite learning strategy and rough data reasoning to optimize the initialization and iteration process of the algorithm, improve the convergence speed of the algorithm in medium and high dimensions, and improve the escape ability when the algorithm falls into local optimization; The performance of MSRFA is tested in the simulation experiment of DED problem. The experimental results show that MSRFA can search the optimal power generation cost and minimum load error in the experiment, which reflects MSRFA superior stability and ability to jump out of local optimization. Therefore, MSRFA is an efficient way to solve the DED problem.

Keywords: power system dynamic economic dispatch; firefly algorithm; opposing learning strategies; rough data reasoning; relational reasoning

1. Introduction

With the rapid and high quality development of the country's social economy, the demand for electricity in most industries across the country is also increasing continuously. At the same time the crisis in the world in terms of non-renewable energy is getting worse day by day, and the rational use of resources is receiving more and more attention from experts and scholars. Therefore, the rational utilization of electric power resources has become an important research topic. Dynamic Economic

Dispatch (DED) plays an important role in the operation and control of the power system, which is a dynamic decision problem coupled in time and space, and its main objective is to reduce the total fuel cost as much as possible while meeting the load demand and operating conditions.

Economic dispatch of power systems solves a multi-constrained, nonlinear and complex problem. At present, the main research methods are divided into two kinds, one is linear programming technique based on Lagrangian optimization function and neural network [1], but linear programming requires derivatives of the problem as well as gradients, and the degree of calculation is too complicated when dealing with high-dimensional power systems. The other is the iterative population intelligence based optimization algorithm, which is also a commonly used method, some classical algorithms such as differential evolution algorithm (DE) [2], genetic algorithm (GA) [3,4], particle swarm algorithm (PSO) [5,6], etc. In addition to the classical intelligent algorithms, numerous emerging algorithms are also widely used in DED problems. Literature [7] has proposed an enhanced exploratory whale optimization algorithm (EEWOA) that improves population diversity using the proposed adaptive enhanced exploratory mechanism, increases the ability of the algorithm to jump out of the local optimum, and solves the DED problem using the algorithm. In the literature [8], a chaotic bat algorithm is proposed to solve the dynamic scheduling problem. The authors demonstrate the effectiveness of the proposed algorithm by applying it to the 150 kilovolt power system on East Kalimantanand by comparing it with the bat algorithm and the particle swarm algorithm. In the literature [9], a multiswarm statistical particle swarm optimization is proposed for solving the economic management of power planning and resources, and it is tested on a solar photovoltaic power system, and the experiments prove that the algorithm can effectively solve the renewable energy dynamic economic dispatch problem. In the literature [10], a dragonfly algorithm (DA), is used to solve the DED problem, which is derived from the static and dynamic swarming behavior of dragonflies. The effectiveness of the proposed algorithm in solving the DED problem is demonstrated through the simulation experiments of the algorithm in the IEEE 5 unit test system. These algorithms are cited to solve the DED problem and obtain good results.

As a kind of swarm intelligence optimization algorithm, the firefly optimization algorithm inherits the advantages of its relatively simple structure and easy understanding, and possesses the features of few parameters, easy adjustment and suitable for parallel computation. Through the study of literature [11], we found that the firefly algorithm has good convergence speed and solution accuracy when facing continuous, single-peaked and multi-peaked objective functions, especially when facing nonlinear and multi-constrained objective functions, the firefly algorithm also has good solution performance, and compared with other algorithms, the firefly algorithm has strong adaptability of objective functions; meanwhile, the firefly algorithm has been widely used in scheduling problems in various industrial fields in recent years [12–15], such as complex optimization scheduling [16], intelligent production scheduling [17], and cloud computing task scheduling [18], which proves that the Firefly algorithm has unique advantages in dealing with scheduling problems. The dynamic power scheduling problem is one of the typical scheduling problems, and in general the DED problem is modeled with a nonlinear objective function and multiple complex constraints. The Firefly algorithm is adapted to solve this kind of problem, so the Firefly algorithm is chosen as the base algorithm to solve the DED problem in this paper.

However, the DED problem includes high-dimensional objectives and functions as well as nonlinear constraints, and the computation is very complex, so the FA algorithm is prone to problems such as falling into local optimum, slow convergence and low solution accuracy. This will largely affect the solution quality of the actual DED problem. Inspired by the references [19–22] and the rough data inference literature [23], this paper proposes a firefly algorithm (MSRFA) based on rough data inference and with three iterative strategies for solving the DED problem.

The MSRFA algorithm proposed in this paper introduces the opposing learning strategy, the greedy algorithm idea and the rough inference theory, which are quite different from other metaheuristic algorithms. 1) The MSRFA algorithm uses a specific contrastive learning approach for population initialization, which is different from the random initialization of general metaheuristic algorithms. The initialization approach of contrastive learning strategy can help the algorithm to obtain good population distribution, diversity and population quality at initialization. Such an approach can help the algorithm get rid of the influence of initial values on the algorithm solution performance and improve the algorithm solution accuracy to a certain extent; 2) the introduction of the attractiveness selection mechanism based on the greedy algorithm and the reasoning of association relations in the theory of rough inference directly changes the iterative approach of the firefly algorithm. Unlike the probabilistic selection mechanism of genetic algorithm (GA) and particle swarm algorithm (PSO) that directly leans toward the better individual, the MSRFA method first uses the attractiveness selection mechanism when iterating, and the individual calculates the individual with the highest attractiveness in the population to move, and at the same time uses the associative relationship reasoning when moving to search around the moving target individual to find other possible better moving points. Such an approach can greatly accelerate the convergence speed of the algorithm at the early stage of its operation on the one hand, and also help the algorithm to improve the solution accuracy. 3) The general metaheuristic algorithm converges faster, but there is no corresponding strategy to help the algorithm jump out of the local optimum, and it can only rely on the unique mechanism of the algorithm itself, so most metaheuristic algorithms are easy to fall into the local optimum when solving complex problems, and cannot obtain better solutions. in the MSRFA algorithm, we introduce the equivalence relation inference strategy, and when the algorithm falls into the local optimum, it will automatically enable the strategy is automatically enabled when the algorithm falls into a local optimum, and the better individuals in the current population are divided into equivalence classes, and the individuals are adjusted within a certain range to help the algorithm jump out of the local optimum. This approach also ensures that the algorithm can obtain a good solution accuracy. Therefore, MSRFA is well suited to solve the DED problem.

2. Dynamic power economic dispatch

The dynamic power economic dispatch problem is one of the fundamental problems in power system operation. DED [24–26] aims to dispatch the output of online generating units with the expected load demand in a certain time period so that the power system operates most economically within the safety limits. That is, the constraints are satisfied on the basis of meeting the hourly load demand while reducing the total generation cost of the unit so that the generation cost is minimized.

The objective function of the DED is the total production cost of NG generating units at intervals of T. The production cost can be approximated as a quadratic function from the active power output of the generating units and is described as

$$F_c = \sum_{k=1}^{T} \sum_{i=1}^{N_G} F_{ih}(P_{ih}) \tag{1}$$

$$F_{it}(P_{it}) = a_i P_{it}^2 + b_i P_{it} + c_i + \left| e_i \sin(f_i (P_{it}^{\min} - P_{it})) \right| \tag{2}$$

where, i = 1,2,3,…,NG. a_i, b_i and c_i is the cost factor, e_i and f_i is the consumption factor, P_{it} is the actual power output (in MW) of the i-th generator at time t, NG is the number of online generating units to be dispatched, and T is the total dispatch time.

The load and loss constraint of the system is Eq (3), where, P_D denotes the total system load and P_L denotes the loss.

$$\sum_{i=1}^{N_G} P_{it} = P_{Dt} + P_{Lt} \tag{3}$$

where B is obtained using the P_{Lt} factor with the following equation.

$$P_{Lt} = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} P_{it} B_{ij} P_{jt} \tag{4}$$

The output power of each generator set is at the upper and lower limits directly. The generator constraint is represented by a pair of inequality constraints.

$$P_i^{\min} \le P_{it} \le P_i^{\max} \tag{5}$$

where P_i^{min} and P_i^{max} are the lower and upper output limits of the power of the i-th station generator unit. Also the rise and fall rates of each unit satisfy the following constraints.

If the generation capacity increases,

$$P_i - P_i^{t-1} \le UR_i \tag{6}$$

If the generation capacity decreases,

$$P_i^{t-1} - P_i \le DR_i \tag{7}$$

where, P_i^{t-1} is the power generation of generator unit I in the previous hour, UR_i denotes the upper slope, and DR_i denotes the lower slope. The experiments are simulated using the following fitness function model.

$$f_k = \sum_{t=1}^{n} \sum_{i=1}^{N} F_i(P_{it}) + \lambda_1 \left(\sum_{t=1}^{n} \sum_{i=1}^{N} P_{it} - P_{Dt} \right)^2 + \lambda_2 \left(\sum_{t=1}^{n} \sum_{i=1}^{N} P_{it} - P_{r\lim} \right)^2 \tag{8}$$

where λ_1 and λ_2 are penalty factors, n is the number of hours, and N is the number of units. The penalty factor regulates the objective function such that the algorithm provides a higher cost value.

