A Review of Particle Swarm Optimization Algorithms

Ruyun Wang[1]*

Abstract

In this paper, the principle, process, social behavior analysis and algorithm components of Particle Swarm Optimization (PSO) are described in detail, and the development, research content, characteristics and application of PSO and its improved algorithm are reviewed. As an important optimization tool, PSO algorithm has been successfully used in the fields of objective function optimization, neural network training, fuzzy control system, pattern recognition, signal processing, image processing, statistical learning model parameter optimization, machine learning model parameter optimization, etc. This article also introduces several main algorithm to improve the direction, including improved method based on parameter adjustment, with contraction factor improvement methods, for discrete optimization problems of two typical version of particle swarm optimization algorithm, based on the genetic improvement strategy of ideas and gradient information, and the algorithm in constrained optimization and multi-objective optimization solutions of two classes of complex environment. Through eight classical test functions of the simulation, the standard PSO algorithm and its various improvement compared to the statistical performance of different algorithms in different function, it is concluded that the particle swarm algorithm in the parameter improvement, mixing, improvement and learning operator topology structure improvement aspects of their respective characteristics, the algorithm of improved play a guiding role in the future.

Keywords

Particle Swarm Optimization algorithm computational intelligence evolutionary algorithm

[1]*School of Information Science and Engineering, Shandong Normal University, Jinan, China*

***Corresponding author**: ruyunw@163.com

All involved algorithms' code and experiment results is avaliable at `https://github.com/RuYunW/Review_of_PSO`

1. Introduction

Particle Swarm Optimization (PSO) belongs to the field of computational intelligence. It is a global search algorithm proposed by Kennedy, PhD, social psychology, and Eberhart, PhD, electrical engineering, in 1995[1][2]. The algorithm is derived from the simulation of migration and social behavior of birds in the process of foraging. It is a random search algorithm that simulates biological activities in nature and swarm intelligence. The position of each particle in the population represents a candidate solution in the search space. Individual particles adjust their search direction through the communication and learning of location information. Particle swarm optimization because of its algorithm is simple, easy to implement, without gradient information, less parameters etc in continuous optimization problems, and discrete optimization problems are showed good result, especially because of its natural characteristics of real number encoding is suitable for processing the optimization problem, in recent years become a popular in research of intelligent optimization in the world. As an important optimization tool, particle swarm optimization (PSO) algorithm has been successfully used in the fields of objective function optimization, neural network training, fuzzy control system, pattern recognition, signal processing, image processing, statistical learning model parameter optimization, machine learning model parameter optimization, etc.

1.1 The Development of PSO

PSO simulates the predation behavior of birds in random search for food. Assuming that there is only one piece of food in the search area, all the birds do not know where the food is. Therefore, Kenndy et al. believe that there is information exchange between birds, and the distance from the food can be obtained by estimating their own fitness value. So groups of birds working together to find the best solution, searching the area around the bird closest to the food, is an effective method. The PSO algorithm imagines each potential solution as a bird in the search space, called a "particle. "The particle is important to follow the current optimal particle to search in the solution space.

PSO first initializes a group of random particles as the initial population, and then searches for the optimal solution through iterative optimization. In each iteration, the particle updates itself by tracking two "extreme values". The first is the optimal solution that the particle itself finds in successive iterations, which is called individual extreme value pBest; the other extreme value is the optimal solution that the whole population has found since the beginning of iteration, which is called global extreme value gBest. These two optimal variables guide the motion of the particles. In addition, global extremum can also be replaced by local extremum, that is, some particles form a neighborhood and share information with each other to guide the motion of particles in the neighborhood. The particle always follows the two extreme values and updates its own speed and position in each iteration until it finds the optimal solution.

Like other optimization algorithms, particle swarm optimization has the problem of premature convergence. Premature Convergence, also known as "Premature Convergence", means that the current algorithm falls into a non-global optimal state after iteration to a certain extent in the process of search evolution. At the same time,

particle swarm optimization (PSO) can not prove theoretically that it converges to the global maximum point of any function. Vanden Bergh proved that although the traditional standard PSO algorithm is efficient, it does not guarantee that it converges to the global optimal point, nor does it guarantee that the global optimal point cannot be searched, but the global optimal point can be searched with a certain probability. Up to now, the development of particle swarm optimization (PSO) has attracted more and more attention and research from many fields and scholars. In view of the problems existing in the PSO algorithm, domestic and foreign scholars mainly improve the algorithm from the following four aspects: adjustment of parameters, change of population topology structure, algorithm fusion and niche technology.

At present, the development trend of particle swarm optimization is as follows:

- Improvement of particle swarm optimization algorithm. Particle swarm optimization algorithm is widely used in solving continuous problems and single-objective problems. How to apply it to discrete problems and multi-objective problems is an important research direction of PSO algorithm. How to combine with other evolutionary algorithms to give full play to the advantages and improve the shortcomings of PSO algorithm is also worth studying.

- Theoretical analysis of particle swarm optimization algorithm. The PSO algorithm has not been proposed for a long time, and its mathematical foundation is weak. The research on the operation behavior, convergence and computational complexity of the algorithm is insufficient. How to guide the selection and design of parameters, how to design adaptive value functions, how to improve the efficiency of the algorithm and ensure convergence are all the research directions of PSO algorithm.

- The biological basis of particle swarm optimization. How to perfect the algorithm according to the group behavior and infuse the group intelligence into the algorithm is also one of the research directions.

- The comparison between particle swarm optimization algorithm and other evolutionary algorithms. With the integration of other evolutionary algorithms, how to combine the advantages of other evolutionary algorithms with particle swarm optimization and construct a hybrid algorithm with characteristics and use value is an important direction of the current algorithm improvement.

- The application of particle swarm optimization algorithm. The effectiveness of the algorithm must be demonstrated in the application. It is significant to expand the application field of particle swarm optimization (PSO), and to further study PSO.

