Nonlinear Dissipative Particle Swarm Algorithm and Its Applications

QINGHE XU, TUNHUA WU, AND WEIWEI WANG
School of Information and Technology, Wenzhou Business College, Zhejiang 325006, China
Corresponding author: Qinghe Xu (247312662@qq.com).

ABSTRACT A nonlinear dissipative particle swarm algorithm is proposed for the problem of poor search accuracy by the particle swarm algorithm, especially for the optimization of high-dimensional functions. The algorithm dissipates the particles in a nonlinear increasing way: avoiding a large amount of unnecessary dissipation at the beginning of the iteration and putting more effort into dissipation at the end of the iteration, which improves the operation efficiency and global search ability of the algorithm. On this basis, the inertia weights are adjusted with a nonlinear decreasing strategy, which improves the search ability of the algorithm when there is no dissipation. The experimental results show that the nonlinear dissipative particle swarm algorithm has superior performance when dealing with high-dimensional function optimization problems and mobile robot path planning problems.

INDEX TERMS particle swarm optimization, nonlinear dissipative, inertia weight, route planning.

I. INTRODUCTION Particle swarm optimization (PSO) [1], which originated from the study of bird predation behavior, is a new intelligence optimization algorithm based on iteration. It initializes a set of random solutions and uses the current position of each particle to evaluate the objective function. At each moment, the particle uses its current position, its own historical best position, and the position of one or more best particles in the swarm to determine the movement of the particle in the search space. The entire population completes one iteration after all particles have finished moving once. Finally, the whole population moves in the direction of the best advantage of the fitness function like a flock of collaborative foraging birds. That is, the particles track the individual extremes searched by the current particles and the global extremes searched by the whole population to search for the optimal value in the solution space. However, unlike the "survival of the fittest" evolutionary idea of the genetic algorithm [2], the particle swarm optimization algorithm searches for the optimal position through the coordination between individual particles. Compared with other bionic algorithms, such as genetic algorithms, the particle swarm algorithm is simple in concept, has few adjustable parameters, and is easy to implement, which has attracted the attention of many scholars at home and abroad and is widely used in solving complex multipeak optimization problems.

At the beginning of the research, the initial PSO algorithm was formed by considering multidimensional search and acceleration according to the distance and by matching the velocity of the neighborhood, and then the concept of inertia weight was introduced to achieve a better search of the solution space, which gradually formed the basic particle swarm algorithm [3] that is now commonly used. However, it is an indisputable fact that the basic particle swarm algorithm has a premature convergence problem. To address this problem, a large number of solutions have been generated. Q. Lv et al. [4] used the pheromone mechanism to design the particle behavior to improve the algorithm's search capability. L. L. Kang et al. [5] designed an inertia-free adaptive elite variational inverse particle strategy. B. Q. Lv et al. [6] used the fuzzy rules used by the control system fuzzy controller to improve the algorithm. L. F. Xu et al. [7] used multi-level perturbation to obtain better optimization search results. S. Yi et al. [8] improved the particle swarm algorithm under semisupervised clustering objectives. H. X. Lu et al. [9] used neural networks to dynamically generate the parameters required for the operation of the algorithm. W. B. Liu et al. [10] proposed a particle swarm algorithm with a weighting strategy based on the sigmoid function to adaptively adjust the acceleration coefficients. X. W. Xia et
al. [11] proposed a triple archive particle swarm algorithm to improve the solution accuracy and convergence speed. In addition, many different versions of particle swarm algorithms have emerged to achieve adaptation and application in different fields. For example, for the typical NP backpack problem, researchers have proposed a discrete particle swarm algorithm based on the chaotic behavior of ant colonies [12]. In the field of new energy, researchers have also redesigned particle swarm algorithms for PV maximum power point tracking and inverter parameter identification [13-14]. In the field of optics, improved particle swarm algorithms are also playing a role in spectral line separation and spectral absorption [15-16].

The various PSO-based improvement algorithms mentioned above have their own merits, but in general, they all try to solve the inherent problem of the original PSO algorithm - the problem of easily falling into a local optimum.