$P_{r\,lim}$ is determined by Eq (9).

$$P_{r\,\mathrm{lim}} = \begin{cases} P_{i(t-1)} - DR_i, P_{it} < P_{i(t-1)} - DR_i \\ P_{i(t-1)} + DR_i, P_{it} > P_{i(t-1)} + DR_i \\ P_{it}, otherwise \end{cases} \tag{9}$$

3. Overview of relevant theories and optimization algorithms

3.1. Opposite learning strategies

For population intelligence algorithms, the quality of the initial population affects the search capability of the algorithm. Generally, the population is initialized randomly, but it does not guarantee diversity and does not effectively expand the spatial distribution of solutions, so a contrastive learning strategy is used to expand the distribution of the initial population as much as possible. First, set the population size N. Randomly initialize the location $p=(x_1,x_2,...x_n,)$ of each individual of the population to be a point in the n-dimensional space, where $x_i \in [L_i,U_i];i=1,2,...,n;$ L and U denote the upper and lower bounds, respectively. The opposing point is.

$$p^t = (x_1^t, x_2^t, ..., x_d^t), x_i^t = L_i + U_i - x_i \tag{10}$$

In this way, we can obtain a total of 2N individuals, merge the 2N individuals, and select the best adapted N individuals from them as the initial population.

3.2. Rough data inference

Rough data inference is data-based inference, and data links that are unclear, non-deterministic, seemingly present or latent between data are called rough data links. The purpose of introducing rough data inference is the desire to describe the actual rough data links. Before introducing rough data inference we introduce the concept of rough inference space.

The rough inference space originates from the approximation space $M = (U,R)$ and is an extension of the approximation space. To define the rough inference space, let U be the data set, called the argument domain, whose elements are called the data; let $K=\{R_1,R_2,...,R_n\}(n \geq 1),$ where $R_1, R_2,..., R_n$ are n distinct equivalence relations on U. The equivalence relation $S = U \times U$ given on U is called the S-reasoning relation. The structure consisting of U, K and S is denoted as W and $W = (U, K, S)$ is called the rough inference space [23].The rough inference space is introduced here to refine the rough data inference more and to make the theory of rough data inference operating on the space more concrete and clear.

The concepts of antecedents and successors are then introduced on the basis of the rough reasoning space.

Let $W = (U, K, S)$ be a rough inference space, and for $a,b \in U$, if $< a,b > \in S$, then $< a,b >$ is said to be a S-directed edge, a is called the S-predecessor of b, and b is called the successor of a; the sequence $< a,b_1 >,< b_1,b_2 >,..., < b_{n-1},b_n >,< b_n,b > (n \geq 0)$ of S-directed edges is called the path from a to b.

On this basis we can obtain the S-successor set $[a-R]$ of $[a]_R$ in the rough inference space, which leads to the definition of rough data inference.

Definition 2.1 [23]

$W=(U,K,S)$ is the space of rough reasoning, $a \in U$ and $R \in K$.

1) $b \in U$, if $b \in R^*[a-R]$, then we say that a is a direct rough introduction to b, with respect to R, and we denote it as $a \Rightarrow_R b$;

2) $b_1, b_2, \ldots, b_n, b \in U$, $a \Rightarrow_R b_1, b_1 \Rightarrow_R b_2, \ldots, b_{n-1} \Rightarrow_R b_n, b_n \Rightarrow_R b (n \geq 0)$ says that a is roughly introduced b with respect to R and is denoted as $a \vDash_R b$.

The inference for $R \in K$, a about R that is directly roughly introduced or roughly introduced to b is called rough data inference for R about in $W=(U,K,S)$, or rough data inference for short [23].

3.3. Standard firefly algorithm

Firefly algorithm (FA) is a mathematical model constructed by imitating the behavior of information exchange between fireflies and attracting each other's collections. It mainly uses the luminescence property of fireflies to simulate the behavior of mutual attraction between fireflies and search for the optimal solution in the search space. The adaptation value of the objective function is used to represent the brightness of the fireflies, and each firefly moves its position due to the attraction of a brighter firefly in the neighboring fireflies. If there is no neighboring firefly that is brighter than this firefly, then it moves randomly. The mutual attraction and position update between fireflies can be modeled as an optimization process. The attraction between fireflies increases as the distance between them decreases.

In the D-dimensional solution space, the position of each firefly is.

$$X=(x_1,x_2,\ldots,x_n) \tag{11}$$

The relative attraction between the fireflies is given by Eq (12).

$$\beta(\gamma)=\beta_0 e^{-\gamma r^2} \tag{12}$$

In Eq (12), β_0 is its initial attraction, that is, the attraction when the distance between two fireflies is 0, γ is the absorption rate of light by the medium, and r is the distance between two fireflies.

The distance traveled is calculated by the following Eq (13).

$$X_i^{'}=X_i+\beta_0 e^{-\gamma r^2}(X_i-X_j)+\alpha rand() \tag{13}$$

where X_i denotes the location of a firefly that is brighter than the i-th individual, and r denotes the distance between the i-th firefly and the j-th firefly. $rand()$ is a random perturbation and is the step factor of the perturbation. Since all individuals will only fly toward individuals with higher brightness than themselves, then the brightest individual in the population will not update its position. The brightest individual in the group will update its position according to the following Eq (14).

$$X_i^{'}=X_i+\alpha randGuass() \tag{14}$$

when the algorithm reaches the maximum number of iterations, the optimal location of the searched fireflies is output as the solution, otherwise it continues to update the locations of all fireflies.

Table 1. FA pseudo-codes.

Algorithm 1 Firefly Algorithm

```
1: Objective function $f(\boldsymbol{x}), \boldsymbol{x} = (x_1, ..., x_d)$
2: Generate initial population of fireflies $x_i (i = 1, 2, ..., n)$
3: Light intensity $I_i$ at $x_i$ determined by $f(x_i)$
4: Define light absorption coefficient $\gamma$
5: while (t < Max-Generation) do
6:    for $i = 1:n$ all $n$ fireflies do
7:       for $j = 1:i$ all $n$ fireflies do
8:          if ($I_j > I_i$), Move firefly $i$ towards $j$ in d-dimension; Then
9:          end if
10:         Attractiveness varies with distance $r$ via $\exp[-\gamma r]$
11:         Evaluate new solutions and update light intensity
12:      end for $j$
13:   end for $i$
14:   Rank the fireflies and find the current best
15: end while
16: Postprocess results and visualization
```

4. Multi-strategy firefly algorithm

4.1. Initialization

The standard FA initializes the population information in a random way, and the original initialization method produces a large population randomness and unstable population diversity due to the initial value-sensitive characteristics of the heuristic algorithm, which has a large impact on the accuracy and speed of the algorithm solution; in this paper, we use the contrastive learning strategy to initialize the population; the contrastive learning strategy constructs the initial population by calculating the contrastive points of the initial point In this paper, we use the opposing learning strategy to initialize the population; the opposing learning strategy calculates the opposing points of the initial points, constructs *2N* individuals, and selects the *N* individuals with the best fitness as the initial population, which expands the distribution of the initial population as much as possible and effectively expands the spatial distribution of the understanding, and also ensures the diversity of the population.

4.2. Update strategy

In the standard FA, two firefly individuals are compared directly to determine the direction of movement and distance of the individuals, in such a way that most of the individuals will move in each iteration of the algorithm, resulting in a slow convergence of the algorithm in the beginning stage, and there is no adjustment of the rules after falling into a local optimum, leading to an algorithm that is easily fallen into a local optimum, and the algorithm has a low solution accuracy and poor stability.

In this paper, three strategies are proposed to improve the iteration rules of the FA algorithm.

Strategy 1: attractiveness selection strategy

In the initial stage of the algorithm operation, the formula for attractiveness selection is introduced in conjunction with the greedy algorithm as follows.

$$P_{ij} = \frac{x_{B(t)}}{x_{B(t)} + R_{ij}} \tag{15}$$

where $x_{B(t)}$ denotes the relative brightness of the two bodies at the current time t, R_{ij} denotes the distance between the two bodies, P_{ij} denotes the attractiveness of the two bodies, and each individual chooses the individual with higher P_{ij} to move, in contrast to the standard FA algorithm, where most individuals must move among themselves, such a strategy can speed up the convergence of the algorithm in the initial situation.

Strategy 2: Reasoning strategy under approximation rules

First construct the rough inference space $W = (U, K, S)$ as follows.

U : The thesis domain U is the set of all individuals in the firefly population.

K : Let $K = \{R_1, R_2\}$, where both R_1 and R_2 are equivalence relations, and they are obtained by the division of U .

S: The inference relation S is determined by the mutual attraction relation between individuals, i.e., $S = \{ <u,v> | u,v \in U, u \to v\}$. Due to the frequent movement between firefly individuals in the iteration rule of the FA algorithm, the attraction relation gives the individuals a connection, which provides the information for the inference relation to be formed. Suppose the inference relation is.

$$S = \{<w_1, w_4>, <w_2, w_5>, <w_3, w_5>, <w_5, w_6>\} \tag{16}$$

where, $w_1 - w_7$ is the current individual and the surrounding points and candidate points when the current individual moves. The movement of the current individual to a brighter individual will be determined by the following inference rules.

For the equivalence relation $R \in K$, assume that the division of U with respect to R is.

$$U / R = \{\{w_1, w_2, w_3\}, \{w_4, w_5\}, \{w_6, w_7\}\} \tag{17}$$

The division here is based on the distance between individuals, and the rough inference schematic in the algorithm can be obtained by combining the above information, as shown in Figure 1.

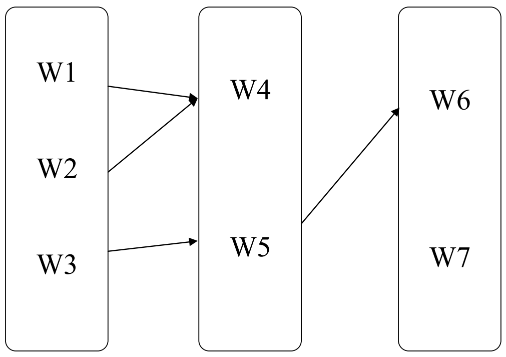


Figure 1. Schematic diagram of rough data reasoning in individual movement.

