Hindawi
Complexity
Volume 2020, Article ID 6693411, 10 pages
https://doi.org/10.1155/2020/6693411

Research Article

Smart City Landscape Design Based on Improved Particle Swarm Optimization Algorithm

Wenting Yao [1] and Yongjun Ding [2]

[1] *School of Art and Media, Xi'an Technological University, Xi'an, Shaanxi 710000, China*
[2] *School of Electronic and Information Engineering, Lanzhou City University, Lanzhou, Gansu 730030, China*

Correspondence should be addressed to Wenting Yao; yaowenting@xatu.edu.cn

Received 16 October 2020; Revised 13 November 2020; Accepted 18 November 2020; Published 1 December 2020

Academic Editor: Wei Wang

Aiming at the shortcomings of standard particle swarm optimization (PSO) algorithms that easily fall into local optimum, this paper proposes an optimization algorithm (LTQPSO) that improves quantum behavioral particle swarms. Aiming at the problem of premature convergence of the particle swarm algorithm, the evolution speed of individual particles and the population dispersion are used to dynamically adjust the inertia weights to make them adaptive and controllable, thereby avoiding premature convergence. At the same time, the natural selection method is introduced into the traditional position update formula to maintain the diversity of the population, strengthen the global search ability of the LTQPSO algorithm, and accelerate the convergence speed of the algorithm. The improved LTQPSO algorithm is applied to landscape trail path planning, and the research results prove the effectiveness and feasibility of the algorithm.

1. Introduction

On the premise of prioritizing environmental beautification, the optimal path design of landscape trails in the green belt should be people-oriented. Landscape trails are places for people to take a leisurely stroll. Therefore, establishing an optimal road model is essential for designing landscape trails that are convenient for pedestrians. People have carried out many aspects of research on landscape trail design and proposed some related optimization algorithms. Among them, the particle swarm optimization algorithm (PSO algorithm) was proposed by Russell Eberhart and James Kennedy in 1995. The algorithm is derived from the bird predation behavior of the group [1–3]. The PSO algorithm is simple to operate and has the characteristics of portability, easy implementation, fast convergence, and so forth and can obtain satisfactory solutions through self-adjustment, so it has received extensive attention from many scholars. At present, the theoretical research on particle swarm optimization algorithm is mainly to analyze the convergence of the algorithm. Due to the randomness of the particle swarm optimization algorithm, it is not very good to use some mathematical methods to verify the convergence effect of the algorithm. Therefore, it is particularly important to study the trajectory of particles in the algorithm, the distribution of particles in the group, and its mathematical theory. Rabanel et al. [4] studied the trajectory of a single particle in the population and related parameters in the algorithm. Xie et al. [5] studied the convergence of the particle swarm optimization algorithm and gave the parameter value range to improve the algorithm's search ability. Cui et al. [6] introduced a compression factor in the algorithm to control the particle trajectory and ensure the convergence of the algorithm. Qi et al. [7] analyzed the stability of the particle swarm optimization algorithm from the perspective of dynamics by using the Lyapunov stability and passive system theory in mathematics. Jatana and Suri [8] analyzed and verified the convergence of the algorithm from the interaction between particles. Yadav [9] theoretically analyzed the impact of speed on the convergence of the particle swarm optimization algorithm and gave a concrete proof. Chen and Li [10] used Markov's related theories to derive the necessary conditions for the convergence of the standard particle swarm optimization algorithm.

The research on the improvement of particle swarm optimization algorithm is mainly in the following aspects: the particle swarm optimization algorithm is improved from the perspective of inertia weight, learning factor, position and velocity update formula, and fusion with other algorithms [11–14]. In terms of inertia weight improvement, Li et al. [15] proposed a particle swarm optimization algorithm with linearly decreasing inertia weight, which improved the overall search ability of the algorithm. Nagra et al. [16] used random strategies to improve inertia weights. The improved algorithm not only improved the accuracy of the solution but also improved the solution speed of the algorithm. Elbaz et al. [17] used the cosine function to make nonlinear adjustments to the inertia weight, which improved the search efficiency of the algorithm and could effectively improve the premature phenomenon. In terms of learning factors, Wang et al. [18] proposed a particle swarm algorithm with variable acceleration factor, which improved the overall search ability of the algorithm. In terms of algorithm update formula improvement, Wei et al. [19] removed the speed formula in the algorithm and only the position update formula in the algorithm and proved the convergence of the improved algorithm; the algorithm became simpler and more efficient. Xu et al. [20] used the linear combination of the individual optimal position and the global optimal position to modify the individual optimal position and the global optimal position in the velocity formula, and the convergence speed of the algorithm was greatly improved. Luo and Gao [21] improved the location update formula, which effectively improved the search efficiency of the algorithm. In terms of fusion with other algorithms, Zheng et al. [22] added the selection mechanism in the genetic algorithm to the particle swarm optimization algorithm. The improved algorithm has an obvious optimization effect in solving the optimization problem of high-dimensional complex functions. Meshkati and Safi-Esfahani [23] merged the artificial bee colony algorithm with the particle algorithm optimization algorithm and successfully applied it to image segmentation. Che et al. [2] introduced the ant colony algorithm, and the improved particle swarm optimization algorithm has played a good optimization effect on solving the multiprocessor scheduling problem. Saffaran et al. [24] integrated the mechanism of the simulated annealing algorithm into the particle swarm optimization algorithm, thereby speeding up the algorithm's convergence speed and improving the algorithm's solution accuracy. Tang et al. [25] merged the differential evolution algorithm with the particle swarm algorithm to speed up the convergence speed of the algorithm. Wu et al. [26] designed an improved quantum evolution algorithm IPOQEA based on a niche coevolution strategy and enhanced particle swarm optimization (PSO). A method of boarding gate allocation based on IPOQEA is proposed, which allocates flights to appropriate boarding gates in different time periods. Finally, the actual operation data of Baiyun Airport is taken as an example to verify the effectiveness of the proposed method. In order to solve these problems, Wu et al. [27] designed an optimal mutation strategy based on the complementary advantages of five mutation strategies to develop a new improved DE algorithm WMSDE with wavelet basis functions, which can improve search quality and accelerate convergence to avoid falling into local optima and stagnation.