To address this problem, considering a certain balance between the search speed of the algorithm and the maintenance of population diversity, this paper dissipates particles in a nonlinear incremental way, avoiding a large amount of unnecessary dissipation at the beginning of the iteration and devoting more energy to dissipation at the end of the iteration, which improves the operational efficiency and global search ability of the algorithm. Based on this, a nonlinear dissipative particle swarm optimization (NDPSO) algorithm is proposed by using nonlinear functions to adjust the inertia weights to enhance the ability of the population to utilize the information. In general, the main contribution of this algorithm is to overcome the poor performance of the particle swarm algorithm for high-dimensional function optimization and to improve the operational efficiency and global search capability of the algorithm. By introducing six numerical benchmark test functions for experiments and comparing the analysis with basic PSO, DPSO [17] and other intelligent optimization algorithms [18], the results show that NDPSO is an algorithm with higher search accuracy and more stable performance. In addition, the superiority of this algorithm is shown in experiments on the optimization of artificial potential field parameters for mobile robot path planning.

This paper is organized as follows: Section II briefly introduces the dissipative particle swarm algorithm, Section III proposes the nonlinear dissipative particle swarm algorithm and the implementation process and analyzes its convergence, Section IV shows the simulation experiments, and Section V concludes the paper.

II. DISSIPATIVE PARTICLE SWARM ALGORITHM

To address the drawback that the search capability of the PSO algorithm gradually decreases with the number of iterations, the literature [15] introduces a dissipation operation that gives the particles a greater chance of escaping from the local minima, as described by the following equations:

\[
\begin{align*}
\nu_{i+1} &= \omega \nu_i + c_i \xi (p_i^* - x_i) + c_2 \eta (p_g^* - x_i) \quad (1) \\
\chi_{i+1} &= \chi_i + \nu_{i+1} \\
\text{if } (\text{rand}) (\leq c_i) \text{ then } &\nu_i = \text{rand}(\nu_{\text{max}}) \quad (3) \\
\text{if } (\text{rand}) (\leq c_i) \text{ then } &x_i = \text{Random}(l_d, u_d) \quad (4)
\end{align*}
\]

where \( \nu_i = [\nu_{i1}, \nu_{i2}, \ldots, \nu_{in}] \) is the velocity of particle \( i \), which represents the distance between the present position of particle \( i \) and its next target position. \( x_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \) is the current position of particle \( i \). \( p_i^* \) is the individual optimal solution searched thus far. \( p_g^* \) is the optimal solution searched thus far for the whole population. \( \omega \) is the coefficient of maintaining the original velocity, called the inertia weight. \( c_1 \) is the weight coefficient of the particle tracking its own historical optimum, which represents the particle's recommended deleted knowledge, called the individual cognitive constant. \( c_2 \) is the coefficient of particle tracking, which is the weight coefficient of the group optimum. It indicates the particle's knowledge of the whole group and is called the social knowledge constant. \( \xi \) and \( \eta \) are random numbers uniformly distributed in the interval of \([0, 1]\). \( c_v \) and \( c_i \) are jump factors with values of \([0, 1]\). \text{Random}(l_d, u_d) \) is a random number between \( l_d \) and \( u_d \), and \((l_d, u_d)\) is the range of values of the position component.

III. NONLINEAR DISSIPATIVE PARTICLE SWARM ALGORITHM

DPSO is blind in the selection of the jump factor, and the algorithm is too large at the beginning of the operation and produces unnecessary dissipation, which affects the efficiency. Meanwhile, it is too small at the end of the operation because there is no timely dissipation, and the algorithm easily falls into the local optimum. Therefore, this paper proposes a nonlinear dissipative particle swarm algorithm to enhance the management of the population and the particle search method.

A. EQUATIONS PROPOSED FOR THE NDPSO ALGORITHM

As we can see from Equation (1), the magnitude of the velocity indirectly represents the distance of the particle relative to the best position, and the dissipation of the velocity and position in their own way can easily lead to confusion in the population. Therefore, in this paper, the velocity and position of the particles are dissipated according to a unified standard, i.e., the velocity and position of the particles are dissipated simultaneously when a certain condition is reached. At the same time, the dissipation of the particles shows a nonlinear increase with the operation of the algorithm, which is described by the following equation:
if (rand() < k) then \( v_i = \text{rand()}v_{\max}, x_i = \text{Random}(l_{\min}, u_{\max}) \) (5)

Here,

\[ k = (k_{\max} - k_{\min})(\text{iter} / \text{Gen})^n + k_{\min} \] (6)

where \( k \) denotes the jump factor, whose value is \([0, 1]\), \( k_{\max} \) represents the maximum jump factor, \( k_{\min} \) shows the minimum jump factor, \( n \) is a constant, \( \text{Gen} \) means the number of iterations, and \( \text{iter} \) means the number of current iterations.