In Figure 1, in the process of rough inference, for individual w_1, w_2, w_3 is obtained by the approximation rule (dividing individuals in a certain range into a group), then dividing w_1, w_2, w_3 into a data set, and dividing $w_4 \sim w_7$ by the same reasoning; then w_4 is obtained by the inference rule in w_1, and w_5, w_6, w_7 by the same reasoning. By the definition of rough data inference, for w_1 there is $[w_1]_R = \{w_1,w_2,w_3\}$, $R^*[w_1 - R] = \{w_4,w_5\}$ then $w_1 \Rightarrow_R w_5$. For individual w_5 we have $[w_5]_R = \{w_4,w_5\}$, $R^*[w_5 - R] = \{w_6,w_7\}$. Then $w_5 \Rightarrow_R w_7$. From $w_1 \Rightarrow_R w_5$, $w_5 \Rightarrow_R w_7$, we can obtain $w_1 \Rightarrow_R w_7$. From the above, it can be concluded that there is also a potential correlation between w_1 and w_7, which can provide individuals with new options when moving the selection point.

Strategy 3: Equivalence Reasoning Strategy

Using equivalent inference strategies to help the algorithm improve escape when it reaches a local optimum.

Using equivalent inference strategies to help the algorithm improve escape when it reaches a local optimum. In rough data inference for the S path $<u,b_1>, <b_1,b_2>, ..., <b_{n-1},b_n>, <b_n,v>$ recorded in the deterministic data linkage corresponds to the trajectory of the individual movement in the algorithm, the data in the path b_i. By b_i the path can be decomposed into two sub-paths: $<u,b_1>, <b_1,b_2>, ..., <b_{i-1},b_i>$ and $<b_i,b_{i+1}>, ..., <b_{n-1},b_n>, <b_n,v>$, which u are the S path to b_i and the S path from b_i to v, respectively. $u \vDash_{\mathcal{R}} b_i$ and $b_i \vDash_{\mathcal{R}} v$, which are indicative of the existence of a definite data link from u to b_i.The R-equivalence class $[b_i]_R$ is considered when the data represented by b_i is problematic or the corresponding fitness falls into a local optimum. For $w \in [b_i]_R (w \neq b_i)$, there are $u \vDash_{\mathcal{R}} w$ and $w \vDash_{\mathcal{R}} v$, i.e., u is rough about R pushing out w, and w is rough about R pushing out v. We can obtain that the equivalence class w of b_i provides us with new path choices, which can improve the ability of the algorithm to jump out of the local optimum.

4.3. MSRFA implementation process

Based on the above update initialization method and three generation strategies, this paper proposes the implementation process of MSRFA. In this process, the population is first initialized with the opposing learning strategy; then the individuals in the firefly are moved using the three generative strategies; finally, the best solution after the movement is used as the global optimal solution. Figure 2 shows the process of MSRFA implementation. Table 2 shows the pseudo-code of MSRFA.

5. Experimental analysis of the algorithm

Section 5 is divided into two parts. In Section 5.1, the paper describes the functions and some parameter settings for the benchmark test. In Section 5.2, the proposed MSRFA is compared with PSO, PPSO [27], FA, and CFA [28] to verify its performance.

5.1. Test function introduction and parameter setting

In Section 5.1, the proposed algorithm will be tested using the CEC2013 suite, which contains 28 benchmark functions (f1–f28). Among them, f1–f5 have 5 single-peaked functions. These functions have only one global optimal solution. It is usually used to test the convergence speed of the algorithm. f6–f20 are 15 basic multimodal functions. These functions contain exponential local optimal solutions. It is used to test the global search ability of the algorithm and to determine whether it can jump out of the local trap. f21–f28 are eight complex functions. These functions have multiple random global optimal solutions. The search ability of the algorithm can be detected comprehensively. These 28 functions can test the performance of the algorithm in different aspects. More information about CEC2013 can be found in [29,30].

To reflect the fairness of the experiment, each algorithm was tested 30 times on each benchmark function. The data obtained include the "mean", "best" and "standard deviation" of the fitness error. Table 3 shows the relevant parameters for each algorithm. The dimension of the benchmark function is set to 30, the search space is from −100 to 100, and the number of initial solutions and iterations is 100 and 500, respectively. In addition, the number of function tests is the same for all algorithms.

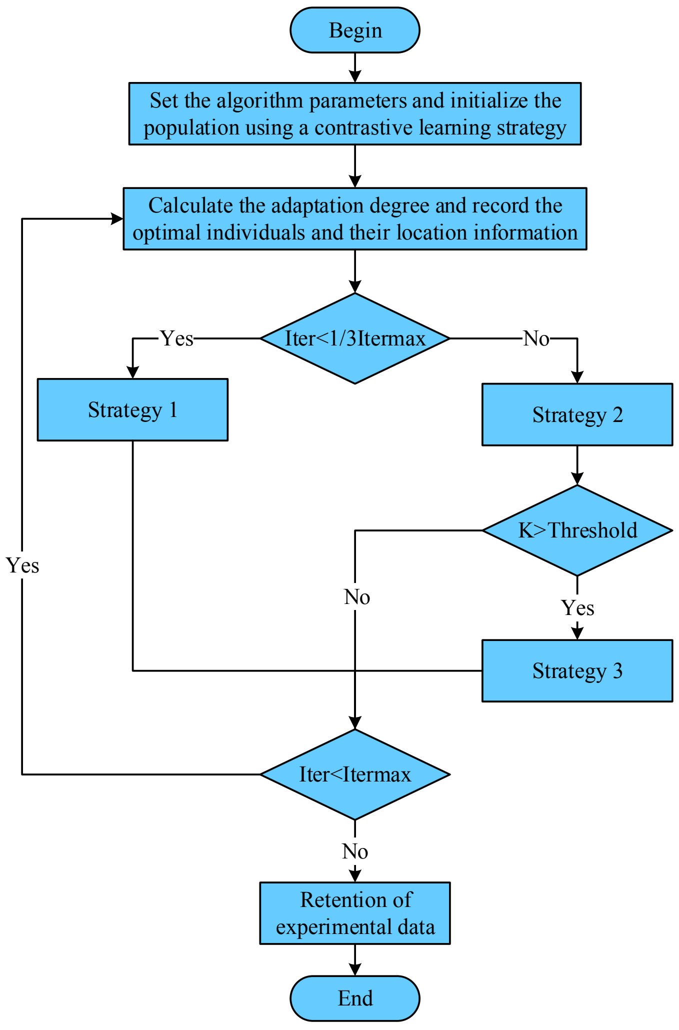

Begin
Set the algorithm parameters and initialize the population using a contrastive learning strategy
Calculate the adaptation degree and record the optimal individuals and their location information
Iter<1/3Itermax
Yes
No
Strategy 1
Strategy 2
K>Threshold
No
Yes
Strategy 3
Yes
Iter<Itermax
No
Retention of experimental data
End

Figure 2. MSRFA algorithm flow chart.

5.2. Algorithm testing

In this section, MSRFA is compared with particle swarm algorithm (PSO) [5], A parallel particle swarm optimization algorithm (PPSO) [27], Firefly Optimization Algorithm (FA) [11] and Firefly algorithm based on Chaos Theory (CFA) [28] to verify its performance. In this paper, the performance of the algorithms is evaluated by solution accuracy. Tables 4 and 5 show the solution accuracy of each algorithm. Figure 3 shows the convergence curves of the above algorithm on the test function.

Based on the data in Tables 4 and 5, we find that the proposed MSRFA performs better in the test environment of cec2013. Compared with PSO, MRSFA achieves 27, 27 and 24 better scores in “mean”, “optimal” and “standard deviation”, respectively; under the same conditions Finally, the MSRFA achieves 25, 26 and 24 better results compared with the CFA with chaos optimization, and MSRFA achieves 25, 26 and 24 better results for complex test functions 21–28. These results demonstrate better solution accuracy.

Table 2. MSRFA pseudo-codes.

Algorithm 2 MSRFA

```
1: Objective function $f(\boldsymbol{x}), \boldsymbol{x} = (x_1, ..., x_d)$
2: Initialization population using opposing learning strategies $x_i (i = 1, 2, ..., n)$
3: Light intensity $I_i$ at $x_i$ determined by $f(x_i)$
4: Define light absorption coefficient $\gamma$
5: while (t < Max-Generation) do
6:     Calculate the fitness of each individual
7:     if $Iter < 1/3MaxIter$ then
8:       for $i = 1:n$ all $n$ fireflies do
9:         According to strategy one, Calculate the degree of attraction
10:        Select the individual whose attraction value is the top $1/3$ to move
11:      end for
12:    end if
13:    if $Iter > 1/3MaxIter$ then
14:      for $i = 1:n$ all $n$ fireflies do
15:        According to strategy two, using reasoning strategy to select moving points
16:      end for
17:    end if
18:    Rank the fireflies and find the current best
19:    if current best = Last best then $k = k + 1$
20:    end if
21:    if $k! < Threshold$ then
22:      Use strategy three to replace the individual of the optimal value
23:    end if
24: end while
25: Postprocess results and visualization
```

Table 3. Parameter settings.