1.2 Research Contents

Particle swarm optimization is a very simple algorithm, which can effectively optimize various functions. To some extent, this algorithm is somewhere between genetic algorithm and evolutionary programming. PSO algorithm is very dependent on random process, which is similar to evolutionary programming. The adjustment of the approach to global and local optimality in the algorithm is similar to the crossover operator in the genetic algorithm.

The main research contents of particle swarm optimization are as follows:

- Look for global best.

- Has a higher convergence speed.

The algorithm also USES the concept of adaptive values, a feature common to all evolutionary algorithms.

1.3 Characteristic

Particle swarm optimization (PSO) is essentially a random search algorithm, which is a new intelligent optimization technology. It is a branch of swarm intelligence and also a simulation of simple social system. Compared with traditional optimization algorithms, this algorithm has faster speed and better global search capability. Its features are as follows:

- Particle swarm optimization (PSO) is an optimization algorithm based on swarm intelligence theory. Compared with evolutionary algorithm, PSO is a more efficient parallel search algorithm.

- PSO and GA[3] has many in common, Both of them randomly initialize the population, use the fitness value to evaluate the individual's degree of fitness, and then conduct random search. However, PSO adopts a speed-displacement model, which is simple to operate, avoids complex genetic operations, and retains the global search strategy based on population.

- Each particle still retains individual extremum at the end of the algorithm. Therefore, if PSO is used for scheduling and decision-making problems, a variety of meaningful choices can be given.

- PSO's unique memory allows it to dynamically track current searches and adjust its search strategy.

- The characteristics of PSO algorithm make it maintain diversity and directivity in the search process.

- In the case of convergence, the particles tend to be the same at the end of iteration, and the convergence speed is obviously slow, so the optimization cannot be continued when the convergence reaches a certain accuracy.

- PSO algorithm is not sensitive to the population size, and even if the population number drops, it has little impact on the performance of the algorithm.

1.4 Application

Particle swarm optimization (PSO) provides a general framework for solving optimization problems. The meaning of its application is: in a given condition to find the best solution, make the resources play the maximum utility. Its important application areas are as follows:

- Engineering design and optimization[4]: neural network training[5], RBF network optimization[6], Controller design and optimization[7], Circuit and filter design optimization.
- Power system field[8]: Optimal power flow calculation[9][10], Grid expansion plan[11], Power recovery system[12], repair schedule[13], Unit optimization combination[14], Capacitor optimization[15].
- Transportation field. Vehicle path optimization problem[16], Logistics VRP distribution problem[17], Traffic control optimization problem[18], Robot path planning problem[19][20][21].
- Computer realm: Data mining classification problem[22][23], picture processing[24][25], task allocation[26].
- Information communication field: Routing and mobile communication base station layout optimization[27][28], Antenna array control optimization[29][30], Optimization of PMD control[31].

1.5 Research Contents

This paper mainly explains the generation, research status and algorithm principle of traditional particle swarm optimization (PSO). Meanwhile, it lists improved versions of various particle swarm optimization (PSO) algorithms for interpretation, analysis and comparison.

2. Principle of PSO

Particle swarm optimization (PSO) is a stochastic evolution method based on swarm intelligence. The algorithm explores the problem solution space by simulating the predation behavior of birds in nature, and moves the position according to the Fitness Value of individuals to the environment. The birds eventually find out where the food is through exploration and group cooperation. Each bird continuously records and updates the closest distance it has ever been to the food during its flight. At the same time, they compare the best location found by others through information communication to get the best location found by the whole group. In this way, each bird has a guiding direction when it flies. They will adjust their flying speed and position based on their own experience and the experience of the whole group, constantly looking for the position closer to the food, so that the group finally gathers to the food position.

2.1 Elementary Particle Swarm Optimization Algorithm

PSO algorithm originated from the simulation of simple social system, has a good biological social background, and has been widely concerned in scientific research and engineering practice. It is easy to understand, easy to implement, and has strong global searching ability for linear and multi-peak problems, which makes it a good optimization tool.

2.2 Algorithm Principle

Elementary PSO [32] is the initial version of PSO[32]. A Swarm composed of m particles flies at a certain speed in the d-dimensional search space. When searching, each Particle takes into account the best historical points it has searched and the best historical points of other particles in the group (or neighborhood) and changes its position on this basis.

The position of the ith particle is expressed as:

$$x_i = [x_i^1, x_i^2, x_i^3, ..., x_i^D]$$

The velocity of the ith particle is expressed as:

$$v_i = (v_i^1, v_i^2, v_i^3, ..., v_i^D), 1 \leq i \leq m$$

The best point in the history of the ith particle is denoted as: p_{Best}

The best point passed by all particles in the population (or neighborhood) is represented as gBest, and this global optimal vector ACTS as a guide for particles to converge to the global optimal region.

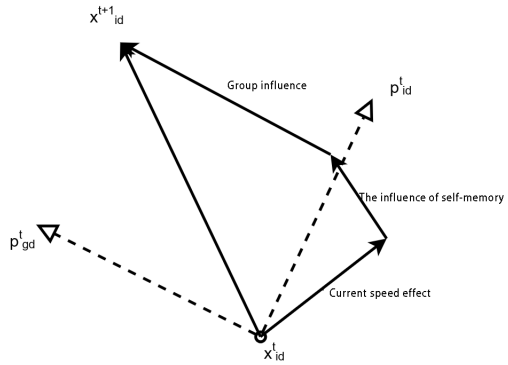


Figure 1. The update mode of particle position in each generation.

In general, the position and velocity of a particle are evaluated in a continuous real space.

2.2.1 Algorithm Flow

The basic particle swarm optimization algorithm flow is as follows:

1. Initialize all particles, randomly initialize the position and velocity of particles, and set the historical optimal pBest of the individual as the current position, and the optimal individual in the group as the current gBest.
2. Calculate the fitness function value of each particle.
3. For each particle, its historical optimal fitness value is compared with the fitness value of the best position experienced in the group. If it is better, it is regarded as the current global optimal position.

