Improved Dragonfly Algorithm Based on Mixed Strategy

Shenyang Xia(✉) and Xing Liu

CCCC Second Highway Consultants Co. Ltd., Wuhan 430056, China
xiasy@cccc.com

Abstract. With the rapid development of today, the swarm intelligence optimization algorithm is very popular and has been used in many fields. Dragonfly algorithm (DA) is one of the optimization algorithms, which has been used in some aspects. But it still has some shortcomings, such as slow convergence speed and search precision, and it is also prone to fall into local optimal solutions. Aiming at the shortcomings of the original dragonfly algorithm, a mixed strategy improved dragonfly algorithm (MSDA) is proposed. The algorithm introduces some improvement strategies. Firstly, initialize the population with the Sobol sequence. This allows the algorithm to obtain a better initial population, improve the quality of initial solution. Secondly, inertia weight improvement, use a non-linear decreasing inertia weight. The modification of inertia weight makes the algorithm better adapt to the convergence process. Then using Cauchy mutation to increase the diversity of the population, improve the global search ability of the algorithm, and increase the search space. Finally, a random learning strategy is added. The random learning strategy enhances the diversity of the population, effectively improve the global optimization performance of the algorithm. The experiment is tested by eight standard test functions, results show that MSDA is better than the original DA algorithm in terms of convergence speed, solution accuracy and stability.

Keywords: Optimization · Swarm intelligence · Dragonfly algorithm

1 Introduction

In nature, if every creature wants to survive, they must have their own ability to survive. Under the screening of nature, only the unique survival ability that can adapt to nature can be preserved. Nowadays, people can observe the ability of various animals in nature, get inspiration from it, and come up with many new ideas that can solve practical optimization problems. This is the origin of the swarm intelligence optimization algorithm. The swarm intelligence optimization algorithm is an evolutionary algorithm based on random search. It mainly simulates the group behavior of insects, beasts, birds and fish. The main behavior of these groups is to forage and avoid enemies. Look for food and constantly exchange food information, so that those animals can find more food faster and avoid the enemy's attack. Studying these swarm behaviors and abstracting an algorithm is the swarm intelligence algorithm.

https://doi.org/10.1007/978-981-97-0730-0_11

The swarm intelligence optimization algorithm is an approximate optimization algorithm, which has aroused great interest of many researchers in computer science, engineering, medicine and other fields [1, 2]. This is because these algorithms can solve various complex optimization problems in a relatively short period of time. Since some complex tasks cannot be solved in a short period of time. And at the same time swarm intelligence optimization algorithms are simple, flexible and robust [3].

Dragonfly Algorithm (DA) is a new swarm intelligence optimization algorithm proposed by scholar Seyedali Mirjalili in 2015. Dragonflies are a relatively primitive and relatively few species of insects. There are about 5000 species in the world. They are carnivorous insects. They prey on a variety of agricultural, forestry and animal husbandry pests such as flies, mosquitoes, leafhoppers, horseflies and small butterfly moths. Dragonflies are a class of important natural enemy insects that are beneficial to humans. The main inspiration of the DA algorithm comes from the static and dynamic flocking behavior of dragonflies in nature, that is the behavior of foraging groups and the behavior of migratory groups. The algorithm is simple in principle, easy to understand and easy to implement, has strong search ability, and has been widely studied and discussed by scholars from many countries. The Dragonfly algorithm has been well applied and is currently used to solve various optimization problems, such as image processing, medicine, computer science, engineering, etc. And the dragonfly algorithm has shown excellent performance in these optimization problems. But the algorithm still has some common defects, such as low accuracy, premature convergence, and insufficient global search ability.

In view of the above shortcomings of the Dragonfly algorithm, this paper improves it and proposes an improved Dragonfly algorithm based on mixed strategy (Mixed Strategy Improved Dragonfly Algorithm, MSDA). At the beginning of the algorithm, the Sobol sequence is used to initialize the population to make the distribution in the initial solution space more uniform. This allows the algorithm to obtain a better initial population, improve the quality of initial solution. Then, the global search force and local development force are improved by nonlinearly decreasing inertia weight. The modification of inertia weight makes the algorithm better adapt to the convergence process. Using Cauchy mutation to increase the diversity of the population, improve the global search ability of the algorithm, and increase the search space. And the position of the individual is adjusted by a random learning strategy. The random learning strategy enhances the diversity of the population, effectively improve the global optimization performance of the algorithm. Through the test experiments of 8 benchmark functions. The experimental results show that compared with the original Dragonfly algorithm and some other algorithms, this algorithm has better search power in global search and local development, and has a certain improvement in performance.