Although the above algorithm has achieved good results, the PSO algorithm is still easy to fall into the local optimum. This will not only not achieve better experimental results but also increase time consumption. Therefore, this paper proposes an optimization algorithm based on improved particle swarms. The contributions of this article are as follows:

(1) Aiming at the problem of premature convergence of the particle swarm algorithm, this paper uses the evolution speed of individual particles and the population dispersion to dynamically adjust the inertia weights to make them adaptive and controllable, thereby avoiding premature convergence.

(2) This paper introduces the natural selection method into the traditional location update formula to maintain the diversity of the population, strengthen the global search ability of the LTQPSO algorithm, and accelerate the convergence speed of the algorithm.

2. Particle Swarm Optimization Algorithm

2.1. Introduction to Particle Swarm Optimization Algorithm. The particle swarm optimization algorithm [28] is a mathematical simulation model of the process of birds looking for food. In particle swarm optimization (PSO), two simple equations of motion are designed to guide particles to find the global optimal solution in order to simulate the predator-prey flight behavior of birds, thus realizing the mathematical modelling of swarm behavior. At the same time, as an iterative algorithm based on population, the concept of PSO algorithm is simple and easy to implement. In solving real engineering optimization problems, PSO algorithm has been successfully applied in many fields [29].

In the basic particle swarm optimization algorithm, each particle is regarded as a potential solution of the problem to be optimized, and each particle has a fitness value determined by the optimization function. At the same time, each particle continuously iteratively updates its own speed and position, and the global optimal position finally found is the optimal solution to the problem to be optimized found by the algorithm.

2.2. Basic Principles of Particle Swarm Optimization Algorithm. Assuming that, in an n_1-dimensional target search space, there are n_2 particles in the population, the position of the i-th particle can be expressed as $A_i = (a_{i1}, \ldots, a_{in_1})$, and the velocity of the particle is $B_i = (b_{i1}, \ldots, b_{in_1})$, where $i = 1, \ldots, n_2$. The optimal position searched by the i-th particle itself so far is denoted as la, and the optimal position found by the entire particle swarm is denoted as lb. Equations (1) and (2) give the particle speed and position update formulas:

$$B_i^{p+1} = B_i^p + k_1 l_1 \left(la - A_i^p\right) + k_2 l_2 \left(lb - A_i^p\right), \tag{1}$$

$$A_i^{p+1} = A_i^p + B_i^{p+1}. \tag{2}$$

Among them, B_i^p in formula (1) represents the velocity of the i-th particle in the p-th iteration; A_i^p represents the position of the i-th particle in the p-th iteration; k_1 and k_2 are learning factors, and usually $c_1 = c_2 = 2$; l_1 and l_2 are two random numbers with values within [0.1]; p represents the current iteration number of the algorithm.

The inertia weight f is introduced and formula (1) is improved. The revised speed update formula is as follows:

$$B_i^{p+1} = fB_i^p + k_1 l_1 \left(la - A_i^p\right) + k_2 l_2 \left(lb - A_i^p\right). \tag{3}$$

The experimental results in [30] show that when the inertia weight value $f > 1.2$, the particles can be developed in a larger search space, thereby improving the accuracy of the algorithm; when the inertia weight value $f < 0.8$, the particle will quickly move closer to the global optimal solution, and the algorithm can perform a fine search in a local area. Then when the value range of the inertia weight is within [0.8, 1.2], the algorithm may obtain the global optimal value, and it also requires a suitable number of iterations. Reference [30] also shows that introducing the inertia weight of the linear decrement strategy in formula (3) can significantly improve the performance of the algorithm.

2.3. Basic Steps and Flowchart of Particle Swarm Optimization Algorithm. The basic steps of the particle swarm optimization algorithm are as follows:

Step 1: Set relevant parameters in the particle swarm algorithm

Step 2: Initialize the position and velocity of all particles in the population

Step 3: Calculate the fitness value of each particle in the particle swarm algorithm, and calculate the optimal position of the particle and the optimal position of the entire group at the same time

Step 4: Use formulas (2) and (3) to update the velocity and position of the particles

Step 5: If the maximum number of iterations is reached, the algorithm stops calculating and enters Step 6; otherwise, it returns to Step 3

Step 6: Output the optimal value

The flow chart of the particle swarm optimization algorithm is shown in Figure 1.

2.4. Problems of Particle Swarm Optimization Algorithm. Because the particle swarm optimization algorithm has the advantages of simple concept, few adjustment parameters, simple programming, and easy implementation, it has been successfully applied in many practical engineering optimization fields, but the particle swarm optimization algorithm itself still has the following problems [31, 32].

2.4.1. Improvement of Inertia Weight Strategy. The improvement strategy of inertia weight has always been a hot research topic of many scholars. The general improvement strategy is based on the following idea: in the early stage of algorithm search, the inertia weight can obtain a larger weight, and the algorithm's global search ability is strengthened. In the later stage of the algorithm search, the inertia weight can obtain a smaller weight, which is beneficial to the algorithm for local search. However, most of the improved inertia weighting strategies based on this idea lack rigorous theoretical proofs. Sometimes, the improved algorithm based on this strategy will decrease particle velocity in the late search period, leading to the tendency of particles in the population to the optimal solution position, the diversity of the population will be gradually lost, and the search efficiency of the algorithm will also be gradually weakened, resulting in the inability of the algorithm to jump out of the local optimum.

2.4.2. Parameter Setting. In the particle swarm optimization algorithm, when different parameters are selected with different values, the optimization results of the particle swarm optimization algorithm are also different; for different improved algorithms, there is no specific parameter setting standard to solve the problem. For particle swarm optimization algorithm in specific practical applications, the improved algorithm may not be able to achieve good optimization performance, and sometimes it is necessary to combine the problem itself to make corresponding improvements to the algorithm.

2.4.3. Algorithm Convergence Speed and Solution Accuracy. For most improved particle swarm optimization algorithms, when solving multimodal function problems, the algorithm may often fall into the local optimal value point. As a result, the particles may not be able to escape the local optimum in the subsequent search process and premature convergence appears. The accuracy of the solution is also difficult to improve. For most improved particle swarm algorithms, how to improve the convergence speed of the algorithm after reaching the specified accuracy is also one of the main research problems of the particle swarm algorithm.