4. For each particle, its historical optimal adaptive value is compared with the adaptive value of the best position it has experienced. If the result is better, it is regarded as the current global optimal position.

5. Refrash each particle i's dimension d's speed and position according to equation (1) and equation (2).

$$v_i^d = v_i^d + c_1 * rand_1^d * (pBest_i^d - x_i^d) + c_2 * rand_2^d * (gBest^d - x_i^d) \quad (1)$$

$$x_i^d = x_i^d + v_i^d \quad (2)$$

6. If the termination condition is not met, proceed to step 2.In general, the termination condition is set as a good enough adaptive value or reaches a preset maximum iteration algebra.

Algorithm 1: Elementary Particle Swarm Optimization

Input : Number of iterations T; Population size NP; Problem dimension D

Output : Globally optimal position vector$\boldsymbol{x^*(t)}$

$t \leftarrow 1$(initialization);
Initializes the position vector of the particle $x_i^d = x_{min} + rand() * (x_{max} - x_{min})$;
Initializes the velocity vector of the particle $v_i^d = v_{min} + rand() * (v_{max} - v_{min})$;
while $(|f(x(t)^* \geq \varepsilon)|)or(t \leq T)$ **do**
  for *i=1 to NP* **do**
    $f(x_i(t))$ ←Calculate the fitness value of the particle$x_i(t)$;
    Update the local optimal value of particle i $pbest_i(t)$;
    Update the global optimal position of the population$gbest(t)$;
  end
  for *i=1 to NP* **do**
    for *j=1 to D* **do**
      Update the velocity vector of the particle$V_i(t)$and position vector$X_i(t)$;
      $v_i^j \leftarrow v_i^j(t) + c_1 r_1 * (P_i^j(t) - x_i^j(t)) + c_2 r_2 * (G^j(t) - x_i^j(t))$;
      $x_i^j \leftarrow x_i^j(t) + v_i^j(t)$;
    end
  end
  t←t+1;
end
return $\boldsymbol{x^*(t)}$;

Among them, c1 and c2 are called the Learning Factor or Acceleration Coefficient. The Learning Factor enables particles to have the ability to self-summarize and learn from excellent individuals in the group, so as to get closer to their historical best advantage and the historical best advantage in the group. Traditionally, the fixed value is 2.0.$rand_1^d$ and $rand_2^d$ is random numbers on two [0,1] intervals.

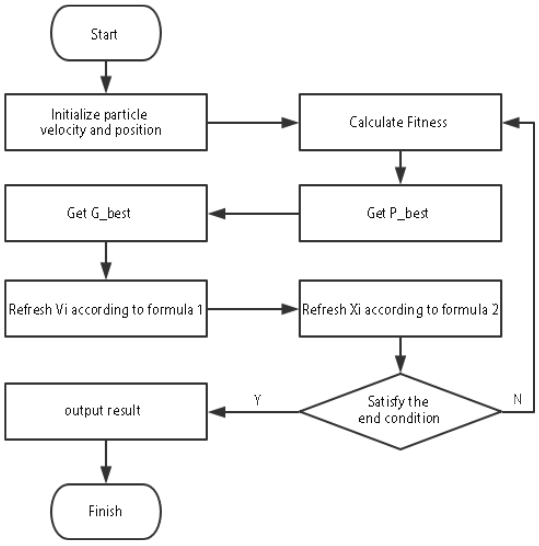


Figure 2. PSO algorithm flow chart

The particle velocity is limited to the maximum velocity of V_{Max}, and each dimension of V_{Max} can take 10%-20% of the corresponding dimension. In addition, the update of the position in the formula (2) must be legal, so the legality detection must be carried out after each update, and the illegal position should be adjusted. The general correction method can be re-randomly set or restricted in the boundary.

2.2.2 Social Behavior Analysis of Particles

As can be seen from the update equation of particle velocity (1), the particle velocity in the basic particle swarm optimization algorithm is mainly composed of three parts.

1. **The velocity of itself in the previous iteration** Equation(1)Right hand side first term, This is the inertia action in the particle flight, is the basic guarantee that the particle can fly.

2. **The Self-knowledge Part** Equation(1)Second term on the right, Represents the particle's own experience taken into account in flight, and moves closer to the best point it has ever found.

3. **Social Experience** Equation(1)Third term on the right, It means that particle flying takes into account social experience and learns from other particles in the neighborhood, so that the particle flies closer to the global best point.

2.3 Standard PSO Algorithm

In order to improve the convergence performance of the algorithm, Shi and Eberhart entered the concept of inertia weight in their 2002 paper [33]. The position update formula of the inertia weight algorithm follows the position update formula in the basic PSO (2), and the speed update formula is as follows (3) :

$$v_i^d = \omega \times v_i^d + c_1 * rand_1^d * (pBest_i^d - x_i^d) + c_2 * rand_2^d * (gBest^d - x_i^d) \quad (3)$$

In the study of Shi and Ebhert, inertia weight in the form of a linear gradient from 0.9 to 0.4, weight in the process of adjustment, the particles search is also gradually by based on the breadth of the global search to search based on local depth, has realized the balance of global search and local search, weaken or even eliminate the basic particle swarm algorithm for maximum velocity of turbulent flow boundary treatment effect the performance of the algorithm. A better optimization strategy is that in the early stage, the algorithm has a better breadth search ability, so as to get an excellent seed solution; in the later stage, the algorithm should have a better depth search ability, so as to speed up the convergence speed of the algorithm.

Inertia weight ωreflects the current particle's ability to inherit the velocity of the previous generation. When ω=1, the algorithm degenerates into the elementary particle swarm optimization (PSO) algorithm. When $\omega \geq 1$, the particle velocity increases gradually with the number of iterations, and it is easy to make the particle fly through the optimal solution. When $0 \leq \omega \leq 1$, the velocity of the particle gradually decreases in the process of evolution, and the convergence performance depends on the acceleration parameters c1 and c2.