2 Related Work

The source of inspiration for the dragonfly algorithm is the behavior of dragonflies in nature. Dragonflies use these behaviors to ensure their own survival, which is a simulation of dragonflies in nature. Dragonflies have social behaviors among individuals. Different from the hierarchical social behaviors of insects such as bees and ants, each individual dragonfly has corresponding activities for its survival. The biggest problem in its survival is the source of food and the avoidance of natural enemies. Under the above conditions, individual dragonflies will follow the society in which they live. Behavior, keeping moving while searching for food and avoiding predators.

In the dragonfly optimization algorithm, dragonfly individual behaviors are mainly described as 5 kinds, they are:

1) Separation behavior. It is a behavior that avoids collision between individual, Each individual in the group tends to maintain the same speed of movement;
2) Alignment behavior. It is the behavior of maintaining speed consistency between adjacent individuals;
3) Cohesion behavior. That is the behavior of individuals moving toward the average position of adjacent individuals;
4) Foraging behavior is that the individual is attracted by the food source;
5) Enemy avoidance behavior is the behavior of the individual to avoid natural enemies. Each behavior has its corresponding weight.

The behaviors of foraging behavior and enemy avoidance behavior are mainly included in the static group behavior. In static group behavior, there is a change in step size, while dragonflies in small groups are in a state of local movement. Dynamic flock behavior is when large numbers of dragonflies come together to make long-distance migrations in the same direction. In the dynamic group behavior, there are three main behaviors: separation behavior, alignment behavior and cohesion behavior. They fly in the same direction, which is also a major feature of dynamic group behavior. Each dragonfly in the group is equivalent to the solution of the search space (Fig. 1).

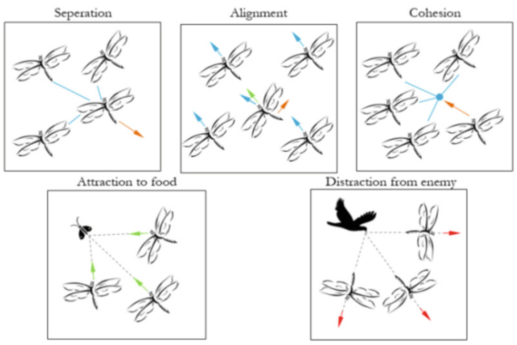

Fig. 1. Inception V1 structure

These five behaviors are the most basic and the most important. Each of these behaviors is essential, and if one of them is missing, the dragonflies will not be able to survive. Dragonflies survive under the above behavior. This is their unique survivability, with which they have survived nature's long sifting. Similar to most swarm intelligence optimization algorithms, the dragonfly optimization algorithm imitates the behavior of individual dragonflies in nature. The Dragonfly algorithm mathematically describes the above five behaviors. The specific mathematical description is as follows:

- Separation

Separation behavior is also known as collision avoidance behavior, it represents the degree of separation, refers to the avoidance behavior between adjacent individuals, and the mathematical expression is as follows:

$$S_i = -\sum_{j=1}^{N} X - X_j \tag{1}$$

Here X represents the position of the current individual, and X_j represents the position of the j-th adjacent individual. N represents the number of adjacent individuals.

- Alignment

The alignment behavior, it represents the degree of alignment. The behavior of individuals tending to the same speed, and the mathematical expression is as follows:

$$Ai = \frac{\sum_{j=1}^{N} Vj}{N} \tag{2}$$

Among them, V_j represents the velocity of the j-th adjacent individual.

- Cohesion

Aggregation behavior, it represents the degree of cohesion, refers to the behavior of individuals tending towards the center of the population, and the mathematical expression is as follows:

$$Ci = \frac{\sum_{j=1}^{N} Xj}{N} - X \tag{3}$$

where X is the position of the current individual, N is the total number of adjacent individuals, and X_j represents the position of the j-th adjacent individual.

- Foraging

Foraging behavior, that is, the attraction degree of food, dragonflies are attracted by food and move towards food, and the mathematical expression is as follows:

$$Fi = X^{+} - X \tag{4}$$

where X is the current location of the individual and X^{+} is the location of the food. The location of the food is selected from the currently found optimal solution.

- Enemy avoidance

The avoidance behavior of enemies, that is, the repelling force of enemies, the behavior of dragonflies away from enemies when threatened by enemies, and the mathematical expression is as follows:

$$Ei = X^{-} + X \tag{5}$$

where X is the current location of the individual and X^{-} is the location of the enemy. The location of the enemy is selected from the worst solution found so far.