3. Hybrid Improved Quantum Behavior Particle Swarm Optimization Algorithm

3.1. QPSO Algorithm. According to the basic convergence properties of particle swarms and inspired by the basic theories of quantum physics, Sun et al. proposed the Quantum Particle Swarm Optimization (QPSO) algorithm. This algorithm improves the search strategy of the entire PSO algorithm. Its evolution equation does not require a velocity vector and has a simple form, fewer parameters, and easier control. The QPSO algorithm is superior to all developed PSO algorithms in terms of search capabilities.

To transform the PSO algorithm, the equation expression obtained is as follows:

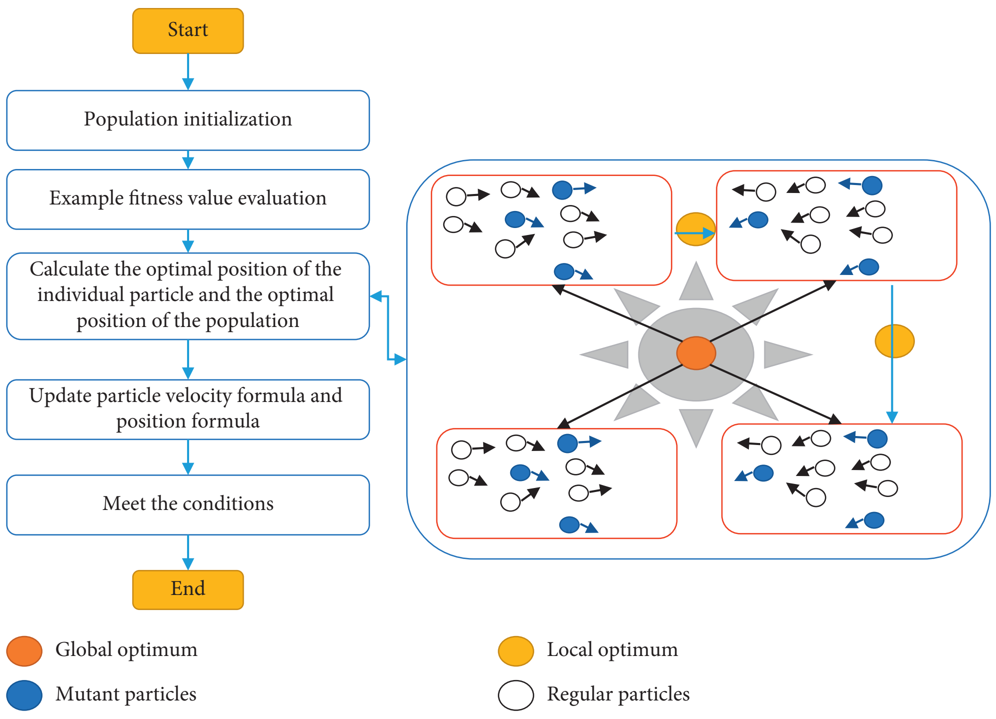
Start
Population initialization
Example fitness value evaluation
Calculate the optimal position of the individual particle and the optimal position of the population
Update particle velocity formula and position formula
Meet the conditions
End
Global optimum
Local optimum
Mutant particles
Regular particles

Figure 1: Flow chart of particle swarm optimization algorithm.

$$A_{ij}(p+1) = C_{ij}(p) + \lambda|D_j(p) - A_{ij}(p)|\ln\left(\frac{1}{y}\right), \tag{4}$$

$$C_{ij} = \xi la_{ij}(p) + (1-\xi) lb_j(p), \tag{5}$$

$$\begin{aligned} lm &= (D_1(p), \ldots, D_d(p)) \\ &= \left(\frac{1}{n}\sum_{i=1}^{n} la_{i1}(p), \ldots, \frac{1}{n}\sum_{i=1}^{n} la_{i1}(p)\right), \end{aligned} \tag{6}$$

$$\xi = \frac{k_1 l_1}{(k_1 l_1 + k_2 l_2)}. \tag{7}$$

Among them, the position of the i-th particle at time p is $A_i = (A_i, 1(p), \ldots, a_i, D(p))$, the best position of the individual is $la_i = (la_{i1}(p), \ldots, la_{id}(p))$, the best position of the group is $lb_i = (lb_{i1}(p), \ldots, lb_{id}(p))$, the average best position is $lm = (D_1(p), \ldots, D_d(p))$, the dimension and number of particles are d and n, respectively, λ is expansion-compression factor, k_1 and k_2 represent learning factors, and l_1 and l_2 represent uniformly distributed values.

The particle swarm algorithm of formula (5) is collectively called the quantum behavior particle swarm optimization algorithm.

3.2. The Principle of Hybrid Improved Quantum Behavior Particle Swarm Optimization Algorithm. In the evolutionary algorithm, reasonable algorithm control parameters have a greater impact on the performance of the algorithm. In order to improve the convergence of the QPSO algorithm, its evolutionary expressions (4) and (7) have been studied. According to the attraction point of the particle $C_{ij}(p+1)$, formula (5) can be transformed into

$$C_{ij}(p+1) = lb_{ij}(p) + \xi\big(lb_{ij}(p) - lb_{ij}(p)\big). \tag{8}$$

From equations (4) and (8), we can see that there is a certain relationship between the point of attraction $C_{ij}(p+1)$ of the updated particle and the best position $lb(p)$ of the global population and the best position $la(p)$ of the current particle. $A_{ij}(p+1)$ is also related to the difference between the current particle's average position lm and its own position $A(p)$. At present, many researchers have studied the influence of the average position lm of the entire group on the search behavior of the particle group during the search process. For the average position lm of the entire group, many methods for controlling parameters have been proposed. The inertial weight is an important part of the adjustable parameters of QPSO. When the inertial weight becomes larger, the global search performance of the algorithm can be improved, and the little the inertial weight can enhance the local search performance of the algorithm. However, the inertia weight value of the QPSO algorithm will decrease linearly as the evolutionary algebra becomes larger. If this method is used to describe the actual nonlinear and complex search process, the algorithm is more likely to fall into the local optimization extremum prematurely. As a result, the convergence speed of the algorithm also slows down.

In the selection of algorithm parameters, this paper uses the evolution speed of individual particles and the dispersion of the group to dynamically adjust the weight of inertia, so that the weight of inertia is adaptive, so as to avoid falling into the local optimum; at the same time, natural selection is introduced into the optimal position postprocessing. The advantage of this method is to maintain the diversity and stability of the population, strengthen the global search ability of the QPSO algorithm, and improve the convergence speed of the algorithm.