At present, most researches on particle swarm optimization algorithms take particle swarm optimization algorithm with inertia weight as the object for analysis, expansion and correction. Therefore, most literatures refer to particle swarm optimization algorithm with inertia weight as the standard version of particle swarm optimization algorithm, or the standard particle swarm optimization algorithm. The above particle swarm optimization algorithm is called initial particle swarm optimization algorithm/basic particle swarm optimization algorithm.

2.4 Algorithm Elements

The basic particle swarm optimization (PSO) consists of three parts: PSO coding method, individual fitness evaluation and PSO running parameters. Here is an overview of the components of particle swarm optimization algorithm. The operation parameters of PSO include the relevant parameters of the algorithm: group size, learning factor, maximum speed and inertia weight. It also includes neighborhood topology, initialization of particle space and stop criteria.

2.4.1 Encoding Method

Elementary particle swarm optimization (PSO) USES a fixed length string of binary symbols to represent individuals in a population, whose alleles are composed of a set of binary symbols 0,1. The genetic values of individuals in the initial population can be generated by uniformly distributed random Numbers.

2.4.2 Evaluation of Individual Fitness

By determining the local optimal iteration to reach the global optimal convergence, the results are obtained.

2.4.3 Running Parameters

1. **Population Size m.** M is an integer parameter. When m is very small, the possibility of getting trapped in the optimal situation is very large;When m is very large, the calculation time will be greatly increased and the convergence speed will be very slow.

2. **Learning factors c_1 and c_2.** The learning factors c_1 and c_2 learning factors enable the particles to have the ability to self-summarize and learn from the excellent individuals in the group, so as to approach the best advantages in the group or neighborhood. C1 and c2 are usually equal to 2, but there are other values. But typically c1 is equal to c2, and it ranges from 0 to 4.

3. **Maximum Speed V_{max}.** The maximum velocity determines the maximum moving distance of the particle in the sequential iteration. When V_{Max} is large, it has strong exploration ability, but it is easy to fly over the best solution. V_{Max} is small, strong development ability, but easy to get into the game. Analysis and experiment show that the effect of V_{Max} can be realized by adjusting the inertia weight of ω, so now the experiment basically USES V_{Max} for initialization.

4. **Inertia Weight ω.** The balance between exploration capability and development capability is crucial to the success of intelligent optimization. For particle swarm optimization (PSO), the balance of these two abilities is realized by inertia weight. When ω is larger, the particle has better exploration ability. When ω is smaller, the particle has better development ability.

5. **Neighborhood topology.** The global version of particle swarm optimization algorithm takes the whole group as the neighborhood, which is fast, but has the possibility of falling into the local optimization. The local version of PSO takes the individual with similar index number or position as the neighborhood of the particle, and the convergence speed is slow, but it is not easy to fall into the local optimization.

6. **stopping criterion.** Generally, the maximum number of iterations or acceptable satisfactory solution is used as the stop criterion.

7. **Initialization of Particle Space.** Better selection of initialization space of particles will greatly shorten the convergence time. That's what the problem depends on.

3. Improvement and Deformation of PSO

The improvement of PSO algorithm is mainly aimed at its three components: inertia weight, neighborhood topology and learning factor. Later, some scholars proposed another important improved version of the particle

swarm optimization algorithm: citeBui2007A;Then, for discrete optimization problem, two typical discrete versions of particle swarm optimization algorithm named citeKennedy2002A were proposed. Several improved strategies based on genetic thought[34] and gradient information[35] were also proposed[36][37]. Finally, the solution of algorithm in two complex environments is presented: constrained optimization and multi-objective optimization[38].

3.1 The Selection and Adjustment of Three Constituent Elements

3.1.1 Inertia Weight

Inertia weight is an important parameter of standard particle swarm optimization algorithm, which plays a decisive role in the quality of the algorithm. Nickabadi etc. divided the different inertia weights into three types [39]: (1) constant weights and random weights, such as ω=0.721 or ω= 0.5+rand()/2.0; (2) Time-based dynamic weights, like$\omega = (\frac{2}{iter})^{0.3}$ or $\omega = (\frac{iter_{max}-iter_i}{iter_{max}}) * (\omega_{init} - \omega_{end}) + \omega_{end}$; (3) adaptive weight [40], such a weight. Is usually based on a feedback mechanism.

- **Fixed Weight**

  That is to give the inertia weight a constant value, in general, the value between 0 and 1.The fixed inertia weight makes the particle always have the same exploration and development ability in flight. Obviously, for different problems, the constant to obtain the best optimization effect is different, and a large number of experiments are usually required to determine the constant value. It is found through experiments that the smaller the population size is, the greater the inertia weight is required, because the population needs better exploration ability to make up for the insufficient number of particles. Otherwise, particles are easy to converge. The larger the population size, the smaller the inertia weight required, because each example can be more focused on searching the area near itself.

- **Random Weighting**

  The random weight is evaluated randomly within a certain range citeZhao2013A. For example, you could evaluate it as follows:

$$\omega = 0.5 + \frac{Random}{2} \tag{4}$$

  Where Random is a Random number between 0 and 1.In this way, the inertia weight will change randomly between 0.5 and 1, with an average value of 0.75.The reason for this setting is to apply to the dynamic optimization problem. The inertia weight is set as linearly reduced Time-varying weight in order to make the particle swarm have a better global search ability at the beginning of iteration and a better local search ability at the end of iteration in the static optimization problem. For the dynamic optimization problem, it is impossible to predict whether the particle swarm needs better global search ability or local search ability at a given time, so the inertia weight can change randomly within a certain range.