Dragonflies are developed according to the above five behaviors, and the step size update formula of the offspring dragonflies is as follows:

$$\Delta Xt + 1 = (sSi + aAi + cCi + fFi + eEi) + \omega \Delta Xt \tag{6}$$

Among them, *s, a, c, f,* and *e* represent the weights of separation, alignment, cohesion, foraging, and enemy, respectively, and ω represent the inertia weight, and *S, A, C, F,* and *E* represent the degree of the above five behaviors. The t is the current number of iterations. The position of the next offspring dragonfly individual is expressed by the following formula:

$$Xt + 1 = Xt + \Delta Xt + 1 \tag{7}$$

In order to judge whether there are adjacent individuals around the dragonfly, draw a circle with radius r around the dragonfly individual as the search radius of the dragonfly. Individuals in the circle are considered adjacent. As the number of iterations increases, the search radius of the dragonfly is also updated. The update formula of the search radius is as follows:

$$r = \frac{a-b}{4} + 2(a-b)\frac{t}{t\max} \tag{8}$$

Among them, r is the search radius of dragonfly, t is the current number of iterations, $t_{\max}$ is the maximum number of iterations, and a and b are the upper and lower limits of the search range, respectively. If there are no adjacent individuals around the dragonfly, perform the Levy flight behavior of random walk:

$$X_{t+1} = X_t + \text{Levy}(d) \times X_t \tag{9}$$

where t represents the number of iterations of the current iteration, and d represents the dimension of the current position vector. Levy flight is a random walk strategy, where each step can be displaced in any direction and by any length. Many creatures in nature have similar laws in their activities, and their Levy function is:

$$\text{Levy}(x) = 0.01 \times \frac{r1 \times \sigma}{|r2|^{1/\beta}} \tag{10}$$

where r_1 and r_2 is a random number in the range [0,1], β is a constant (equal to 1.5 in the original dragonfly algorithm), and σ is expressed as:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\beta-1)/2}} \right)^{1/\beta} \tag{11}$$

3 Mixed Strategy Improved Dragonfly Algorithm

3.1 Initialization Population Based on Sobol Sequence

In many swarm intelligence optimization algorithms, the state of the initial distribution of the population will affect the performance of the next algorithm, and the same is true of the Dragonfly algorithm. In the original DA algorithm, in the initialization phase of the algorithm, the random generation of the initialization population is used. However, the distribution of randomly generated population states is relatively uneven and cannot cover every great point well, which will affect the subsequent performance of the algorithm. In order to improve this defect, this paper discards the original random initialization population, and adopts the initialization population based on Sobol sequence. The Sobol sequence is a low variance and random sequence with good distribution uniformity, which can generate a relatively uniform distribution in the probability space, resulting in non-repetitive and uniform points. The figure below compares the original random number distribution of DA with the spatial distribution of random numbers generated by the Sobol sequence. In the case of population number N = 500, the randomly generated initialization population and the population distribution map generated by Sobol sequence. It can be seen that the population distribution obtained by the Sobol sequence is more uniform and the coverage of the solution space is more complete (Fig. 2).

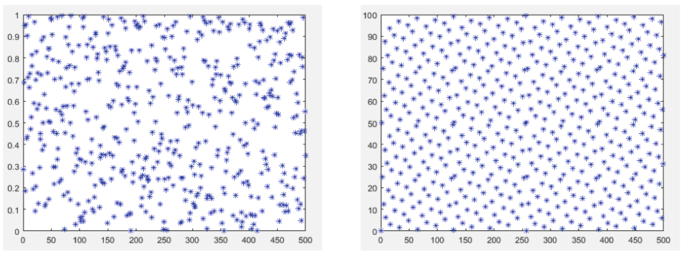
(a) Random generation (b) Sobel generation

Fig. 2. Random generation sequence and Sobel generation sequence

3.2 Nonlinearly Decreasing Inertia Weights

Inertial weights are used to simulate inertia in real-world physics, enabling objects to maintain their original speed and continue to fly. In the swarm intelligence optimization algorithm, the algorithm should change the corresponding inertia weight at different times to make the algorithm more adaptable to the current changing state. In the original DA algorithm, a linearly decreasing inertia weight is used, and the calculation formula of the inertia weight is as follows:

$$w = 0.9 - \frac{(0.9 - 0.4)t}{t\max} \tag{12}$$

In this formula, t_{max} is the maximum number of iterations, t is the number of iterations in the current iteration. 0.9 and 0.4 represent the upper and lower bounds of the inertia weight, respectively.