Algorithm	Parameters settings
PSO	$V_{\max} = 10, V_{\min} = -10, \omega = 0.9to0.4, c_1 = c_2 = 1.49$
PPSO	$G = 4, R = 20, V_{\max} = 10, V_{\min} = -10, \omega = 0.9to0.4,$ $c_1 = c_2 = 1.49$
FA	$\beta_0 = 1.0,$ $\beta_{\min} = 0.2, \beta_{\max} = 1.0, \gamma = 1, \alpha_0 = 0.97$
CFA	$\beta_0 = 1.0,$ $\beta_{\min} = 0.2, \beta_{\max} = 1.0, \gamma = 1, \alpha_0 = 0.97$
MSRFA	$\beta_0 = 1.0,$ $\beta_{\min} = 0.2, \beta_{\max} = 1.0, \gamma = 1, \alpha_0 = 0.97$

From these data, it can be seen that the MSRFA proposed in this paper has good performance, which proves the effectiveness of the contrastive learning strategy and the update iteration strategy. Meanwhile, the above results show that the optimization algorithm has good performance in solving complex functions, and it can be obtained that the optimization algorithm will have better performance in solving complex problems with multiple constraints and non-linearity like DED. However, in the table we can also get relatively poor data of the improved algorithm, for example, on f4, f13, f15, f19, f21 and f24, the MSRFA has significantly poorer optimization impact compared to other algorithms, so in future research, the escape capability of the algorithm when it falls into local optimum will be further enhanced.

Table 4 Performance of PSO, PPSO and MSRFA in CEC2013.

	PSO			PPSO			MSRFA		
30D	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std
1	1.54×10^{2}	2.79	2.66×10^{2}	0.664	8.54×10^{-2}	0.535	1.65×10^{-2}	3.59×10^{-3}	4.56×10^{-2}
2	6.07×10^{6}	1.69×10^{6}	3.59×10^{6}	3.34×10^{6}	8.64×10^{5}	1.42×10^{6}	7.56×10^{5}	7.20×10^{2}	8.22×10^{5}
3	8.61×10^{9}	1.22×10^{9}	7.88×10^{9}	3.18×10^{9}	6.31×10^{8}	2.23×10^{9}	5.26×10^{8}	2.56×10^{7}	6.24×10^{8}
4	1.61×10^{4}	5.37×10^{3}	5.54×10^{3}	2.52×10^{4}	8.78×10^{3}	5.17×10^{3}	6.45×10^{3}	4.58×10^{3}	7.59×10^{2}
5	98.4	25.0	36.4	33.4	1.45	26.8	1.59×10^{-2}	8.55×10^{-3}	4.51×10^{-2}
6	93.7	26.7	34.5	65.4	15.4	28.7	58.1	5.43	19.2
7	1.45×10^{2}	57.5	41.8	1.07×10^{2}	50.7	26.7	44.9	30.6	16.1
8	25.8	20.6	6.53×10^{-2}	21.9	20.8	8.22×10^{-2}	19.6	15.8	4.56×10^{-2}
9	38.8	25.4	3.54	31.5	24.3	3.44	6.04	2.46	8.54
10	65.7	7.05	55.8	7.88	2.58	3.67	1.23	0.458	6.54
11	2.52×10^{2}	1.35×10^{2}	54.8	2.48×10^{2}	1.23×10^{2}	46.8	13.6	12.3	27.5
12	2.48×10^{2}	1.06×10^{2}	57.3	2.38×10^{2}	1.02×10^{2}	55.8	52.6	24.6	57.6
13	3.59×10^{2}	2.27×10^{2}	55.9	3.41×10^{2}	2.16×10^{2}	45.2	6.51E+01	40.2	32.4
14	4.34×10^{3}	2.53×10^{3}	6.07×10^{2}	4.51×10^{3}	1.86×10^{3}	6.44×10^{2}	1.05×10^{3}	6.58×10^{2}	2.37×10^{2}
15	4.53×10^{3}	3.21×10^{3}	5.08×10^{2}	4.21×10^{3}	2.09×10^{3}	6.73×10^{2}	5.24×10^{3}	2.58×10^{3}	4.26×10^{2}
16	1.56	0.438	0.427	1.23	0.407	0.389	0.765	0.897	9.92×10^{-2}
17	2.76×10^{2}	1.72×10^{2}	59.1	2.08×10^{2}	1.22×10^{2}	33.8	85.1	54.9	13.4
18	2.43×10^{2}	1.70×10^{2}	52.6	2.02×10^{2}	95.0	34.6	91.4	64.8	17.5
19	23.0	9.59	8.64	12.8	4.59	4.63	12.6	9.58	0.615
20	15.3	13.1	0.370	15.1	14.6	0.257	2.56	1.49	2.37
21	3.82×10^{2}	2.25×10^{2}	77.6	3.76×10^{2}	1.29×10^{2}	81.3	3.64×10^{2}	1.17×10^{2}	61.5
22	5.47×10^{3}	3.66×10^{3}	9.06×10^{2}	5.13×10^{3}	3.60×10^{3}	8.24×10^{2}	1.96×10^{3}	9.84×10^{2}	2.48×10^{2}
23	5.63×10^{3}	3.13×10^{3}	1.14×10^{3}	5.30×10^{3}	3.80×10^{3}	7.28×10^{2}	1.85×10^{3}	8.74×10^{2}	3.44×10^{2}
24	3.12×10^{2}	2.77×10^{2}	16.1	3.06×10^{2}	2.47×10^{2}	10.9	1.52×10^{2}	1.32×10^{2}	9.15×10^{2}
25	3.45×10^{2}	2.86×10^{2}	17.9	3.43×10^{2}	3.02×10^{2}	17.6	2.75×10^{2}	2.42×10^{2}	6.15
26	3.31×10^{2}	2.05×10^{2}	85.5	2.77×10^{2}	2.00×10^{2}	87.5	2.25×10^{2}	1.93×10^{2}	6.59
27	1.28×10^{3}	9.02×10^{2}	1.15×10^{2}	1.23×10^{3}	9.57×10^{2}	89.2	8.19×10^{2}	6.35×10^{2}	91.6
28	2.78×10^{3}	4.55×10^{2}	8.67×10^{2}	2.18×10^{3}	1.23×10^{2}	1.03×10^{3}	7.26×10^{2}	3.48×10^{2}	6.49×10^{2}

Table 5. Performance of FA, CFA and MSRFA in CEC 2013.

	FA			CFA			MSRFA		
30D	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std
1	2.49×10^{-2}	2.60×10^{-2}	1.51×10^{-2}	2.21×10^{-2}	1.52×10^{-2}	5.20×10^{-3}	1.65×10^{-2}	3.59×10^{-3}	4.56×10^{-2}
2	7.88×10^{5}	7.62×10^{5}	8.14×10^{5}	1.16×10^{6}	2.10×10^{5}	8.01×10^{5}	7.56×10^{5}	7.20×10^{2}	8.22×10^{5}
3	7.05×10^{8}	9.54×10^{8}	6.57×10^{9}	7.03×10^{9}	7.26×10^{8}	8.95×10^{9}	5.26×10^{8}	2.56×10^{7}	6.24×10^{8}
4	2.74×10^{4}	2.36×10^{4}	3.47×10^{3}	2.66×10^{4}	1.30×10^{4}	5.69×10^{4}	6.45×10^{3}	4.58×10^{3}	7.59×10^{2}
5	3.03	0.509	4.07	1.48	7.10×10^{-2}	0.423	1.59×10^{-2}	8.55×10^{-3}	4.51×10^{-2}
6	53.5	7.97	75.9	70.5	8.86	67.5	58.1	5.43	19.2
7	2.59×10^{2}	40.5	1.78×10^{2}	2.21×10^{2}	25.7	19.2	44.9	30.6	16.1
8	20.4	20.3	9.88×10^{-2}	20.5	10.3	20.5	19.6	15.8	4.56×10^{-2}
9	8.02	6.36	2.42	7.08	3.20	9.10	6.04	2.46	8.54
10	1.80	0.660	0.994	1.76	0.580	2.58	1.23	0.458	6.54
11	21.7	1.96	12.2	14.9	11.9	0.995	13.6	12.3	27.5
12	96.7	6.39	25.3	43.3	35.8	43.8	52.6	24.6	57.6
13	72.0	38.9	16.0	69.0	48.2	96.2	65.1	40.2	32.4
14	3.80×10^{3}	1.89×10^{3}	1.43×10^{3}	1.24×10^{3}	9.78×10^{2}	2.06×10^{2}	1.05×10^{3}	6.58×10^{2}	2.37×10^{2}
15	3.55×10^{3}	2.14×10^{3}	2.26×10^{2}	3.54×10^{3}	3.48×10^{3}	1.47×10^{3}	3.24×10^{3}	2.58×10^{3}	4.26×10^{2}
16	66.8	17.8	88.3	22.9	13.3	7.80	0.765	0.897	9.92×10^{-2}
17	4.25×10^{2}	3.81×10^{2}	2.83×10^{2}	2.44×10^{2}	1.53×10^{2}	59.0	85.1	54.9	13.4
18	4.12×10^{2}	3.29×10^{2}	2.97×10^{2}	3.10×10^{2}	2.03×10^{2}	57.1	91.4	64.8	17.5
19	27.2	25.2	19.2	19.6	14.8	3.49	12.6	9.58	0.615
20	4.45	3.85	0.336	2.44	1.55	2.49	2.56	1.49	2.37
21	3.77×10^{2}	3.01×10^{2}	37.8	3.97×10^{2}	1.96×10^{2}	3.96×10^{2}	3.64×10^{2}	1.17×10^{2}	61.5
22	1.88×10^{3}	1.60×10^{3}	2.67×10^{2}	1.85×10^{3}	1.30×10^{3}	1.16×10^{3}	1.96×10^{3}	9.84×10^{2}	2.48×10^{2}
23	1.63×10^{3}	1.23×10^{3}	2.09×10^{2}	1.53×10^{3}	1.08×10^{3}	1.85×10^{2}	1.85×10^{3}	8.74×10^{2}	3.44×10^{2}
24	1.04×10^{2}	1.00×10^{2}	5.36	1.56×10^{2}	1.01×10^{2}	1.01×10^{2}	1.52×10^{2}	1.32×10^{2}	9.15×10^{2}
25	5.40×10^{2}	3.54×10^{2}	1.18×10^{3}	4.49×10^{2}	2.00×10^{2}	1.59×10^{2}	2.75×10^{2}	2.42×10^{2}	6.15
26	3.12×10^{2}	2.96×10^{2}	3.44×10^{2}	2.58×10^{2}	1.47×10^{2}	3.07×10^{2}	2.25×10^{2}	1.93×10^{2}	6.59
27	6.45×10^{3}	3.21×10^{3}	0.663	5.27×10^{3}	2.32×10^{3}	3.21×10^{2}	8.19×10^{2}	6.35×10^{2}	91.6
28	1.31×10^{3}	1.01×10^{3}	1.01×10^{2}	7.39×10^{2}	1.01×10^{2}	5.26×10^{2}	7.26×10^{2}	3.48×10^{2}	6.49×10^{2}