- **The Time-varying Weight**

  Generally speaking, the particle swarm is expected to have a better exploration capability at the beginning of flight, and with the increase of the number of iterations, especially at the later stage of flight, it is expected to have a better development capability. Can be achieved through the setting of Time-varying weight. Shi Y presents a Linear Decreasing inertia Weight (Linear Decreasing Intertia Weight, LDIW) [41], namely:

$$\omega(k) = \frac{\omega_{start} * (\omega_{start} - \omega_{end}) * (T_{max} - k)}{T_{max}} \tag{5}$$

  Where, ω_{start} is the initial inertia weight; ω_{end} is the inertia weight at the maximum number of iterations;K is the current iterative algebra; T_{Max} is the maximum iterative algebra. In general, the algorithm performs best when the inertia weight is ω_{start} =0.9 and ω_{end} =0.4.Thus, with the progress of iteration, the inertia weight decreases linearly from 0.9 to 0.4, and the particle swarm optimization algorithm with linear decreasing inertia weight is denoted as ldiw-PSO. The large inertia weight at the beginning of iteration makes the algorithm maintain a strong global search capability, while the small inertia weight at the end of iteration makes the algorithm carry out more accurate local search. This is a change mode of linear reduction, but also can be used to set the inertia weight of the change mode of nonlinear reduction.

  Common choices of inertia weights include the following:

$$\begin{aligned}
&\omega(k) = \omega_{start} - (\omega_{start} - \omega_{end})(\frac{k}{T_{max}})^2 \\
&\omega(k) = \omega_{start} + (\omega_{start} - \omega_{end})[\frac{2k}{T_{max}} - (\frac{k}{T_{max}})^2] \\
&\omega(k) = \omega_{end}(\frac{\omega_{start}}{\omega_{end}})^{\frac{1}{1+\frac{ck}{T_{max}}}} \\
&\omega(k) = \omega_{max} - k * \frac{\omega_{max} - \omega_{min}}{T_{max}}
\end{aligned} \tag{6}$$

  Determine the maximum weight ω_{Max} and the minimum weight ω_{min} based on the actual problem. Linear Time-varying weight is the most widely used method in practical application.

- **Adaptive Weight**

  This type of weight generally needs to dynamically adjust the feedback information of the group's search situation.

3.1.2 Neighborhood Topology

How to define the neighborhood composition of particles, namely the topological structure of neighborhood, is a basic problem in algorithm implementation[42]. The

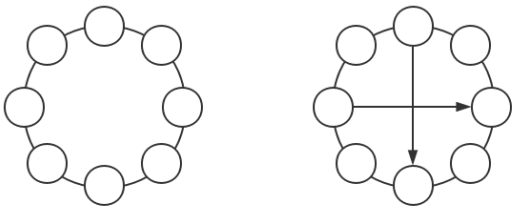
following is a detailed description of the neighborhood composed of particles adjacent to the index number and the neighborhood composed of particles adjacent to the position:

1. **Topological Structure Based on Index Number**

   The biggest advantage of this kind of topology is that the relative positions of particles are not taken into account when determining the neighborhood, so as to avoid the computation cost when determining the neighborhood.

   (a) **Circle Structure**

   Ring structure is a basic neighborhood topological structure. Each particle is only connected to its direct k neighbors, that is, k particles with similar index number of the particle constitute the neighborhood members of the particle. Under the circular structure, one part of the population may gather in one game, while another part may gather in different games, or continue searching to avoid falling into the game prematurely. The effects between neighbors are passed on one by one until the best advantage is found by any part of the population, and then the whole population converges.



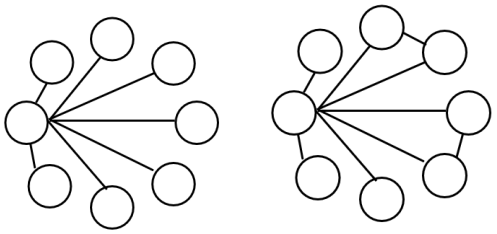
(a) Ring Topology (b) Ring topology with shortcuts

Figure 3. Two ring topologies.

   Two shortcuts can be added to the ring topology to get a ring topology with shortcuts that enhances the communication between different particle neighborhoods. In this way, the changed ring topology shortens the distance between the neighboring regions, and the population will converge faster.

   (b) **Wheel Structure**

   A wheel is a structure in which one particle is the focus, and all the other particles are connected to that focus, while the other particles are not connected to each other. In this way, all the particles can only communicate with the focus particles, effectively separating the particles. The focus particle compares the behavior of all particles in its neighborhood, and then adjusts its flight path to get closer to the best point. This refinement then spreads to other particles through the focal point. Therefore, the focus particle ACTS as a buffer to mitigate the diffusion rate of the better solution in the population.



(a) Wheel topology. (b) Wheel topology with shortcuts.

Figure 4. Two kinds of wheel topology.

   Two kinds of wheel topology can also be added into the wheel topology to get a wheel topology with a shortcut. This has two effects. First, it can produce mini-neighborhood, in which the peripheral particles are affected by the particles directly connected to the focus particles. In this way, the cooperative sub-population in the mini-neighborhood can converge faster, and the buffer of the focus particles can prevent the whole population from premature convergence to local optimization. On the other hand, it is also possible to generate islands, or separate the connections between subpopulations, so that the subpopulations can cooperate and independently optimize the problem. This will lead to a decrease in the exchange of information, so that those separated individuals cannot get the good areas of the whole population. It also prevents other particles in the population from sharing information about the success of the individual search.

   (c) **Star Structure**

   The star topology is that each particle is connected to all the other particles in the population, making the whole population its neighborhood. That's the global version of particle swarm optimization. In this structure, the information Shared by all particles is the information of the best performing particles in the population.

   (d) **Random Structure**

   The random structure is the random establishment of N symmetric pair-wise connections among the population of N particles.

2. **Distance-based Topology**

   The distance-based topology is to calculate the distance between a particle and other particles in the population at each iteration, and then determine the

composition of the neighborhood according to the distance.

At the beginning of the dynamic neighborhood topology search, the neighborhood of particles has only itself. With the increase of the number of iterations, the neighborhood gradually increases until all particles in the population are members of the neighborhood. In this way, the initial iteration has better exploration performance and the later iteration has better development performance.

For the particle I in the neighborhood to be calculated, calculate its distance from all other particles in the population. The distance between this particle and partical l(l≠i) denoted as dist[l]. Max distance denotes as $dist_{max}$.