Since \( \text{rand()} \) belongs to a uniform random distribution between \([0,1]\), a smaller \( k \) value at the beginning of the iteration can avoid a large number of dissipation operations and improve the efficiency of the algorithm; as the number of iterations increases, the speed of particles gradually decreases, the value of the jump factor \( k \) gradually increases, and the dissipation ability of particles is further strengthened, enhancing the diversity of the population and improving the global search ability of the algorithm.

Of course, the dissipation operation is also possible when the value of \( k \) is small, but the small probability of dissipation is more likely to enhance the effect of particle search because at larger velocities, the particles are likely to fly out of the optimal position, thus failing to find the optimal value. In the late iteration, when the value of \( k \) is large, there is also a small probability of no dissipation when the particle is likely to have found the optimal value, thus reducing unnecessary dissipation.

Since the strategy of using nonlinear adjustment of inertia weights is better than the strategy of linear decreasing inertia weights in the performance of the algorithm [19], to improve the algorithmic search capability of particles without dissipation, in this paper, based on the nonlinear dissipation operation, the adjustment of inertia weights is also taken as a nonlinear operation to complement the dissipation operation in an attempt to be able to reflect the particle flight process more reasonably, as described in the following equation:

\[ \omega = - (\omega_{\max} - \omega_{\min})(\text{iter} / \text{Gen})^n + \omega_{\max} \] (7)

where \( \omega \) takes values in the range \([0.95,0.4]\), \( n \) is a constant, \( \omega_{\max} \) denotes the maximum inertia weight value, \( \omega_{\min} \) denotes the minimum inertia weight value, and others are the same as above. The experimental results show that the introduction of Equation (7) enhances the search capability of the algorithm. The experimental results show that the simultaneous nonlinear operation on the dissipation and inertia weights of the particles enables the particles to make full use of the amount of information available to the existing population during the search process, and the particles depend on each other in the search strategy to form a complementary strategy, which improves the algorithm's optimization-seeking ability.

B. CONVERGENCE ANALYSIS OF THE NDPSO ALGORITHM

The NDPSO algorithm proposed in this paper does not change the basic structure of the basic particle swarm algorithm, and the main update mechanism is still similar to the basic particle swarm algorithm, except that the population diversity is increased by nonlinear dissipation of particles, and the inertia weights are adjusted by a nonlinear decreasing strategy to improve the algorithm search performance without dissipation. Thus, the convergence of the algorithm proposed in this paper can be based on the literature [20] method to prove that it will converge to some power balance point in the individual historical optimum and the population optimum. The specific proof is as follows.

To simplify the calculation, let the space of independent variables be 1-dimensional, the number of particles be 1, and the initial conditions be \( x_i^0 = x_0, x_i^1 = x_i \), where \( \phi_1 = x_i, \phi_2 = x_i \). From Equation (2) and Equation (3), the following is obtained:

\[ x_i^{k+1} = (1 + \omega - \phi_1 - \phi_2)x_i^k - \omega x_i^{k-1} + \phi_1 p_i^k + \phi_2 p_g^k \] (8)

The closed form of Equation (8) can be obtained as follows:

\[ x_i^k = k_i + k_1 \alpha^k + k_3 \beta^k \] (9)

The initial conditions lead to \( k_i = (\phi_1 p_i + \phi_2 p_g) / (\phi_1 + \phi_2) \), \( \alpha = (1 + \omega - \phi_1 - \phi_2) / 2 \), and \( \beta = (1 + \omega - \phi_1 - \phi_2) / 2 \), where \( \gamma = [(1 + \omega - \phi_1 - \phi_2)^2 - 4\omega]^{1/2} \). Now consider the extreme value of \( x_i^k \):

\[ \lim_{k \to \infty} x_i^k = \lim_{k \to \infty}(k_i + k_1 \alpha^k + k_3 \beta^k) \]

Clearly, \( \text{max}(\| \alpha \|, \| \beta \|) < 1 \) converges only when \( \{x_i^k\}_{k=0}^\infty \), i.e.,

\[ \lim_{k \to \infty} x_i = \lim_{k \to \infty}(k_i + k_1 \alpha^k + k_3 \beta^k) \]

\[ = k_i = (\phi_1 p_i + \phi_2 p_g) / (\phi_1 + \phi_2) \] (10)

When \( \text{max}(\| \alpha \|, \| \beta \|) < 1 \), we obtain \( \omega > 0.5(\phi_1 + \phi_2) - 1 \). Therefore, the algorithm is guaranteed to converge only when the condition \( \omega > 0.5(\phi_1 + \phi_2) - 1 \) is satisfied and converges to the value of \( K_i \), i.e., converges to some power balance point in the individual historical optimum and the population optimum.