Table 6. Running time of each algorithm.

Algorithm	Convergence iterations	Calculation time/min	Convergence time/min
FA	200	7.82	3.12
CFA	216	8.16	3.52
PSO	273	9.65	5.23
PPSO	327	7.63	5.02
MSRFA	411	7.54	6.15

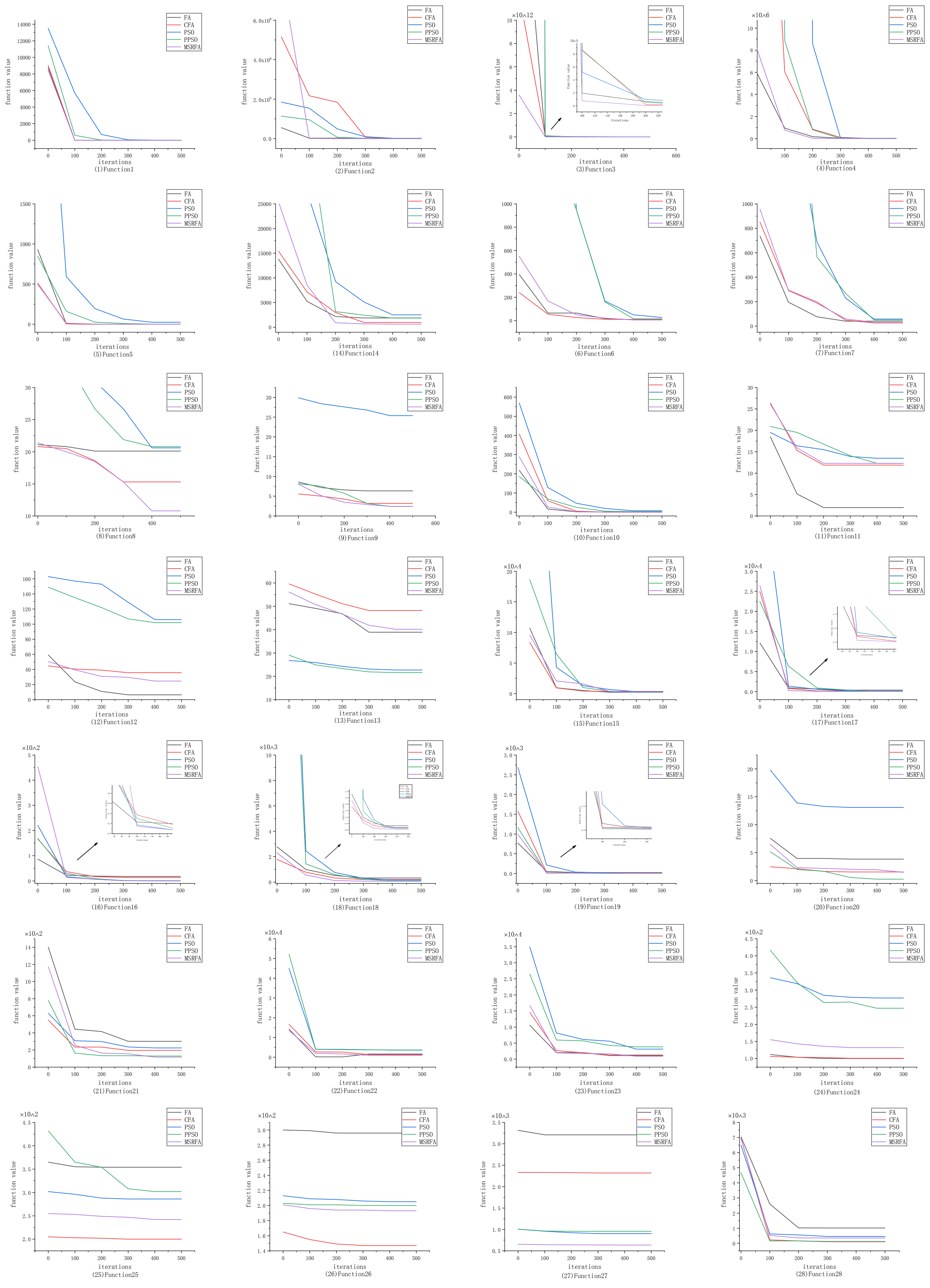


Figure 3. The convergence curves of algorithms under CEC2013.

Table 6 shows the average number of convergence iterations, average running time and average convergence time of each algorithm on 28 test functions.

From the data in the table, it can be seen that the computation time of the proposed algorithm is 7.54 min, which saves 0.09 min compared with the PPSO algorithm; the running time of MSRFA algorithm is 3, 8 and 27% shorter than the running time of FA, CFA and PSO algorithms, respectively. Although, the average number of convergence iterations and the average convergence time of the algorithm in this paper are not superior, it also shows that the algorithm proposed in this paper can continuously escape from the local optimum and approach to the global optimum through the proposed improvement strategy; the data in the table explain, to a certain extent, the reason why the MSRFA algorithm can achieve better convergence accuracy.

5.3. Time complexity analysis

Time complexity is an important indicator to evaluate the efficiency of algorithm execution. Suppose the size of the solution problem is D-dimensional, the size of the firefly population is a positive integer N, and the time required to solve the objective function is $f(D)$. In general the time complexity of the traditional firefly algorithm is:

$$T = O(N^2) \tag{18}$$

However, due to the increasing complexity of the objective function, the time required to calculate the objective function can no longer be ignored, so the time complexity of the firefly algorithm in this paper is:

$$T_F = O(N^2 f(D)) \tag{19}$$

where the time complexity of the firefly algorithm position update formula is:

$$T = O(N^2) \tag{20}$$

In MSRFA, the initialization method based on the dyadic learning strategy is a deterministic initialization method that can be directly retrieved as local information after determining the number of algorithmic individuals N by executing it only once. Assuming that the execution time required to run the contrastive learning strategy is t_1 and the time required to generate a single individual is t_2, the initialization population time complexity of MSRFA is:

$$T_1 = O(N(t_1 + t_2)) = O(N) \tag{21}$$

In the text, strategy 1 needs to be calculated using the attractiveness selection formula, calculating the attractiveness between D-dimensional individuals once takes $t_3 f(D)$, N-1 times, and the total time taken is $(N-1)t_3 f(D)$; and the individual with the greatest attractiveness needs to be selected to move toward it, and the time taken to select the individual is $t_4 N$, and the time taken to move the individual is t_5; then strategy 1 only takes effect when $Iter<1/3Iter_{\max}$ is in, so the time complexity of strategy 1 is.

$$T_2 = O(\frac{Iter_{\max}}{3}((N-1)t_3 f(D) + t_4 N + t_5)) == O(Nf(D)) \tag{22}$$

In strategy 2, the population is divided based on the location and brightness of the individual distribution, and the objective function is computed once in $f(D)$, which takes *N-1* times, and the total time consumed is *(N-1)f(D)*; at the same time, an individual needs to be selected to perform the move operation, which takes t_6 to select the individual and t_7 to move the operation. strategy 2 takes effect at $1/3Iter_{\max} < Iter < Iter_{\max}$, so the operation time consumed is:

$$T_3 = O(\frac{2Iter_{\max}}{3}((N-1)f(D)+t_6+t_7)) = O(Nf(D)) \tag{23}$$

In strategy 3, when the algorithm triggers the threshold, the algorithm is judged to fall into the local optimum, the current population of optimal individuals is selected, and the equivalence relation is reasoned, firstly, the position in the current individual is updated randomly, which takes time t_8, and the value of the individual objective function before and after the update is calculated, which takes time $2f(D)$, and is compared. Therefore, the time complexity of strategy 3 is:

$$T_4 = O(2f(D)+t_8) = O(f(D)) \tag{24}$$

Since the MSRFA algorithm proposed in this paper converges faster and easily falls into local optimum, the consumption time of strategy 3 cannot be ignored, so the time complexity of the MSRFA algorithm when the maximum number of iterations is $Iter_{\max}$ is.

$$T_M = T_1 + T_2 + T_3 + T_4 = O(Nf(D)) \tag{25}$$

In summary, $T_M \le T_F$. The time complexity of MSRFA is less than or equal to the time complexity of FA, so the introduction of the contrastive learning strategy and rough data inference theory does not increase the time complexity of the algorithm.