Since the original DA uses a linearly decreasing inertia weight, and the search process of the algorithm is not linear, in the later stage of the search, the inertia weight decreases too fast, and the search ability and development ability of the algorithm are not utilized accordingly. Therefore, this paper changes the linear decrease of the original DA algorithm to nonlinear, and uses the following formula to express:

$$W = fc - fc \times (2 \times (t/t\max)\exp(h) - ((t/t\max)^2)) \tag{13}$$

where fc and h are the upper limit of the inertia weight and the lower limit of the inertia weight, $fc = 0.9$, $h = 0.4$, t_{max} is the maximum number of iterations, and t is the number of iterations at the current time. The curve of the above formula is represented by the following figure (Fig. 3):

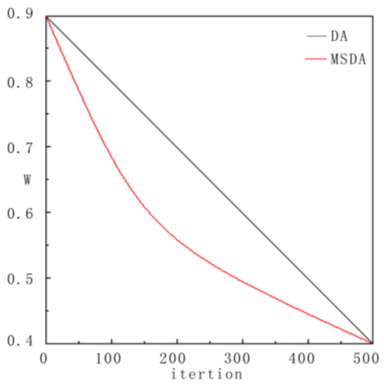

Fig. 3. Modified inertia weight curve

3.3 Random Learning Strategy

In order to enrich the population information and increase the information sharing between individuals among the populations, this paper adds a random learning strategy to the original DA. Randomly select an individual x from the population, and a different individual x_p, and determine the better individual by comparing the fitness values of the two. Adjust the position of individuals by learning from better individuals.

$$xnew = \begin{cases} x + rand(0,1) \times (x - xp), f(xp) < f(x) \\ x + rand(0,1) \times (xp - x), f(xp) \geq f(x) \end{cases} \tag{14}$$

The learning factor rand(0,1) is a random number between 0 and 1, which represents the learning difference of different individuals. After learning, if $f(x_p) < f(x)$, accept the new individual X_{new} and replace the original individual, otherwise reject the new individual X_{new}. This strategy increases the information sharing between individuals, increases the diversity of the population, and can effectively improve the global search ability.

3.4 Cauchy Mutation Strategy

Using Cauchy mutation to increase the diversity of the population, improve the global search ability of the algorithm, and increase the search space. The peak value of the Cauchy distribution function at the origin is small, but the distribution at both ends is relatively long, which can generate greater disturbance to the current individual, making it easier for the algorithm to jump out of the local optimal solution and enhancing the global search ability of the algorithm. In this paper, the following Cauchy variation formula is used to update the position of the current optimal individual:

$$x_{newbest} = x_{best} + x_{best} \times Cauchy(0, 1) \tag{15}$$

where $x_{newbest}$ is the new value obtained after the current optimal value is disturbed by Cauchy mutation. Cauchy(0,1) is the Cauchy operator, and the standard Cauchy distribution function formula is as follows:

$$f(x) = \frac{1}{\pi \times (x^2 + 1)}, x \in (-\infty, +\infty) \tag{16}$$

Compared with other functions, the peak of the Cauchy distribution is lower, which can shorten the time spent by individuals in the search space, thereby speeding up the convergence speed of the algorithm. The extension of the two ends of the Cauchy distribution can generate random numbers farther from the origin, which can generate strong disturbance to the individual, so that the disturbed individual has the ability to quickly avoid local traps.

3.5 Algorithm Implementation

After the above improvements, the algorithm flow of this paper is as follows:

Step 1: Initialization algorithm, dragonfly population size N, maximum iteration number MIT, problem dimension D, neighborhood radius r, step vector Δx;
Step 2: Use Sobol sequence to generate initialization population, and set t = 1;
Step 3: Calculate the moderate value of each individual;
Step 4: Adjust the individual position through the random learning strategy of Eq. (14);
Step 5: Update the behavior weights s, a, c, f, e;
Step 6: Use the nonlinear decreasing strategy of Eq. (13) to update the inertia weight;
Step 7: Calculate S, A, C, F, E;
Step 8: Update the neighborhood radius r, if there are adjacent individuals, use Eq. (9), (10) Update the step vector, otherwise use the Levy flight to update the position vector;
Step 9: Perturbation of individuals using the Cauchy mutation strategy;
Step 10: The number of iterations is increased by one. If the maximum number of iterations is reached, the optimal solution will be output, otherwise, it will return to Step 3.