C. NDPSO ALGORITHM IMPLEMENTATION PROCESS

The specific implementation steps of the NDPSO algorithm are as follows:

Step 1: Initialize the particle swarm, i.e., randomly give the initial position and initial velocity of m particles.

Step 2: Given the fitness function, calculate the fitness value of each particle fitness.

Step 3: Determine the value of inertia weight for each iteration from Equation (7) and the value depending on the context factor according to Equation (6).
IV. EXPERIMENTAL ANALYSIS

To verify the performance of the NDPSO algorithm, six numerical benchmark test functions are selected for testing in this paper, and Table 1 shows the definitions, value ranges and globally optimal solutions of these six test functions. The population size of all experiments is 50, and each function is run independently 30 times with 2000 iterations.

A large number of experimental results showed that the best results were obtained when the maximum jump factor \( k_{\text{max}} = 1 \) and the minimum jump factor \( k_{\text{min}} = 0.1 \), so the maximum jump factor \( k_{\text{max}} = 1 \) and the minimum jump factor \( k_{\text{min}} = 0.1 \) were taken. The values of the individual cognitive constant \( c_1 = 2 \), the social cognitive constant \( c_2 = 2 \), the constant \( n = 0.5 \), \( \omega_{\text{max}} = 0.95 \), and \( \omega_{\text{min}} = 0.4 \) were set with reference to the literature [17]. \( \omega \) varies nonlinearly with the number of iterations between \([0.95, 0.4]\) and \( k \) varies nonlinearly with the number of iterations between \([0.1, 1]\). The default parameters in Ref. [3] are chosen for the basic PSO algorithm and those in Ref. [17] are chosen for the DPSO algorithm. The results for the GA, DE, and ABC algorithms come from the literature [18].

In Table 1, the Rosenbrock function is a complex unimodal function whose global minima lie within a smooth, narrow parabolic-shaped valley, and it is difficult for general algorithms to search for the global minima, so the Rosenbrock function is usually used to evaluate the performance of optimization algorithms. Functions \( f_2 \sim f_6 \) are typical nonlinear multimodal functions; they have a wide search space, many local minima and tall obstacles, which are usually considered to be very difficult to handle complex multimodal problems. Similarly, \( f_4 \sim f_6 \) functions are also commonly used to test the performance of the functions.

| No. | Function | Formulation |
|-----|----------|-------------|
| 1   | Rosenbrock | \( f_1 = \sum_{i=1}^{n} \left( 100(x_{i-1} - x_i^2)^2 + (x_i - 1)^2 \right) \) |
| 2   | Griewank  | \( f_2 = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1 \) |
| 3   | Rastrigin | \( f_3 = \sum_{i=1}^{n} \left( x_i^2 - 10 \cos (2 \pi x_i) + 10 \right) \) |
| 4   | Step      | \( f_4 = \sum_{i=1}^{n} \left( \left| x_i + 0.5 \right| \right)^2 \) |
| 5   | Ackley    | \( f_5 = -20 \exp \left( -0.2 \sum_{i=1}^{n} \frac{1}{\sqrt{n}} x_i^2 \right) \) |
| 6   | Zakharov  | \( f_6 = \sum_{i=1}^{n} x_i^2 + \left( \sum_{i=1}^{n} 0.5x_i \right)^2 + \left( \sum_{i=1}^{n} 0.5x_i \right)^4 \) |

A. PERFORMANCE COMPARISON WITH OTHER INTELLIGENT OPTIMIZATION ALGORITHMS

In this paper, the performance of the NDPSO algorithm is compared with four other intelligent optimization algorithms, which are GA, PSO, DE, and ABC algorithms, and the details are shown in Table 2.