Assuming that $F(lb(p))$ and $F(C_i(p))$, respectively, represent the fitness value of the global optimal position and the fitness value of the current optimal position, the evolution speed of individual particles is defined as

$$g_i(p) = \frac{F(lb(p))}{F(C_i(p))}. \tag{9}$$

In the range of $0 < g \leq 1$, the smaller the value of g is, the faster the evolution speed will be. The particle position is consistent with its historical optimal position, and the current fitness value is compared with the changed particle fitness value. When the value of g remains at 1, it can be determined that the algorithm has obtained the optimal solution of the particle.

Suppose that the standard deviation of the best position of the particle in the dimension of the particle is $A(p) = (C_1(la_i, 1(p)), \ldots, C_d(la_i, d(p)))$, and, at the same time, the separation of the particle population in the evolution process is

$$H_i(p) = (H_{i1}(p), \ldots, H_{id}(p)) = \left(\frac{\partial_{D_1}(la_{i1}(p))}{\partial_{A_1}(A_{i1}(p))}, \ldots, \frac{\partial_{D_d}(la_{ic}(p))}{\partial_{A_d}(A_{ic}(p))}\right). \tag{10}$$

It can be seen from formula (10) that H can describe the degree of discretization of particles and the diversity of the population. If the value of H increases, the degree of discretization of particles increases, but the diversity of particle populations will decrease. When $H = 1$, the best position la is exactly the same as the current position A, but the value of H will continue to change.

We use standard functions to test the proposed inertia weights. The selected test functions are two nonlinear unimodal functions Sphere and two nonlinear multimodal raster functions. The number of test particles is 20, the maximum number of iterations is 100, and the particle dimension is 4. In the QPSO algorithm, $k_1 = 1$, $k_2 = 2.1$, and the expansion-contraction coefficient decreases from 1.0 to 0.5 in turn. In the maximum number of iterations, the floating range of individual particle evolution velocity is $0 \sim 1$, and the degree of population dispersion approximates 1 after constant oscillation, so it can be seen that the inertia weight can maintain the stability and diversity of the population. Using the above-mentioned transformation equation based on inertia weight to improve the QPSO algorithm, the equation for the improved QPSO algorithm is

$$C_{ij}(p+1) = lb_{ij}(p) + \xi g_i(p)\left(lb_{ij}(p) - lb_{ij}(p)\right), \tag{11}$$

$$A_{ij}(p+1) = C_{ij}(p) + \lambda(p)\left|lm - H_{jj}(p)A_{ij}(p)\right| \ln\left(\frac{1}{y}\right). \tag{12}$$

It can be seen from equations (11) and (12) that the velocity parameters of individual particles have a greater impact on the changes in the best position of the particles and the optimal position of the global population. In the process of adjusting the particle position of the population dispersion, these parameters are dynamic control parameters.

In order to improve the accuracy and stability of the algorithm in this paper, the natural selection algorithm is used to select the position of the particles. In the iterative process, the particles are arranged according to the function value from good to bad, and the state quantity of the best particle is selected according to the selection rule. The worst particle is finally sorted with the changed particle population. Equation (12) sorts the previous particle population and saves the historical optimal solution of each individual. Figure 2 shows the basic flow of the LTQPSO algorithm.

4. Results and Discussion

4.1. LTQPSO Algorithm Parameters. For the path planning of the particle swarm algorithm, repeated simulation experiments are usually required. In the evolutionary algorithm, the number of particles, the particle dimension, and the maximum number of iterations are three crucial parameters. When planning the optimal path, first determine the basic parameters and then consider the working environment and establish an estimation method for the relationship between the parameters to realize the connection path planning. In this paper, through a convergent and feasible parameter simulation experiment based on the LTQPSO algorithm in a barrier-free environment, a linear regression equation of the basic parameters is obtained.

In order to further analyze the relationship between the change of particle dimension and the number of iterations, this paper uses Matlab 2014a for simulation, the particle dimension is d $(5 \leq d \leq 20)$, the learning factors $k_1 = 2$ and $k_2 = 2.1$, and expansion-contraction coefficient is successively decreased from 1.0 to 0.5. Using the method where the values of k_1 and k_2 are fixed and λ decreases in sequence, the optimal path using polar coordinates and rectangular coordinates in a barrier-free environment is 14.14 m. Because the initial distribution has an effect on the convergence speed, the initial distribution is set to a uniform distribution and a normal distribution.

The specific method of the simulation experiment is as follows: according to the order of the particle dimensions, the number of particles and the number of iterations are sequentially increased to repeat the experiment, and the average value of the optimal solution is obtained by performing 100 experiments and recording the largest and smallest values when the standard deviation is less than 0.05. The feasible convergence interval can be represented by the maximum value and the minimum value. For the parameters of the algorithm in this paper, the regression analysis of the two initial distributions is performed, respectively. Figure 3 shows the maximum and minimum number of particles and