6. Simulation testing and data analysis

The DED model was tested and experimented on a computer with hardware configuration of Intel(R) Core(TM) i7-9750H CPU @2.6GHz using Python 3.7 programming and compared with FA algorithm, CFA algorithm [28], and Hybrid Immune Genetic Algorithm (HIGA) [3] to analyze the convergence and stability of different algorithms by analyzing the effectiveness of MSRFA for the DED problem was verified by analyzing the convergence and stability of different algorithms.

6.1. Simulation experiment setup

Test Case 1:

The first group was tested using unit 3, where the 24-hour unit 3 test system data, corresponding to the electrical load, were 171 158 153 145 145 145 151 171 198 242 245 258 265 238 245 248 258 274 256 245 234 218 198 185, in megawatts (WM). The data related to unit 3 are shown in Tables 7–9.

Table 7. Unit coefficient settings.

Unit	a_i	b_i	c_i	e_i	f_i
1	0.0053	11.6	213	100	0.0562
2	0.0088	10.3	200	140	0.0486
3	0.0074	10.8	240	160	0.0392

Table 8. Upper and lower limits of unit output power and climbing slope.

P_{min}	P_{max}	UR	DR
50	200	40	40
37.5	150	30	30
45	180	30	30

Table 9. Inter-unit loss factor.

Unit	1	2	3
1	0.0670	0.00953	0.00507
2	0.00953	0.05210	0.00901
3	0.00507	0.00901	0.02940

Test Case 2:

The system data for the second set of tests were taken from [31]. In this case, the system data for the 24-hour 5-unit test, corresponding to the electrical load demand, are: load demand is: 410 435 475 530 558 608 626 654 690 704 720 740 704 690 654 580 558 608 654 704 680 605 527 463, in megawatts (WM). Also, in this test, the algorithm of this paper is compared with the algorithm proposed in the literature [31].

In order to apply the MSRFA algorithm in the DED problem, a maximum number of 1000 iterations is used in the simulation study of the two test systems. Also, to ensure adequate analysis of the optimization results, different population sizes were applied in this experiment to compare the results. Tables 10–12 list the power dispatch related data of the system.

Table 10. Unit factor settings.

Unit	a_i	b_i	c_i	e_i	f_i
1	0.0080	2.0	25	100	0.0422
2	0.0030	1.8	60	140	0.0403
3	0.0012	2.1	100	160	0.0384
4	0.0012	2.0	120	180	0.0375
5	0.0015	1.8	40	200	0.035

Table 11. Upper and lower limits of power output and climbing slope of the unit.

P_{min}	P_{max}	UR	DR
10	75	30	30
20	125	30	30
30	175	40	40
40	250	50	50
50	300	50	50

Table 12. Inter-unit loss factor.

Unit	1	2	3	4	5
1	0.000049	0.000014	0.000015	0.000015	0.0000202
2	0.000014	0.000045	0.000016	0.000020	0.0000183
3	0.000015	0.000016	0.000039	0.000010	0.0000124
4	0.000015	0.000020	0.000010	0.000040	0.0000145
5	0.0000202	0.0000183	0.0000124	0.0000145	0.000035

Test Case 3:

The system data for the third set of tests were taken from [32]. The system contains 10 units, and in this case the corresponding 24-hour electrical load demand is: 1052, 1126, 1259, 1397, 1467, 1663, 1749, 1836, 1954, 2073, 2119, 2267, 1998, 1889, 1843, 1562, 1522, 1653, 1791, 2007, 1904, 1543, 1276, 1228 in megawatts (WM). In this test, the algorithm of this paper is compared with the DE-PSO [32]. Tables 13 and 14 lists the data related to the power dispatch of this system.

Table 13. Unit parameters.

	a_i	b_i	c_i	e_i	f_i	P_{min}	P_{max}	UR	DR
1	4.3×10^{-4}	21.60	958.2	400	0.041	150	470	80	80
2	6.3×10^{-4}	21.05	1313.6	600	0.036	135	460	80	80
3	3.9×10^{-4}	20.81	604.9	320	0.028	73	340	80	80
4	7×10^{-4}	23.90	471.6	260	0.052	60	300	50	50
5	7.9×10^{-4}	21.62	480.29	280	0.063	73	243	50	50
6	5.6×10^{-4}	17.87	601.75	310	0.048	57	160	50	50
7	2.11×10^{-3}	16.51	502.7	300	0.086	20	130	30	30
8	4.8×10^{-3}	23.23	639.4	340	0.082	47	120	30	30
9	0.109	19.58	455.6	270	0.098	20	80	30	30
10	9.5×10^{-3}	22.54	692.4	380	0.094	55	55	30	30

Table 14. Inter-unit loss factor ($B \times 10^{-5}$).

	1	2	3	4	5	6	7	8	9	10
1	8.7	0.43	−4.61	0.36	0.32	−0.66	0.96	−1.6	0.8	−0.1
2	0.43	8.3	-0.97	0.22	0.75	−0.28	5.04	1.7	0.54	7.2
3	−4.61	−0.97	9	−2	0.63	3	1.7	−4.3	3.1	−2
4	0.36	0.22	−2	5.3	0.47	2.62	−1.96	2.1	0.67	1.8
5	0.32	0.75	0.63	0.47	8.6	−0.8	0.37	0.72	-0.9	0.69
6	−0.66	−0.28	3	2.62	-0.8	11.8	−4.9	0.3	3	−3
7	0.96	5.04	1.7	−1.96	0.37	−4.9	8.24	−0.9	5.9	−0.6
8	−1.6	1.7	−4.3	2.1	0.72	0.3	−0.9	1.2	−0.96	0.56
9	0.8	0.54	3.1	0.67	−0.9	3	5.9	−0.96	0.93	−0.3
10	−0.1	7.2	−2	1.8	0.69	−3	−0.6	0.56	−0.3	0.99

6.2. DED experimental results and analysis

Tables 15 and 16 give the experimental results of MSRFA and other comparative algorithms in the DED simulation, Unit 3 and 5 tests. Here, the function dimension of each algorithm is uniformly set to 30 and the number of iterations is 1000 for 30 independent runs, and the experimental simulations are performed at population sizes of 30, 50, 100 and 300.

Table 15. Experimental results of four algorithms in 3 sets with different dimensions.

Size	Algorithm	Best	Worst	Average	Variance
30	FA	76253.08	76366.89	76304.75	2073.05
	CFA	76312.58	76357.45	76336.21	1879.21
	MSRFA	76147.75	76352.68	76249.83	1959.62
	HIGA	76012.53	77403.51	76438.67	24503.3
50	FA	75654.24	76214.56	75982.24	1586.52
	CFA	75632.45	75727.87	75687.64	1549.58
	MSRFA	75563.74	75732.54	75665.58	1654.93
	HIGA	75858.38	77811.49	76157.21	331100
100	FA	75789.56	76154.65	75973.56	1879.56
	CFA	75695.31	75768.27	75729.86	2578.65
	MSRFA	75546.27	75768.96	75664.73	1956.58
	HIGA	75717.74	75877.21	75788.14	2441.64
300	FA	75712.56	76429.76	76143.81	2349.67
	CFA	75697.37	76028.48	75849.61	1849.51
	MSRFA	75459.42	75548.37	75502.18	2102.57
	HIGA	75623.28	75720.46	75664.68	4688.67

From the experimental results in Tables 15 and 16, we can see that the MSRFA has better search accuracy compared with other algorithms with the same population size. MSRFA can obtain the optimal value when the population size is 50, 100 and 300; when the population size is 30, it fails to obtain the optimal value, but its average value is optimal, which indicates that MSRFA reduces the influence of initialization or other factors on the results and enhances the stability of the algorithm. In Table 15, with the expansion of the population size, the MSRFA further improves its optimal search ability. At the population size of 100 and 300, the optimal solutions obtained by the MSRFA are much larger than those obtained by other algorithms, indicating that the introduction of rough data inference helps the algorithm to improve its optimal search ability when the population size is expanded. Looking at the experimental results in Tables 15 and 16, it is found that the MSRFA performs better under the trend of increasing the number of units and the population size. It can not only search for the optimal solution, but also get the optimal mean value, in addition to that, it can also get a better variance value, which indicates that the population convergence is higher when the MSRFA is obtaining the optimal solution.

The test results for the 10-unit system are given in Table 17. Figure 4 shows the specific scheduling scheme of the MSRFA method proposed in this paper. The experimental data from the literature [32] will be cited for comparison in this test.

Table 16. Experimental results of four algorithms in 5 sets with different dimensions.

Size	Algorithm	Best/$	Worst	Average	Variance
30	FA	41332.20	41425.96	41375	866.22
	CFA	41281.08	41321.56	41304.37	110.69
	MSRFA	40575.92	40765.27	40709.14	742.95
	M-GA	40498.67	40920.28	40721.15	34731.18
50	FA	41306.84	41351.97	41329.77	1180.38
	CFA	41267.98	41306.52	41282.69	1451.14
	MSRFA	40635.34	40719.68	40680.11	728.71
	M-GA	41902.52	41951.27	41661.29	1614.23
100	FA	40557.86	40677.41	40619.97	804.11
	CFA	40292.15	40341.03	40312.19	257.77
	MSRFA	40252.18	40292.29	40271.91	212.19
	M-GA	40361.98	40411.44	40357.28	242.81
300	FA	40287.60	40338.87	40312.33	419.11
	CFA	40279.60	40659.60	40469.75	355.26
	MSRFA	40248.47	40299.61	40267.06	227.06
	M-GA	40558.24	40598.83	40619.34	1096.54

The optimization results of the four algorithms show that compared with FA and CFA methods, the MSRFA method proposed in this paper saves 30 and 22% of the cost, respectively, and obtains a better scheduling scheme; compared with the DE-PSO method, it saves $31,255. Although the cost saving is only 3.9%, the algorithm proposed in this paper takes only 7.88 min in terms of running time, which is The time required for scheduling is greatly reduced.