| No. | Function | Algorithm | Mean   | Deviation |
|-----|----------|-----------|--------|-----------|
| 1   | Rosenbrock | DE        | 1.82E+01 | 5.04      |
| 2   | Griewank  | DE        | 1.48E-03 | 2.96E-03 |
| 3   | Rastrigin | DE        | 1.17E+01 | 2.54E+00 |
| 4   | Step      | DE        | 0       | 0         |
| 5   | Ackley    | DE        | 0       | 0         |
| 6   | Zakharov  | DE        | 0       | 0         |
From Table 2, we find that the NDPSO algorithm outperforms the GA, PSO, DE, and ABC algorithms in terms of the mean and variance when optimizing the Rosenbrock, Rastrigin, Step, and Zakharov functions and outperforms the others on the Griewank and Ackley functions, except for the ABC algorithm. In addition, the NDPSO algorithm sets more severe conditions for the optimization (e.g., the number of iterations is 50,000 for the GA, DE, and ABC algorithms, while the number of iterations is only 2,000 for the NDPSO algorithm). To verify the performance of the NDPSO algorithm on the Griewank and Ackley functions more comprehensively, the number of iterations optimized by the NDPSO algorithm on the Griewank and Ackley functions was set to the same iteration number of 50,000 as for the ABC algorithm, and other parameters were unchanged. The final result shows that the optimal value of the NDPSO algorithm is 0 for both the Griewank and Ackley functions, which shows that the NDPSO algorithm has faster convergence speed and better search accuracy.

B. PERFORMANCE TESTING OF HIGHER DIMENSIONAL FUNCTION ALGORITHMS

This section implements the above six functions $f_1 \sim f_6$ in 100 dimensions and NDPSO, DPSO and PSO algorithm performance test experiments were conducted, in addition to the number of dimensions of other parameters set above. Here, the $f_1 \sim f_6$ optimal values are 0, and the fitness is the adaptation value of each search. The algorithm search accuracy is higher when both the current optimal value and fitness are closer to 0. The specific situation is shown in Figures 1-6.
From Figures 1-6, we find that the optimal value search performance of the NDPSO algorithm is much better than the other two particle swarm algorithms when optimizing the above six 100-dimensional functions; specifically, the accuracy of the optimal value search is much higher than that of PSO and DPSO algorithms under a certain number of iterations, and the results obtained by the NDPSO algorithm are already very satisfactory when the number of iterations is small, which fully indicates that the NDPSO algorithm has higher search accuracy and faster convergence speed and verifies that the optimal value search performance of the DPSO algorithm is also better than that of the basic PSO algorithm in high dimensions.

The increase in the dimensionality of the function increases the complexity of the problem and has a large impact on the optimization capability. Therefore, to further investigate the performance of the nonlinear dissipative particle swarm algorithm (NDPSO) proposed in this paper, the mean and variance of the function are tested by using the three standard test functions \( f_1 \sim f_3 \) mentioned above when the dimensionality changes from 1 to 1000 dimensions (where mean is the mean value and deviation is the variance). The changes are shown in Figures 7-12.

From Figures 7-12, it is found that when the dimension changes from 1 to 1000, for the Rosenbrock function, the mean value changes from 0 to 0.1535 and the variance changes less than 0.09 using the proposed method, while the mean value changes from 0 to 1.1995 \times 10^{10} \) and the variance changes up to \( 10^{17} \) using the basic PSO algorithm. This fully shows that this algorithm not only has high search accuracy but is also very stable. Similarly, the results obtained from the Griewank function, Rastrigin function, Step function, Ackley function and Zakharov function also show these advantages of the present algorithm.
Figure 8 Variation of the Griewank function optimization results with dimensionality. (a) Mean and (b) deviation.

Figure 9 Rastrigin function optimization results with dimensionality. (a) Mean and (b) deviation.

Figure 10 Step function optimization results with dimensionality. (a) Mean and (b) deviation.
C. OPTIMIZATION OF ARTIFICIAL POTENTIAL FIELD PARAMETERS IN PATH PLANNING FOR MOBILE ROBOTS

The artificial potential field method proposed by Khatib is a planning method obtained by introducing the concept of field in physics into robotics. Its principle is to represent the environmental information as a superposition of the gravitational field of the target point and the repulsive field of the obstacle and to guide the robot to avoid the obstacle and move toward the target by the virtual force of the potential field on it, finally achieving the task of path planning. However, the artificial potential field method has several drawbacks [21,22]: (1) it easily falls into the local minimal problem; (2) it is prone to jitter problems in front of obstacles; and (3) the path cannot be found between adjacent obstacles. Given the above problems, this paper uses the unique advantages of the nonlinear dissipative particle swarm algorithm in optimization to select the potential field parameters for the artificial potential field method, which is used to solve the defects of the artificial potential field method, such as local minima and the unreachability problem (GNRON).