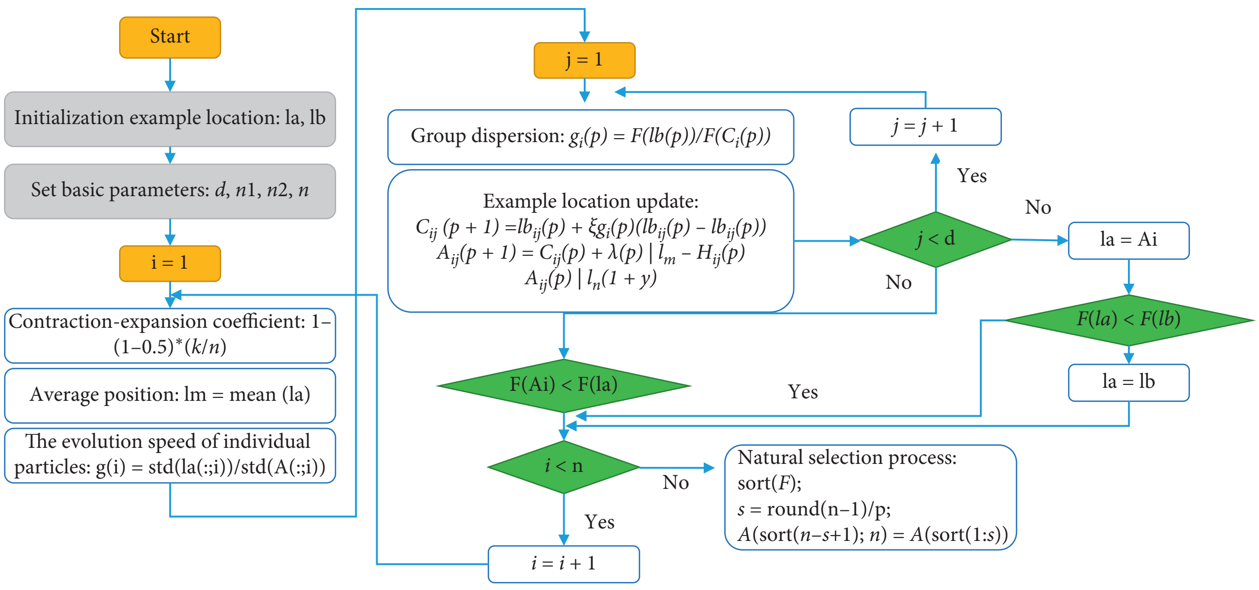
Start
Initialization example location: la, lb
Set basic parameters: d, n1, n2, n
i = 1
Contraction-expansion coefficient: 1−(1−0.5)*(k/n)
Average position: lm = mean (la)
The evolution speed of individual particles: g(i) = std(la(:;i))/std(A(:;i))
j = 1
Group dispersion: g_i(p) = F(lb(p))/F(C_i(p))
Example location update:
C_ij (p + 1) = lb_ij(p) + ξg_i(p)(lb_ij(p) − lb_ij(p))
A_ij(p + 1) = C_ij(p) + λ(p) | l_m − H_ij(p)
A_ij(p) | l_n(1 + y)
j = j + 1
Yes
j < d
No
la = Ai
No
F(la) < F(lb)
la = lb
F(Ai) < F(la)
Yes
i < n
No
Yes
Natural selection process:
sort(F);
s = round(n−1)/p;
A(sort(n−s+1); n) = A(sort(1:s))
i = i + 1

Figure 2: Flow chart of LTQPSO algorithm.

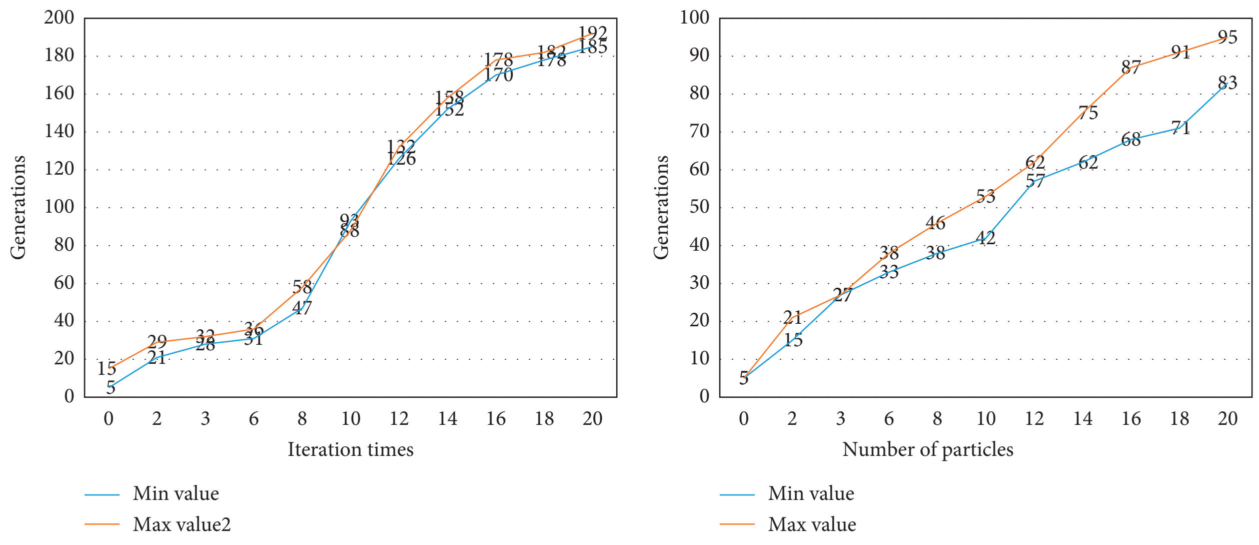

Figure 3: Simulation results of uniform distribution.

the number of iterations of 100 experimental results obtained by averaging the initial distribution of LTQPSO under the change of particle dimensions.

From Figure 3, we can see the change characteristics of the number of particles and the number of iterations. The change of its parameters is similar to that of the exponential function, so a one-variable linear regression equation can be used to express the basic parameters of path planning. Table 1 lists the regression analysis results of the basic parameters obtained by using the Matlab regression analysis “Regress” command.

From the experimental results of the parameter regression analysis in Table 1, it can be seen that the regression variation difference of the maximum value of the basic parameters of the LTQPSO algorithm is better than the minimum value. The unary linear regression equation of the basic parameters is

$$S = \text{round}\left(0.4123L^{1.7091}\right). \tag{13}$$

The number of particles that converged in the average initial distribution and the number of iterations can be calculated by formula (13) according to the change of the number of particles in the LTQPSO algorithm.

According to the above method, using the LTQPSO algorithm to perform 100 experiments on the particle dimensions under a normal distribution, the maximum number of particles, the minimum number of particles, and the number of iterations obtained are shown in Figure 4.

The change characteristics of the number of particles and the number of iterations can be seen in Figure 4. The change of parameters is similar to that of the exponential function, so a linear regression equation can be used to express the basic parameters of path planning.

Table 1: Parametric regression analysis results for uniform distribution.