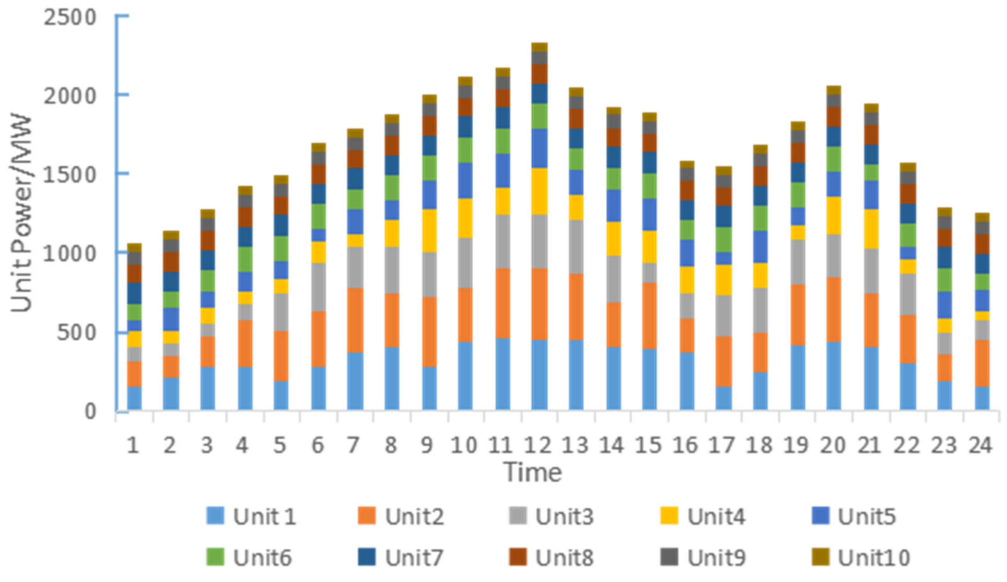


Figure 4. Unit system output value.

Table 17. Results of algorithms related to 10 units.

Algorithm	Best/$	Average	Variance	Running time/min
FA	1047316.54	1110476.49	965423.51	9.26
CFA	956279.18	981254.37	756316.84	10.61
MSRFA	782701.05	784457.31	758253.94	7.88
DE-PSO	813956.81	852325.46	896611.63	10.14

The above data show that for the DED problem, the MSRFA has the following advantages over other algorithms that introduce the idea of contrastive learning strategy with coarse data inference.

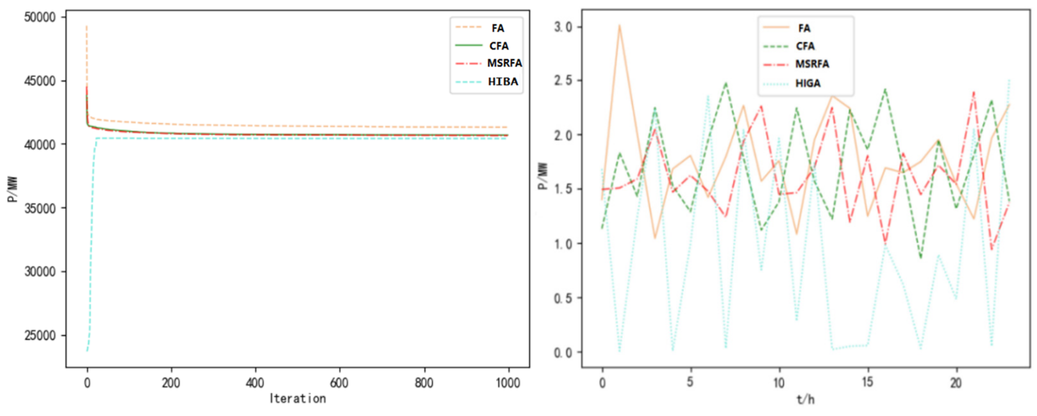
1) Introducing cubic mapping to initialize the population, which is superior to the traditional logistic mapping, this mapping can make a great improvement in the population diversity within the unit output. Thus, it can help the algorithm to have some improvement in the optimization seeking ability, and it can make the algorithm free from the influence of initial values to a certain extent, which is beneficial to the improvement of the stability of the algorithm.

2) MSRFA introduces the idea of rough data inference, and uses the way of relational inference to optimize the way of individual movement; standard FA, each individual moves directly like brighter individuals. MSRFA, before the individual movement, divides the current individual using the approximation rule, and after the division, uses the inference rule of rough data inference, among multiple approximation relations to find the optimal moving trajectory. In this way, the number of moves per traversal is reduced to a certain extent and the complexity of the algorithm is reduced. At the same time, using such rules can make the algorithm have a better optimization finding ability.

3) The traditional FA, when the algorithm falls into the local optimum, there is no good strategy to help the algorithm jump out of the optimum, which is an important reason why the FA results are not stable. The improved MSRFA, when the algorithm falls into a local optimum, triggers the algorithm's get rid of mechanism to help the algorithm jump out of the local optimum. This mechanism uses the idea of equivalence relation reasoning in rough data inference to help the algorithm jump out of the local optimum to some extent by calculating the equivalence class of the current unit output power when the algorithm falls into the local optimum, and then by using the equivalence class to replace the current individual. MSRFA optimization algorithm, using the way of rough data inference, discovers the potential relationship between the unit output power, thus making the algorithm has a good optimization finding capability.

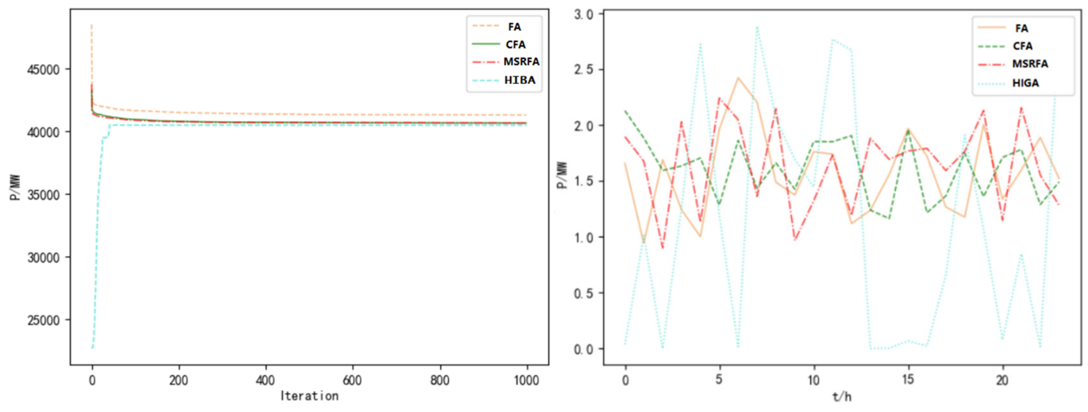
Figures 5 and 6 show the convergence curves and errors of the algorithms for population dimensions of 30 and 100. For the four algorithms, the dimension is uniformly 50 and the number of iterations is 1000 for 30 independent runs. The horizontal axis is the number of iterations and the vertical axis is the value of the algorithm's fitness, which is the total generation cost of the 3 units. From the figure, it can be seen that the initial values of FA, CFA and MSRFA are similar; the initial value of HIGA is very low, but at this time the value of HIGA does not satisfy the error constraint in the actual simulation, and the HIGA will gradually reduce its error value with iterations, so its convergence curve is in a rising state. As can be seen from the figure, the initial position of the convergence curve of the MSRFA is smaller than that of the FA during the early iterations, and has a higher search capability. It can find a better solution in a limited number of iterations, and for the DED problem, it can obtain a lower generation cost. MSRFA can achieve the optimal value when the number of populations is 100, and the curve zigzags as the number of iterations increases, indicating that the algorithm jumps out of the local optimum several times. And it can be seen from the error curve plots in different dimensions that the improved algorithm has more stable error values and smaller mean values. In summary, it can be concluded that the improved algorithm is more suitable for scheduling the unit output in the DED problem, so as to achieve cost reduction.

From the above data, we can see that for this simulation experiment, MSRFA has better performance in providing in providing the optimal solution while ensuring the stability of the algorithm and obtaining the optimal average value. Through the DED simulation test, we found that the MSRFA optimization algorithm has better stability and global search capability. From the obtained results, MSRFA with the introduction of coarse data inference can well avoid falling into local optimum too early and thus obtain good convergence results. When the population dimension increases, or the objective function becomes complex, MSRFA can obtain solutions of better quality than other comparative algorithms. This can also show that it is effective for us to introduce the idea and method of rough data inference to optimize the iterative process of the algorithm, as well as to use the method of equivalence relation inference to improve the measures of the algorithm when it falls into a local optimum.



(a) Convergence curve (b) Error curve

Figure 5. Convergence curve and error curve of 30-dimensional algorithm in 3 units.



(a) Convergence curve (b) Error curve

Figure 6. The 100-dimensional algorithm convergence curve and error curve.