Define a function frac about the current number of iterations:

$$frac = \frac{3 * ITER + 0.6 * MAXITER}{MAXITER} \tag{7}$$

When frac≤0.9, the particles that satisfy the following conditions constitute the neighborhood of the current particle I, $\frac{dist[l]}{dist_{max}} \leq$frac.

When frac≥0.9, Take all the particles in the population as the neighborhood of the current particle I.

3.1.3 Learning factor

The learning factor is generally fixed as a constant, and its value is 2.Other methods have also been tried.

1. **c_1 and c_2 Synchronous Time-varying**

   In the experiment, according to the setting method of Time-varying inertia weight, Suganthan set the learning factor as follows: ammume value of learning factor c_1 and c_2 range is $[c_{min}, c_{max}]$, The maximum number of iterations is $Iter_{max}$, Then, the learning factor at the ith iteration is:

$$c_1 = c_2 = c_i = c_{max} - \frac{c_{max} - c_{min}}{Iter_{max}} * i \tag{8}$$

   Then, the learning factor at the ith iteration is taken as the change method of synchronous linear reduction of two learning factors, which is called synchronous time variation.

2. **c_1 and c_2 Asynchronous Time-varying**

   Ratneera tec. proposed another Time-varying method for setting learning factors[43], Another Time-varying learning factor setting method is proposed to make two learning factors change differently during iteration, which is called asynchronous Time-varying. The purpose of asynchronous Time-varying is to strengthen global search at the beginning of iteration and local search at the end of iteration to promote convergence, namely decreasing c_1 and increasing c_2.

   Specific implementation methods are as follows:

$$c_1 = (c_{1f} - c_{1i}\frac{iter}{Iter_{max}} + c_{1i})c_2 = (c_{2f} - c_{2i}\frac{iter}{Iter_{max}} + c_{2i}) \tag{9}$$

   Where, c_{1i}, c_{1f},c_{2i},c_{2f} is for constant, are the initial value and final solution of c_1 and c_2 respectively. $Iter_{max}$ is the maximum number of iterations, iter is the number of current iterations. Ratnaweera etc. In the study, it was found that in most of the problems, the following Settings had a good optimization effect:

$$c_{1i} = 2.5, c_{1f} = 0.5, c_{2i} = 0.5, c_{2f} = 2.5$$

   In particular, the combination of asynchronous Time-varying learning factors and linearly reduced Time-varying weights is effective.

3.2 Particle Swarm Optimization Algorithm with Shrinkage Factor

Clerc introduced the concept of shrinkage factor into the original particle swarm optimization algorithm[44], it is pointed out that this factor is necessary for the convergence of the algorithm. In particle swarm optimization algorithm with shrinkage factor, the updating equation of velocity is shown below(10)

$$\begin{aligned} v_i^d = K[v_i^d + c_1 * rand_1^d * (pBest_i^d - x_i^d) + \\ c_2 * rand_2^d * (gBest^d - x_i^d)] \end{aligned} \tag{10}$$

Where, K is function of c_1 and c_2:

$$K = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \phi = c_1 + c_2 \geq 4 \tag{11}$$

Clerc set parameter c_1=c_2=2.05, ϕ=4.1.Then get K=0.729. Obviously, if the standard particle swarm optimization algorithm takes the parameter:

$$\omega = 0.729$$

$$c_1 + c_2 = 0.729 * 2.05 = 1.49445$$

The standard particle swarm optimization algorithm is equivalent to the particle swarm optimization algorithm with contraction factor. That is, the PSO with shrinkage factor is a special case of the standard PSO.

3.2.1 Binary Encoding

The discrete particle swarm optimization algorithm was first proposed by Kennedy [45], Binary encoding is used. So each of the position vectors of the particle has a value of 0 or 1.Since the PSO algorithm with binary encoding was proposed earlier, the speed updating formula of the basic PSO algorithm was adopted. The update formula of particle velocity with binary encoding is (1), and the update formula of position is as follows:

$$x_i^d = \begin{cases} 1, & random \leq S(v_i^d) \\ 0, & \text{others} \end{cases} \tag{12}$$

$$S(v_i^d) = \frac{1}{1 + e^{-v_i^d}} \tag{13}$$

Although the above velocity updating equation is identical to the velocity updating formula of PSO, the meaning and position of velocity have changed. Here, v_i^d no longer represents the flying speed of the particle during iteration. Its significance lies in the probability that the value of x_i^d is 0 or 1 according to the value of the velocity, which is expressed by the function $S(v_i^d)$. That is, no matter the bit value of the particle is 0 or 1 in k iteration, the value of x_i^d in k+1 iteration is 1 with probability $S(v_i^d)$, and 0 with probability $1 - S(v_i^d)$. The binary version takes the velocity v_i^d as the probability that the particle has a value of 1, so it is necessary to transform v_i^d into the interval [0,1], that is, the function $S(v_i^d)$.

In addition, the binary version still needs to be limited to V_{Max}. As can be seen from the calculation, when v_i^d>10, the value of $S(v_i^d)$ will be very small, resulting in the value of x_i^d being almost certain to be 0, which loses the significance of probability value, thus requiring the limit of V_{Max}. Kennedy thinks it is good to set $V_{Max}$to 6, when $S(v_i^d)$ is between 0.9975 and 0.0025.It is worth noting that in the basic particle swarm optimization algorithm applied to continuous space, the larger $V_{Max}$is, the better the particle exploration ability will be. In the binary version, however, the opposite is true. The greater the value of V_{Max}, the lower the change probability, because the negative value of v_i^d may be large in absolute value, but the probability is small after the calculation of $S(v_i^d)$.

3.2.2 Sequential Encoding

Hu X et al. presented a particle swarm optimization algorithm for solving the permutation problem, [46]. Because its coding rules are the same as the sequential coding of genetic operators, it is called the sequential coding particle swarm optimization algorithm. Here is an example of coding. Here is an example of coding.

$$X = (1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7)$$

Here the code length is 7, which is also the range of bit values in the example.