Parameter		Min	Max
Regression coefficients	b_0	−1.4432	−0.8512
	b_1	1.8701	1.7805
Coefficient area	b_0	(−1.71, −1.13)	(−1.06, −0.58)
	b_1	(1.78, 1.87)	(1.52, 1.79)
Statistics	R^2	1	1
	F	1478.1	1385.9
	P	0	0
	α	0	0

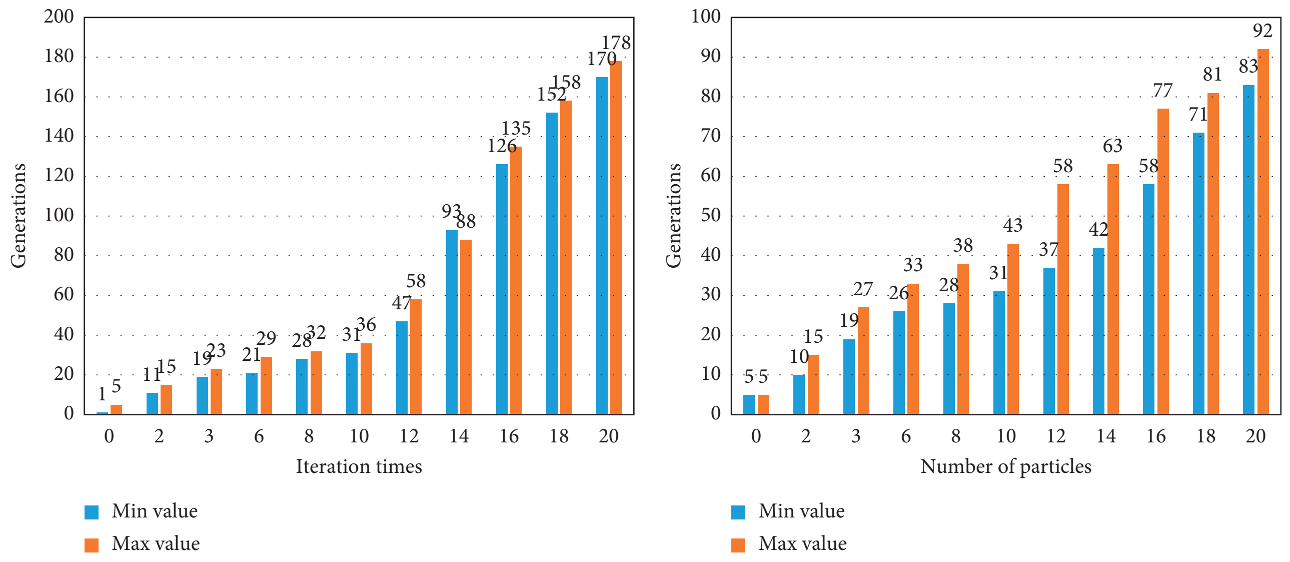

Figure 4: Simulation results of normal distribution.

Table 2 lists the regression analysis results of the basic parameters obtained by using the Matlab regression analysis "Regress" command. According to the experimental results of parameter regression analysis in Table 2, the regression variation difference of the maximum value of the basic parameters of the LTQPSO algorithm is better than the minimum value. The unary linear regression equation of the basic parameters is

$$S = \mathrm{round}\left(0.3107L^{1.0865}\right). \tag{14}$$

In summary, in the path planning of mobile robots based on the LQPSO algorithm proposed in this paper, the LTQPSO algorithm can use equations (13) and (14) to determine the basic parameters, so it can be used for path planning of landscape trails.

4.2. Comparison Algorithm. Using the adaptive BP algorithm with momentum term, basic particle swarm optimization (BPSO) algorithm and improved particle swarm optimization (LTQPSO) algorithm's error curve is shown in Figure 5.

It can be seen from Figure 5 that the improved particle swarm optimization algorithm achieves the given error accuracy at about 260 steps, while the basic particle swarm optimization algorithm and the adaptive BP algorithm with momentum term use about 880 steps and 1,900 steps, respectively. By comparing the optimization error curves of different algorithms, the results show that the improved PSO has a much faster convergence rate, and the proposed algorithm has a smaller error.

Using improved particle swarm optimization to approximate the given function, the curve obtained is shown in Figure 6. Linear regression analysis is performed on the optimized simulation output results and target results, and the analysis results are shown in Figure 7. It can be seen from Figure 7 that the correlation coefficient between the optimized output result and the target result reaches 0.999.

In order to verify the effectiveness of the proposed improved particle swarm algorithm, the above test function was optimized with 200 steps, and the test was repeated 30 times. The test results of the network mean square error obtained by the optimization of the three algorithms are shown in Table 3.

The test results in Table 3 show that the optimal MSE and the worst MSE of the proposed algorithm are the lowest, and the variance is also the smallest, which shows that the algorithm proposed in this paper has strong stability. It can be

Table 2: Parameter regression analysis results of normal distribution.

Parameter		Min	Max
Regression coefficients	b_0	−1.7086	−1.1562
	b_1	1.9906	1.8865
Coefficient area	b_0	(−2.11, −1.33)	(−1.76, −0.98)
	b_1	(1.98, 2.17)	(1.72, 1.99)
Statistics	R^2	0.988	1
	F	778.1	1125.9
	P	0	0
	α	0.011	0

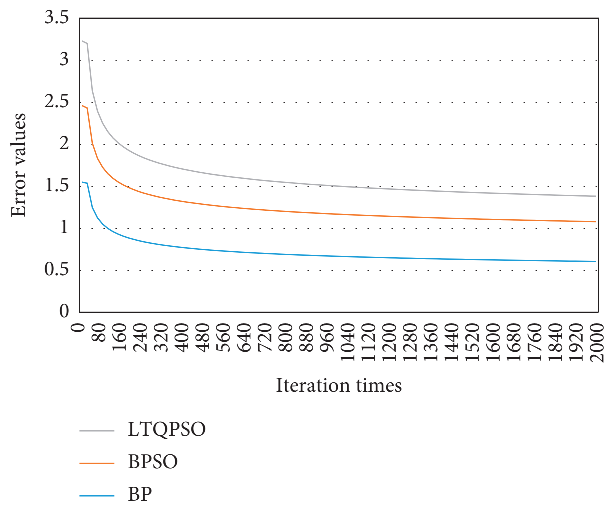

Figure 5: Error curve of particle swarm optimization algorithm.