The robot path planning problem can be modeled as a constrained function optimization problem. Let the robot's motion region in the two-dimensional plane be \( A \) with \( n \) obstacles \( S_1 \cdots S_n \), \( s_i \in S, i = 1, \cdots n \) and \( S \subset A \). The robot's motion sequence in \( A \) is \( g_j \), whose coordinates are \((x_j, y_j), j = 0, \cdots k\). Then, each time the robot walks through the environment, the path is represented by the function:

\[
f(X) = \sum_{j=0}^{k} d_j
\]  

Here,

\[
d_j = \sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2}, \quad s.t. \quad (x_j, y_j) \notin S
\]  

Figure 11 Ackley function optimization results with dimensionality. (a) Mean and (b) deviation.

Figure 12 Zakharov function optimization results with dimensionality. (a) Mean and (b) deviation.
Let \( f(X) = \text{step} \times l \), \( \text{step} \) be the number of path steps and \( l \) be the step length. Then, the fitness function can be defined as

\[
\text{Fit} = \begin{cases} 
\text{step} \times l, & \text{getgoal} \\
0, & \text{missgoal}
\end{cases}
\] (13)

With the gravitational gain coefficient \( \alpha \) and repulsive gain coefficient \( \beta \), the maximum influence distance \( \rho_0 \) of the obstacle on the mobile robot that is improperly taken will lead to the combined force sum \( \overrightarrow{F} = 0 \) applied to the robot, at which time the mobile robot is caught in the potential field trap. At the same time, the selection of the forward progress length \( L \) also affects the path length and the degree of smoothing. Therefore, the parameters that need to be optimized by the nonlinear dissipative particle swarm algorithm are \( \alpha \), \( \beta \), \( \rho_0 \), and \( l \). The superiority index of the evaluation algorithm is that the optimized parameters enable the robot to take the shortest path from the starting point to the end point after bypassing the obstacle, and the fewer failed attempts to find the target point within a specified number of experiments, the better.

The simulation tests are conducted in an environment of a 12 x 12 unit area. The robot starting point (start) is located at (0, 0), and the goal point (goal) is located at (9, 7.5). To facilitate the planning process, the robot can be considered a mass point if the obstacle boundary is expanded outward by 1/2 of the maximum dimensions of the robot length and width, thus simplifying the description of the problem. Figure 10 shows the situation when the artificial potential field method falls into a local optimum and the robot cannot reach the target point. Figure 11 shows the path diagram obtained by the NDPSO algorithm, where the robot can avoid the potential field trap and reach the target point, thus solving the defects of the artificial potential field method, such as local minima and the unreachability problem (GNRON). Table 3 shows the results of the parameters obtained by optimization when the above path diagram is obtained.

Here, Min is the minimum path obtained in 2000 runs, Mean is the average path obtained in 2000 runs, Max is the maximum path obtained in 2000 runs, Deviation is the variance obtained in 2000 runs, and Missgoal is the number of failed attempts to find the target point in 2000 runs. From Table 4, it can be seen that the NDPSO algorithm proposed in this chapter has the smallest values of minimum path, mean path, maximum path and variance of path among all of the improved algorithms in path planning for mobile robots, and the number of failed searches for target points in 2000 runs is 39 for the NDPSO algorithm, 216 for the PSO algorithm and 176 for the DPSO algorithm.

V. CONCLUSION

The nonlinear dissipative particle swarm algorithm proposed in this paper is, in general, based on the following two improvements: 1) dissipating particles in a nonlinear
incremental manner, avoiding a large amount of unnecessary dissipation at the beginning of the iteration, while devoting more effort to dissipation at the end of the iteration, improving the operational efficiency and global search capability of the algorithm and 2) using a nonlinear decreasing strategy to adjust the inertia weights so that the algorithm improves its searchability when there is no dissipation. The experimental results show that the search capability of the NDPSO algorithm is significantly higher than that of the basic PSO and dissipative particle swarm algorithms, and it has advantages over other intelligent optimization algorithms. The performance of the algorithm is very stable, which is more suitable for high-dimensional complex function optimization problems and mobile robot path planning problems.