In this improved discrete version of the algorithm, the velocity is also defined as the probability of particle variation, and the updating formula of the velocity remains unchanged. If the velocity is large, the particles are more likely to change into a new sequence. Here the velocity is obviously also restricted, mapped to [0,1]. The normalization of particle velocity is as follows:

$$Swap(v_i^d) = \frac{|v_i^d|}{n} \tag{14}$$

Obviouslv, value of $Swap(v_i^d)$ range in [0,1], It determines whether the encoding of a particle produces an exchange. If an exchange is generated with this probability, it is best to solve the corresponding bit value when the d-th position of particle I changes to the neighborhood after the exchange.

Since the particle is similar to the arrangement of the optimal solution in the neighborhood with a certain probability, if it is the same as the arrangement of the optimal solution in the neighborhood, it will remain unchanged. In order to avoid this situation, variation is introduced to overcome it, that is, when the particles and the optimal solution of the neighborhood are arranged at the same time, two positions in the code are randomly selected and their bit values are exchanged.

3.3 An Improved Strategy Based on Genetic Thought and Gradient Information

Genetic strategy and gradient information are two important improvement strategies. Genetic algorithm is widely used, and its basic genetic strategies, including selection, hybridization and variation, can achieve good optimization effect. Gradient information will greatly improve the optimization efficiency of the algorithm.

3.3.1 An Improved Algorithm Based on Selection

Angeline combined natural selection mechanism with particle Swarm optimization algorithm and proposed a Hybrid Swarm Swarm algorithm[47]. The hybrid algorithm USES the Tournament Selection Method, in which each particle compares its fitness in its current position with the fitness of k other particles, records the worst score, and then queues up the entire swarm of particles in order of rank. In this process, the historical optimal value of the individual is not considered. After the group sequencing is completed, replace the position and speed of the worst side with the current position and speed of the best half of the group, while retaining the original historical information so as to update the position of the next generation of particles.

This selection method should be added to make the hybrid algorithm have a stronger search ability, especially for the current better area development ability, so that the convergence speed is accelerated. However, it increases the likelihood of falling into a trap.

3.3.2 An Improved Algorithm Based on Crossover

Lovbjerg et al[36]. proposed a hybrid particle swarm optimization algorithm with crossover and subpopulation. The structure of this hybrid algorithm is shown below.

Algorithm 2: An improved algorithm based on crossover

Input : Number of iterations T; Population size NP; Problem dimension D; Hybrid probability BC; Hybrid pool size ratio BS

Output : Globally optimal position vector$\boldsymbol{x^*(t)}$

```
t ← 1(initialization);
Initializes the position vector of the particle;
Initializes the velocity vector of the particle;
while (|f(x(t)* ≥ ε)|)or(t≤T) do
    evaluate;
    Calculate the new speed according to formula (16);
    Move according to formula (15);
    Cross breeding;
end
return x*(t);
```

Here, the calculation method of speed adopts the formula combining inertia weight and contraction factor (10),

and the updated formula of position is (2).

The way of crossover is as follows: during each iteration, a certain number of particles in the particle swarm are selected and put into a pool according to a certain probability. The particles in the pool are randomly crossed in pairs to produce a corresponding number of children particles, and children particles are used to replace their parents to keep the population size unchanged.

In each dimension, the position of children is obtained by cross calculation of the position of parents:

$$\begin{aligned} child_1(x_i) &= p_i * parent_1(x_i) + (1.0 - p_i) * parent_2(x_i) \\ child_2(x_i) &= p_i * parent_2(x_i) + (1.0 - p_i) * parent_1(x_i) \end{aligned} \tag{15}$$

Here, p_i is a random number between 0 and 1.The velocity vector of children is obtained by normalizing the sum of the velocity vectors of parents:

$$\begin{aligned} child_1(v) &= \frac{parent_1(v) + parent_2(v)}{|parent_1(v) + parent_2(v)|}|parent_1(v)| \\ child_2(v) &= \frac{parent_1(v) + parent_2(v)}{|parent_1(v) + parent_2(v)|}|parent_2(v)| \end{aligned} \tag{16}$$

The idea of subpopulation is to divide the whole particle swarm into subpopulations, each of which has its own internal historical optimal solution. These cross operations can be performed within the same subpopulation or between different subpopulations. Cross operation and subpopulation operation, can make the particle benefit from both parents, enhance the search ability, easy to jump out of the game.

3.3.3 Improved Algorithm Based on Variation

Higashi and lba use gaussian variation in evolutionary computation to avoid getting trapped in the particle swarm ([48]) by incorporating it into the particle swarm position and velocity update. When each particle moves to another position in the search area, it is not only based on the prior probability, as the standard formula does, and is not affected by other particles, but adds certain uncertainty through gaussian variation. The calculation formula of variation is:

$$mut(x) = x * [gaussian(\sigma)] \tag{17}$$

Where, mut(x) is the position of the mutated particle, and $\sigma$is 0.1 times of the length of each dimension of the search space. The experiment shows that σhas the best effect. The algorithm selects individual mutants with predetermined probability and determines their new location with gaussian distribution. In this way, a wide range of search can be carried out in the early stage of the algorithm, and then the search efficiency can be improved by gradually reducing the mutation probability in the middle and late stage of the algorithm. The authors set the variation probability to decrease linearly from 1.0 to 0.1.

The hybrid algorithm based on gaussian variation is more likely to jump out of the local optimum, so it has better performance in solving multi-peak functions.

Stacey et al. attempted to use Cauchy distribution for mutation operation [49], and its probability distribution function was:

$$f(x) = \frac{a}{\pi} * \frac{1}{x^2 + a^2} \tag{18}$$

Where, a=0.2.The Chuchy distribution is similar to the normal distribution, but has a larger probability at the tail, which increases the probability of producing a larger value. Moreover, the variation probability of each component is set as $\frac{1}{d}$.