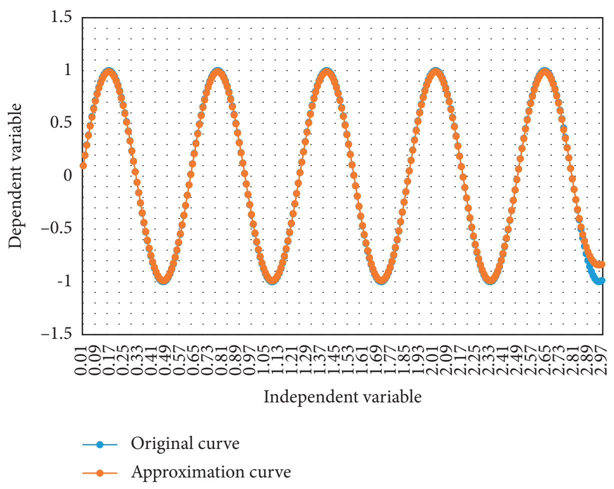

Figure 6: Optimized function approximation effect diagram.

known from the experimental results that the improved particle swarm optimization algorithm effectively improves the optimization accuracy and efficiency compared with the BP algorithm and the basic particle swarm algorithm.

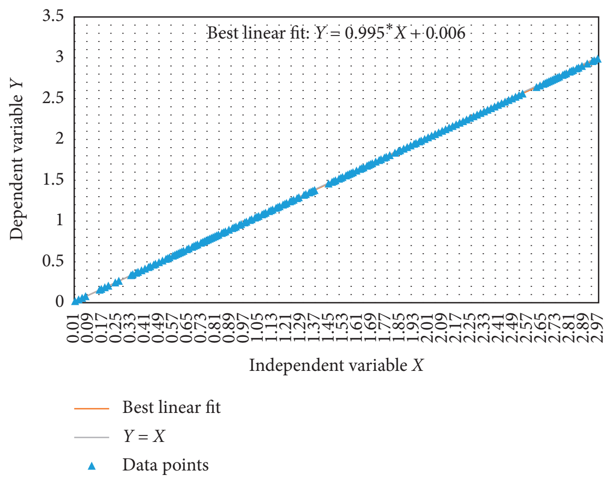

Figure 7: Regression analysis results after optimization.

Table 3: Comparison of the mean square error results of the three algorithms on the test function.

Methods	BP	BPSO	LTQPSO
Optimal MSE	0.0018	0.0037	0.0009
Worst MSE	0.0321	0.0978	0.0015
Mean	0.0187	0.0265	0.0018
Variance	0.0012	0.0024	0.0006

4.3. Comparison of Algorithm Running Time and Number of Iterations. The traditional design method consumes huge computing resources and wastes time. The design of the optimization algorithm in this paper solves this problem well. The genetic algorithm has the best effect at present. The biggest feature of particle swarm optimization algorithm is that it can save time and computing resources more than genetic algorithm. The comparison result is shown in Figure 8.

It can be seen from Figure 8 that the algorithm in this paper can complete convergence in about 500 generations, while the genetic algorithm needs 1100 generations, and other algorithms need more than that. At the same time, comparing the running times of the algorithms, the running time of the genetic algorithm is 369.43 s, and the particle swarm optimization algorithm only takes 146.97 s. It is shown that the particle swarm optimization algorithm is faster than the genetic algorithm in design. Combined with the performance comparison in Table 3, it can be seen that the particle swarm optimization algorithm has improved the design efficiency of landscape trail path planning.

Table 4 shows the running time results of different algorithms. The computational complexity of the algorithm is proportional to the running time. It can be seen from Table 4 that the complexity of the algorithm in this paper is the lowest, and the corresponding running time is also the lowest.

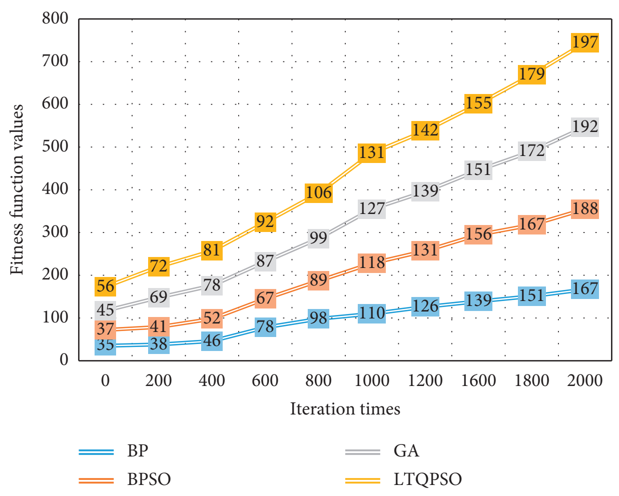

Figure 8: Comparison of fitness function values and iteration times of the four algorithms.

Table 4: Comparison of the running time results of the four algorithms.

Methods	BP (s)	BPSO (s)	GA (s)	LTQPSO (s)
Time	0.115	0.137	1.002	0.019

5. Conclusion

The landscape trail optimization model designed by the improved particle swarm algorithm is feasible and effective and can reduce the construction cost of landscape trails. This paper proposes an optimization algorithm for improving quantum behavioral particle swarms (LTQPSO). Aiming at the problem of premature convergence of the particle swarm algorithm, the evolution speed of individual particles and the population dispersion are used to dynamically adjust the inertia weights to make them adaptive and controllable, thereby avoiding premature convergence. At the same time, the natural selection method is introduced into the traditional position update formula to maintain the diversity of the population, strengthen the global search ability of the LTQPSO algorithm, and accelerate the convergence speed of the algorithm. The improved LTQPSO algorithm is applied to landscape trail path planning. The research results prove that the algorithm is easier to guide and has a more efficient global search capability, showing higher efficiency and robustness.

Data Availability

The data used to support the findings of this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare that they have no known conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Z. Cui, J. Zhang, D. Wu et al., "Hybrid many-objective particle swarm optimization algorithm for green coal production problem," *Information Sciences*, vol. 518, pp. 256–271, 2020.

[2] G. Che, L. Liu, and Z. Yu, "An improved ant colony optimization algorithm based on particle swarm optimization algorithm for path planning of autonomous underwater," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 8, pp. 3349–3354, 2020.

[3] X. Zhang, X. Wang, Q. Kang, and J. Cheng, "Differential mutation and novel social learning particle swarm optimization algorithm," *Information Sciences*, vol. 480, pp. 109–129, 2019.

[4] J.-M. Rabanel, V. Adibnia, S. F. Tehrani et al., "Nanoparticle heterogeneity: an emerging structural parameter influencing particle fate in biological media?" *Nanoscale*, vol. 11, no. 2, pp. 383–406, 2019.