REFERENCES

[1] Kennedy and R. C. Eberhart, “Particle swarm optimization”, Proceedings of the IEEE International Conference on Neural Networks. Piscataway, 1995, pp. 1942–1948.

[2] J. D Farmer, N. H Packed, A. S Perelson, “The immune System, adaptation, and machine learning”, Physica, 1986, vol. 22, no. 2, pp. 187-204.

[3] Y. Shi and R. Eberhart, “Empirical study of particle swarm optimization”, International Conference on Evolutionary Computation. Washington, USA, 1999, pp. 1945-1950.

[4] Q. Lv, S. R. Liu and X. N. Qiu, “Design and realization of particle swarm optimization based on pheromone mechanism”, Acta Automatica Sinica. 2009, vol. 35, no. 11, pp. 1410–1419.

[5] L. L. Kang, W. Y. Dong, W. J. Song, J. Wan and K. S. Li, “Non-intereal opposition- based particle swarm optimization with adaptive elite mutation”, Journal on Communications. 2017, vol. 38, no. 8, pp. 66-78.

[6] B. Q. Lv, J. J. Zhang, Z. P. Li and T.Z.Liu, “Fuzzy partial swarm optimization based on filled function and transformation Function”, Acta Automatica Sinica. 2018, vol. 44, no. 1, pp. 74-85.

[7] L. F. Xu, Z. S. Huang, Z. S. Yang and W. L. Ding, “Mixed particle swarm optimization algorithm with multistage disturbances”, Journal of Software. 2019, vol. 30, no. 6, pp. 1835–1852.

[8] Y. Sun and Q. Z. Xia, “Analysis and improvement of semi-supervised K-means clustering based on particle swarm optimization algorithm”, Journal of Beijing University of Posts and Telecommunications. 2020, vol. 43, no. 5, pp. 21-26.

[9] H. X. Lu, S. Y. Yin, G. L. Gong, Y. Liu and G. Cheng, “A particle swarm optimization algorithm based on deep deterministic policy gradient”, Journal of University of Electronic Science and Technology of China. 2021, vol. 50, no. 2, pp. 199-206.

[10] W. Liu, Z. Wang, Y. Yuan, N. Zeng, K. Hone and X. Liu, “A Novel Sigmoid-Function-Based Adaptive Weighted Particle Swarm Optimizer”, in IEEE Transactions on Cybernetics, vol. 51, no. 2, pp. 1085–1093, Feb. 2021, doi: 10.1109/TCYB.2019.2945015.

[11] X. Xia et al., “Triple Archives Particle Swarm Optimization”, in IEEE Transactions on Cybernetics, vol. 50, no. 12, pp. 4862-4875, Dec. 2020, doi: 10.1109/TCYB.2019.2943928.

[12] Q. H. Xu, S. R. Liu and Q. Lv, “Discrete particle swarm optimization based on chaotic behavior of ant colony and its application”, Computer science. 2010, vol. 37, no. 5, pp. 178-180.

[13] Z. J. Li, Y. N. Zhang, L. J. Wang, X. Y. Jia and Y. W. Zhang, “Study of photovoltaic multi-mode maximum power point tracking based on improved quantum particle swarm optimization”, Acta Energiae Solaris Sinica. 2021, vol. 42, no. 5, pp. 221-229.

QINGHE XU, born in China in 1984, is a Ph.D student at Muban Chom Bueng Rajabhat University, Thailand. In 2010, he graduated from Hangzhou University of Electronic Science and Technology with a master's degree in Control Theory and Control Engineering. He is a senior manager of photovoltaic power generation and a lecturer at Wenzhou Business College (China), with research interests in new energy and artificial intelligence algorithms.

TUNHUA WU was born in Quanzhou City, Fujin Province, China in 1978. He received his Ph.D. from Xiamen University in 2008. He used to work in Zhejiang University of Technology and Wenzhou Medical University. Now, he is an associate professor at Wenzhou Business College, Wenzhou, Zhejiang, China. His research interests lie in the areas of digital image analysis, pattern recognition and computer graphics.

WEIWEI WANG, born in 1996 in China, graduated from the University of Hong Kong with a master's degree in science. She graduated from South China Agricultural University with a bachelor's degree in engineering in 2018. Now, she is a teaching assistant at Wenzhou Business College (China), with research interests in e-commerce and artificial intelligence.