6.3. Simulation time efficiency analysis

Table 18 shows the comparison of the FA, CFA, MSRFA and GA in terms of computational time for the DED problem. Each algorithm is run independently 50 times with 1000 iterations and a function dimension of 50, and the mean value of the time required for the test algorithm to complete one cycle of scheduling for 5 groups. From the table, it is clear that the MSRFA algorithm outperforms the other methods in terms of time simulation. The above data show that the MSRFA algorithm effectively improves the search efficiency and reduces the search time.

Table 18. Time comparison of the four algorithms on the DED problem.

Algorithm	Time (s)
FA	567.88
CFA	578.65
MSRFA	431.32
HIGA	489.34

7. Conclusions

In order to solve the DED problem, an improved firefly optimization algorithm MSRFA is proposed, and its improvement measures are mainly three, 1) initialize the population by using a contrastive learning strategy to increase the population diversity in the search range and obtain a better initial solution; 2) propose the attractiveness selection strategy combined with greedy algorithm, by screening the individuals with greater attractiveness to move in the beginning of the algorithm accelerates the convergence of the algorithm; 3) proposes to combine rough inference with the firefly algorithm to optimize the iterative process of the algorithm by using associative relational inference under approximate rules; 4) uses equivalence relational inference to enhance the ability of the algorithm to jump out of the local optimum. The comparison of MSRFA with other three algorithms to solve the DED problem in the paper verifies that MSRFA has better convergence, global search ability and stability, and is an efficient method to solve the DED problem. At the same time, we realize that realistically, the cost function of the DED problem is influenced by various factors, such as: generator equipment wear and tear, aging, and other issues. When the accurate mathematical model is not available, the swarm intelligence algorithm will not be able to have a better performance in solving this problem. The literature [33] proposes a method to fit the cost function of the DED problem using reinforcement learning, after which we will try to combine this method with the swarm intelligence algorithm to solve the DED problem with uncertain objective function.

Acknowledgments

The work was supported by Tianyou Innovation Team of Lanzhou Jiaotong University (TY202003).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. A. M. Malyscheff, D. Sharma, S. C. Linn, J. N. Jiang, Challenges towards an improved economic dispatch in an interconnected power system network, *Electr. J.*, **32** (2019), 44–49. https://doi.org/10.1016/j.tej.2019.01.013
2. G. Chen, X. Ding, E. Bian, Application of a dynamic differential evolution algorithm based on chaotic sequence in dynamic economic dispatching of power system, *China Power*, **49** (2016), 101–106. http://doi.org/10.11930/j.issn.1004-9649.2016.06.101.06
3. B. Mohammadi-Ivatloo, A. Rabiee, A. Soroudi, Nonconvex dynamic economic power dispatch problems solution using hybrid immune-genetic algorithm, *IEEE Syst. J.*, **7** (2013), 777–785. https://doi.org/10.1109/JSYST.2013.2258747
4. C. L. Chiang, Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels, *IEEE Trans. Power Syst.*, **20** (2005), 1690–1699. https://doi.org/10.1109/TPWRS.2005.857924
5. D. X. Zou, S. Li, Z. Li, X. Kong, A new global particle swarm optimization for the economic emission dispatch with or without transmission losses, *Energy Convers. Manage.*, **139** (2017), 45–70. https://doi.org/10.1016/j.enconman.2017.02.035
6. P. Somasundaram, N. M. J. Swaroopan, Fuzzified particle swarm optimization algorithm for multi-area security constrained economic dispatch, *Electr. Power Compon. Syst.*, **39** (2011), 979–990.
7. W. Yang, Z. Peng, Z. Yang, Y. Guo, X. Chen, An enhanced exploratory whale optimization algorithm for dynamic economic dispatch, *Energy Rep.*, **7** (2021), 7015–7029.
8. Y. T. K. Priyanto, M. F. Maulana, A. Giyantara, Dynamic economic dispatch using chaotic bat algorithm on 150kV Mahakam power system, in *2017 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, (2017), 116–121. https://doi.org/10.1109/ISITIA.2017.8124065
9. R. Keswani, H. K. Verma, S. K. Sharma, Dynamic economic load dispatch considering renewable energy sources using multiswarm statistical particle swarm optimization, in *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, (2020), 405–410. https://doi.org/10.1109/GUCON48875.2020.9231171
10. Q. Iqbal, A. Ahmad, M. K. Sattar, S. Fayyaz, H. A. Hussain, M. S. Saddique, Solution of non-convex dynamic economic dispatch (DED) problem using dragonfly algorithm, in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, (2020), 1–5. https://doi.org/10.1109/ICECCE49384.2020.9179177
11. X. S. Yang, Firefly algorithms for multimodal optimization, in *Proceedings of the 5th Internationa Conference on Stochastic Algorithms: Foundations and Applications*, (2009), 169–178. https://doi.org/10.1007/978-3-642-04944-6_14
12. H. Zhuo, Q. Chen, H. He, Development of welding process expert system based on firefly neural network, *Mechatron. Technol.*, **4** (2020), 27–29. http://doi.org/10.19508/j.cnki.1672-4801.2020.04.008
13. J. Lai, S. Liang, Optimization of wireless sensor network coverage based on improved artificial firefly algorithm, *Comput. Meas. Control.*, **22** (2014), 1862–1864. http://doi.org/10.3969/j.issn.1671-4598.2014.06.062

14. J. Wang, Z. Wang, J. Chen, X. Wang, X. Wang, L. Tian, Microgrid source-load game model and analysis based on firefly optimization algorithm, *Autom. Electr. Power Syst.*, **38** (2014), 7–12. http://doi.org/10.7500/AEPS20131127010
15. X. Pei, R. Zhang, X. Yu, Hybrid firefly algorithm for multi-object replacement flow shop scheduling problem, *Inf. Control*, **4** (2020), 478–488.
16. J. Zhao, W. Chen, R. Xiao, J. Ye, Firefly algorithm with division of roles for complex optimal scheduling, *Front. Inf. Technol. Electr. Eng.*, **10** (2021), 1311–1332.
17. J. Zhang, X. Li, Research on intelligent production line scheduling problem based on Lévy firefly algorithm, *Comput. Sci.*, **48** (2021), 668–672. http://doi.org/ 10.11896/jsjkx.210300118
18. J. Tal, *Research on multi-objective task scheduling problem of cloud computing based on improved particle swarm algorithm*, Master thesis, Hefei University of Technology, 2020.
19. J. Yan, Z. Pan, J. Tan, H. Tian, Water quality evaluation based on BP neural network based on firefly algorithm, *South-to-North Water Diversion Water Sci. Technol.*, **4** (2020), 104–110.
20. Y. Sun, Z. Liu, Application of convolutional networks optimized by firefly algorithm in image saliency detection, *Comput. Digital Eng.*, **48** (2020), 1474–1478. http://doi.org/10.3969/j.issn.1672-9722.2020.06.040
21. W. Liu, Y. Sun, Y. An, X. Gao, C. Sun, Optimization of vehicle routing problem based on FA-IACS algorithm, *J. Shenyang Univ. Technol.*, **42** (2020), 442–447. http://doi.org/10.7688/j.issn.1000-1646.2020.04.16
22. H. Zhang, J. Yang, J. Zhang, X. Xu, Energy management optimization of on-board fuel cell DC microgrid based on multiple firefly algorithm, *Proc. Chin. Soc. Electr. Eng.*, **41** (2021), 13. http://doi.org/10.13334/j.0258-8013.pcsee.201117
23. S. Yan, *Research and application of rough data reasoning based on upper approximation*, Ph. D thesis, Beijing Jiaotong University, 2017.
24. L. Zuo, Y. Yu, H. Sun, Research on dynamic environmental economic dispatch model of power system, *J. East China Jiaotong Univ.*, **35** (2018), 134–142. https://doi.org/10.16749/j.cnki.jecjtu.2018.03.020
25. X. Jiang, J. Zhou, H. Wang, Y. Zhang, Modeling and solving economic dispatch of power system dynamic environment, *Power Grid Technol.*, **37** (2013), 385v391.
26. Y. Zhu, *Research on environmental economic optimal dispatch of power system*, Ph. D thesis, Zhengzhou University, 2016
27. J. Chang, J. Roddick, J. Pan, S. Chu, A parallel particle swarm optimization algorithm with communication strategies, *J. Inf. Sci. Eng.*, **21** (2005), 809–818.
28. Y. Feng, J. Liu, Y. He, Dynamic population firefly algorithm based on chaos theory, *Comput. Appl.*, **54** (2013), 796–799. https://doi.org/10.3724/SP.J.1087.2013.00796
29. J. J. Liang, B. Y. Qu, P. N. Suganthan, A. G. Hernández-Díaz, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, in *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nan yang Technological University, Singapore*, (2013), 281–295.
30. J. Tvrdík, R. Poláková, Competitive differential evolution applied to CEC 2013 problems, in *2013 IEEE Congress on Evolutionary Computation*, (2013), 1651–1657. https://doi.org/10.1109/CEC.2013.6557759
31. Y. J, Y. Fang, Q. Li, Multi-objective genetic algorithm for solving economic load allocation of power system, *East China Electr. Power*, **40** (2012), 648–651.

32. S. Kong, *Research on dynamic economic dispatch of power system based on particle computing*, Master thesis, Yanshan University, 2020.
33. P. Dai, W. Yu, G. Wen, S. Baldi, Distributed reinforcement learning algorithm for dynamic economic dispatch with unknown generation cost functions, *IEEE Trans. Ind. Inf.*, **16** (2019): 2258–2267. https://doi.org/10.1109/TII.2019.2933443