[5] Y. Xie, Y. Zhu, Y. Wang et al., "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud-edge environment," *Future Generation Computer Systems*, vol. 97, pp. 361–378, 2019.

[6] J. Cui, Q. Jiang, S. Li, X. Feng, Y. Zhang, and Y.-E. Shi, "Numerical study of anisotropic weakening mechanism and degree of non-persistent open joint set on rock strength with particle flow code," *KSCE Journal of Civil Engineering*, vol. 24, no. 3, pp. 988–1009, 2020.

[7] Z. Qi, Q. Shi, and H. Zhang, "Tuning of digital PID controllers using particle swarm optimization algorithm for a CAN-based DC motor subject to stochastic delays," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 7, pp. 5637–5646, 2019.

[8] N. Jatana and B. Suri, "Particle swarm and genetic algorithm applied to mutation testing for test data generation: a comparative evaluation," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 4, pp. 514–521, 2020.

[9] A. Yadav, "AEFA: artificial electric field algorithm for global optimization," *Swarm and Evolutionary Computation*, vol. 48, pp. 93–108, 2019.

[10] G. Chen and S. Li, "Markov and improved particle swarm optimization-based privacy preservation algorithm for user space geographical location," *European Journal of Remote Sensing*, vol. 53, no. 1, pp. 31–40, 2020.

[11] H. Zhengtong, G. Zhengqi, M. Xiaokui, and C. Wanglin, "Multimaterial layout optimization of truss structures via an improved particle swarm optimization algorithm," *Computers & Structures*, vol. 222, pp. 10–24, 2019.

[12] W. Ji, G. Chen, B. Xu, X. Meng, and D. Zhao, "Recognition method of green pepper in greenhouse based on least-squares support vector machine optimized by the improved particle swarm optimization," *IEEE Access*, vol. 7, pp. 119742–119754, 2019.

[13] Z. Ouyang, Y. Liu, S.-J. Ruan, and T. Jiang, "An improved particle swarm optimization algorithm for reliability-redundancy allocation problem with mixed redundancy strategy and heterogeneous components," *Reliability Engineering & System Safety*, vol. 181, pp. 62–74, 2019.

[14] Q.-B. Zhang, P. Wang, and Z.-H. Chen, "An improved particle filter for mobile robot localization based on particle swarm optimization," *Expert Systems with Applications*, vol. 135, pp. 181–193, 2019.

[15] M. Li, H. Chen, X. Wang, N. Zhong, and S. Lu, "An improved particle swarm optimization algorithm with adaptive inertia

weights," *International Journal of Information Technology & Decision Making*, vol. 18, no. 3, pp. 833–866, 2019.

[16] A. A. Nagra, F. Han, and Q. H. Ling, "An improved hybrid self-inertia weight adaptive particle swarm optimization algorithm with local search," *Engineering Optimization*, vol. 51, no. 7, pp. 1115–1132, 2019.

[17] K. Elbaz, S.-L. Shen, W.-J. Sun, Z.-Y. Yin, and A. Zhou, "Prediction model of shield performance during tunneling via incorporating improved particle swarm optimization into ANFIS," *IEEE Access*, vol. 8, pp. 39659–39671, 2020.

[18] Q. Wang, S. Chen, and X. Luo, "An adaptive latent factor model via particle swarm optimization," *Neurocomputing*, vol. 369, pp. 176–184, 2019.

[19] L. Wei, X. Li, and R. Fan, "A new multi-objective particle swarm optimisation algorithm based on R2 indicator selection mechanism," *International Journal of Systems Science*, vol. 50, no. 10, pp. 1920–1932, 2019.

[20] G. Xu, Q. Cui, X. Shi et al., "Particle swarm optimization based on dimensional learning strategy," *Swarm and Evolutionary Computation*, vol. 45, pp. 33–51, 2019.

[21] J. Luo and Y. Gao, "Cooperative particle swarm optimization algorithm with cloud mutation operator based on normal cloud model," *International Journal of Machine Learning and Computing*, vol. 9, no. 5, pp. 554–560, 2019.

[22] T. Zheng, Y. Liang, B. Wang et al., "A two-stage improved genetic algorithm-particle swarm optimization algorithm for optimizing the pressurization scheme of coal bed methane gathering networks," *Journal of Cleaner Production*, vol. 229, pp. 941–955, 2019.

[23] J. Meshkati and F. Safi-Esfahani, "Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing," *The Journal of Supercomputing*, vol. 75, no. 5, pp. 2455–2496, 2019.

[24] A. Saffaran, M. A. Moghaddam, and F. Kolahan, "Optimization of backpropagation neural network-based models in EDM process using particle swarm optimization and simulated annealing algorithms," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 1, pp. 1–14, 2020.

[25] B. Tang, K. Xiang, and M. Pang, "An integrated particle swarm optimization approach hybridizing a new self-adaptive particle swarm optimization with a modified differential evolution," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4849–4883, 2020.

[26] D. Wu, J. Xu, and H. Zhao, "A novel gate resource allocation method using improved PSO-based QEA," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2020.

[27] D. Wu, J. Xu, and Y. Song, "Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem," *Applied Soft Computing*, vol. 2020, Article ID 106724, 2020.

[28] K. Gholami and E. Dehnavi, "A modified particle swarm optimization algorithm for scheduling renewable generation in a micro-grid under load uncertainty," *Applied Soft Computing*, vol. 78, pp. 496–514, 2019.

[29] B.-h. Zhou, X.-m. Liao, and K. Wang, "Kalman filter and multi-stage learning-based hybrid differential evolution algorithm with particle swarm for a two-stage flow shops scheduling problem," *Soft Computing*, vol. 23, no. 24, pp. 13067–13083, 2019.

[30] T. Khurshaid, A. Wadood, S. G. Farkoush, C.-H. Kim, N. Cho, and S.-B. Rhee, "Modified particle swarm optimizer as optimization of time dial settings for coordination of directional overcurrent relay," *Journal of Electrical Engineering & Technology*, vol. 14, no. 1, pp. 55–68, 2019.

[31] İ. B. Aydilek, "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems," *Applied Soft Computing*, vol. 66, pp. 232–249, 2018.

[32] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.