The above algorithm flow can be represented by the following chart (Fig. 4):

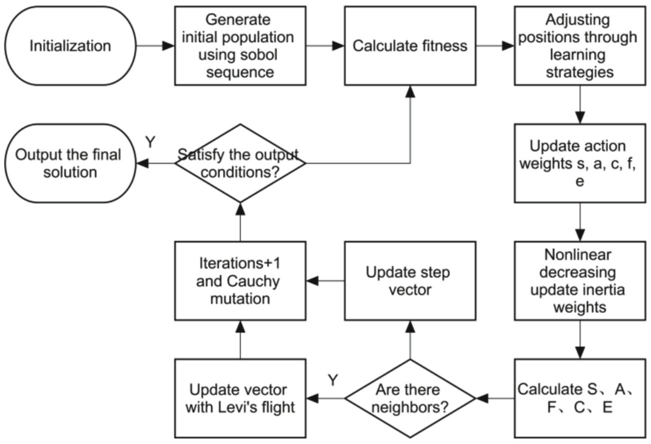
Initialization
Generate initial population using sobol sequence
Calculate fitness
Adjusting positions through learning strategies
Output the final solution
Y
Satisfy the output conditions?
Update action weights s, a, c, f, e
Iterations+1 and Cauchy mutation
Update step vector
Nonlinear decreasing update inertia weights
Update vector with Levi's flight
Y
Are there neighbors?
Calculate S、A、F、C、E

Fig. 4. MSDA algorithm flow chart

4 Experiment

In the experiment, this paper introduces the original Dragonfly Algorithm (DA), Grasshopper Optimization Algorithm (GOA), Particle Swarm Optimization algorithm (PSO), and this algorithm (MSDA) for comparative experiments. The settings of the comparative experiments are as follows: the population size N = 30, the maximum number of iterations MIT = 500, and the dimensions are all set to 30.

In order to verify the improvement effect of the MSDA algorithm, four different standard single-peak test functions and four different standard multi-peak test functions are used for verification tests. The specific function types and attributes are shown in the following table (Table 1):

Table 1. List of test functions

Test functions	Type	Range	Min	Dim
$F1(x) = \sum_{i=1}^{n} xi^2$	Unimodel	[−100,100]	0	30
$F2(x) = \sum_{i=1}^{n} \lvert xi\rvert + \prod_{i=1}^{n} \lvert xi\rvert$	Unimodel	[−10,10]	0	30
$F3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} xj\right)^2$	Unimodel	[−100,100]	0	30
$F4(x) = \max i\{\lvert xi\rvert, 1 \le i \le n\}$	Unimodel	[−100,100]	0	30
$F_5(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	Multimodel	[−5.12,5.12]	0	30

(*continued*)

Table 1. (*continued*)

Test functions	Type	Range	Min	Dim
$F_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	Multimodel	[−32,32]	0	30
$F_7(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Multimodel	[−600,600]	0	30
$F_8(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Multimodel	[−50,50]	0	30

The experimental operating environment of this paper is the 64-bit Windows 7. The processor is Intel Core i5-3230M CPU @ 2.60GHz, and the host has 8 GB of installed memory. The simulation software used for the test is MATLAB R2016b.

In order to obtain a more average result in the experiment and reduce the influence of the randomness of the experiment, this experiment will run each test function 10 times independently, and compare the optimal solution, worst solution, mean and std to evaluate the algorithm. The experimental results are shown in following table (Table 2):

Table 2. Experimental structures for different network structures

TF	algorithm	Best	Worst	Mean	Std
F_1	PSO	1.336E−03	27.783	4.348	6.683
	GOA	4.82E−08	4.662E−07	1.266E−07	1.326E−07
	DA	1.993E−05	83.363	7.655	20.803
	MSDA	0	0	0	0
F_2	PSO	4.615E−01	1.2053	7.474E−01	3.675E−01
	GOA	0.021	1.225	0.441	0.402
	DA	0.0271	15.773	6.822	5.237
	MSDA	0	0.0183	0.00183	0.00579
F_3	PSO	5.325E−02	17.335	3.175	5.115
	GOA	82.825	7.242E+03	494.525	1.532E+03
	DA	72.114	793.824	374.258	2.893E + 03
	MSDA	0	0.0943	0.0311	0.0661
F_4	PSO	4.404E−02	1.522	0.952	0.237

(*continued*)

Table 2. (*continued*)

TF	algorithm	Best	Worst	Mean	Std
	GOA	0.469	6.993	2.524	1.211
	DA	0.329	10.593	7.415	3.661
	MSDA	0	0.0251	4.681E−02	9.491E−02
F_5	PSO	37.441	172.249	93.663	23.331
	GOA	21.632	210.367	77.155	21.215
	DA	26.664	29.549	27.972	1.445
	MSDA	0	34.175E−02	7.901E−03	1.321E−02
F_6	PSO	1.543E−03	2.522	9.145E−01	6.122E−01
	GOA	0.007	3.151	1.515	0.853
	DA	3.788	17.843	10.425	5.892
	MSDA	7.994E−15	4.361E−02	5.922E−10	4.992E−10
F_7	PSO	5.263E−02	9.253E−01	3.153E−01	2.954E−01
	GOA	0.037	0.525	0.210	0.052
	DA	1.225	79.385	9.636	17.395
	MSDA	0	1.083E−03	2.092E−04	1.162E−04
F_8	PSO	5.623	9.453	7.633	1.736
	GOA	3.646E−04	0.151	0.052	0.092
	DA	4.663	1.983E+06	6.935E+04	5.364E+05
	MSDA	0.293	0.994	0.468	0.295

To sum up, the MSDA algorithm has obtained better results than other comparison algorithms in both the single-peak test function and the multi-peak test function, and can obtain better solutions under the same conditions.

5 Conclusion

DA has been successfully used in some fields, proving its strength. Compared with other optimization algorithms, it has the characteristics of less parameters, simple control and easy enhancement. But it still has some shortcomings, such as slow convergence speed and search precision, and it is also prone to fall into local optimal solutions. In order to solve these problems, this paper does some work. On the basis of the original dragonfly algorithm, this paper mixes a variety of improved strategies, and proposes an improved dragonfly algorithm MSDA with a mixed strategy. The algorithm introduces some improvement strategies. First, initialize the population with the Sobol sequence. In this way, the algorithm can obtain a better initial population and improve the quality of the initial solution. Secondly, the inertia weight is improved, and the nonlinear decreasing inertia weight is adopted. The modification of inertia weights makes the algorithm better adapt to the convergence process. Then use the Cauchy mutation to increase the diversity of the population, improve the global search ability of the algorithm, and increase the search space. Finally, a random learning strategy is added. The random learning strategy enhances the diversity of the population and effectively improves the global optimization performance of the algorithm. Simulation experiments prove that its performance has a certain improvement compared with the original DA algorithm. The next research direction is towards combining the MSDA algorithm with optimization problems in other domains.

Acknowledgment. This work is funded by the National Natural Science Foundation of China under Grant No. 61772180, the Key R & D plan of Hubei Province (2020BHB004, 2020BAB012).

References

1. Chen, X., et al.: Forecasting short-term electric load using extreme learning machine with improved tree seed algorithm based on Lévy flight. Eksploatacjai Niezawodnosc Maintenance Reliability **24**(1), 153–162 (2022)
2. Sun, S., Przystupa, K., Wei, M., Yu, H., Ye, Z., Kochan, O.: Fast bearing fault diagnosis of rolling element using Lévy Moth-Flame optimization algorithm and Naive Bayes. Eksploatacjai Niezawodnosc Maintenance Reliability **22**(4), 730–740 (2020)
3. Xu, H., Przystupa, K., Fang, C., Marciniak, A., Kochan, O., Beshley, M.: A combination strategy of feature selection based on an integrated optimization algorithm and weighted k-nearest neighbor to improve the performance of network intrusion detection. Electronics **9**(8), 1206 (2020)
4. Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. **27**(4), 1053–1073 (2015). https://doi.org/10.1007/s00521-015-1920-1
5. Tharwat, A., Gabel, T., Hassanien, A.E.: Parameter optimization of support vector machine using dragonfly algorithm. In: Hassanien, A.E., Shaalan, K., Gaber, T., Tolba, M.F. (eds.) AISI 2017. AISC, vol. 639, pp. 309–319. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-64861-3_29
6. Abdel-Basset, M., Luo, Q., Miao, F., Zhou, Y.: Solving 0–1 knapsack problems by binary dragonfly algorithm. In: Huang, D.-S., Hussain, A., Han, K., Gromiha, M.M. (eds.) ICIC 2017. LNCS (LNAI), vol. 10363, pp. 491–502. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63315-2_43
7. Russell, R.W., May, M.L., Soltesz, K.L., et al.: Massive swarm migrations of dragonflies (Odonata) in eastern North America. Am. Midl. Nat. **140**(2), 325–342 (1998)