3.3.4 Improved Algorithm with Gradient Acceleration

Gradient information often contains some important information of the objective function. For the function f(x), $x = (x_1, x_2, \ldots, x_n)$, the gradient can be expressed as: $\nabla f(x) = [\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}]^T$,The negative gradient direction is the fastest decreasing direction of the function value.

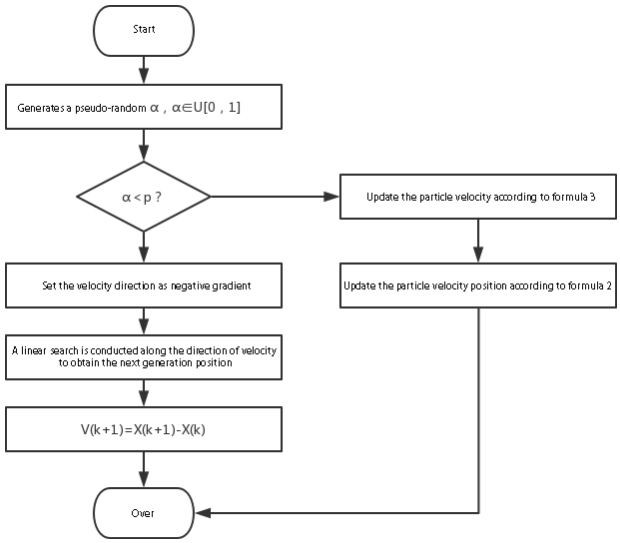


Figure 5. Flow diagram of gradient acceleration

Jun-wei Wang, etc. [50], by adding gradient information to influence the update of velocities of the particles, to construct a particle swarm optimization algorithm with gradient acceleration, updating speed and position of each particle, the probability of each particle to rhoshall be carried out in accordance with the standard particle swarm optimization algorithm formula update, a straight line in the negative gradient direction search to determine the moving step length.

The addition of gradient information makes the movement of particles more targeted and greatly improves the convergence speed of PSO algorithm, but it will increase the problem dependence of the algorithm. The gradient information of some problems is easy to introduce particles into local optimization.

3.4 Algorithm in Constrained Optimization and Multi-objective Optimization Solution

3.4.1 Treatment of Constraints

Constraints of PSO can be used for reference from other optimization algorithm constraints processing methods, and special processing methods can also be designed according to the characteristics of PSO algorithm[48].

1. **Penalty Policy**

   The key to the introduction of punishment strategy into PSO algorithm lies in the designed punishment function. For example, Parsopoulos USES non-fixed multi-stage assignment punishment function to solve the constraint problem[51][52].

2. **Rejection policies** The essence of rejection strategy is to discard all the infeasible solutions in the iterative process. According to the characteristics of the particle swarm optimization algorithm, Hu and Eberhart made the following improvements to maintain the feasibility of the solution: the particle searched in the whole space, but only tracked the feasible solution to accelerate the search process, the non-feasible solution would not be recorded in the historical information, and all particles were initialized to the feasible solution, as follows.

   (a) In initialization, all particles are initialized repeatedly until all constraints are satisfied.

   (b) Only feasible solutions are retained when the best historical solutions of individuals and neighborhoods are calculated and preserved.

   However, the above method may be difficult to find the initial viable population. El - Galled using similar methods, such as [53], when particles fly out feasible space, the resetting to history the best feasible solution, but this method may limit the particle in the region of the initial point.

3. **Pareto-based Approach**

   Ray et al. used multi-order information sharing strategy in particle swarm to deal with single-objective optimization problem [43], that is, Pareto sorting was used to generate Shared information.

   (a) Based on the constraint matrix, Pareto sorting was used to produce the best solution particle, [38], and put into the Better Performer List (BPL).

   (b) Particles other than BPL use information in the most recent BPL, that is, BPL particles form neighborhoods with nearby non-bpl particles.

   (c) In order to avoid premature development, simple evolutionary operators will be used instead of conventional formulas when referring to the information of Leader in the neighborhood. That is, the probability that the value of the variable is generated between the particle itself and the Leader is 50%. The probability that the value of the variable is generated between the lower limit of the value of the variable and the minimum value of the particle and the Leader is 25%. The probability that the value of the variable is generated between the upper limit of the value of the variable and the maximum value of the particle and the Leader is 25 %.

3.4.2 Multi-objective Processing

According to the characteristics of particle swarm optimization (PSO) algorithm, the paper applies PSO to solve multi-objective problem [54], and develops a memory-based method by selecting the optimal neighborhood solution. According to the current research situation, these methods can be divided into two categories: traditional methods and memory-based methods [55].

1. **Traditional Method**

   The weight sum method and vector evaluation method are adopted in Parsopoulos to solve the multi-objective problem, [51], [52].

   - The traditional linear weighting method, "bang-bang" weighting method and dynamic weighting method are adopted.
   - In the vector evaluation method, the individual evaluation method of the single objective optimization function is first used to evaluate the particles, so as to form different subpopulations, each particle is a better solution of a certain objective function. In flight, the particle is influenced by other sub-population information, so it flies towards the direction that satisfies other objective function components, gradually satisfies more objective function components, and finally flies towards Pareto optimal direction.

2. **Memory-based Approach** The information sharing mechanism in particle swarm optimization algorithm is different from other optimization algorithms. Only the neighborhood best solution transmits information to other particles, which is a one-way information sharing mechanism. Due to the characteristics of point attraction, traditional particle swarm cannot approach multiple Pareto optimal solutions at the same time, and can only use different weights to run the algorithm for multiple targets to find Pareto optimal solutions.

   As a non-independent agent, the best solution of individual and neighborhood plays a key role in particle flight, so the selection of optimal solution of individual and neighborhood is the key to solve the multi-objective optimization problem. At present, the research focuses on the selection of the optimal neighborhood solution, which can be divided into two steps.

   (a) **Determining the neighborhood.** That is to determine the topology of the neighborhood, in the multi-objective processing, some special neighborhood strategy can be used.

(b) **Select a well-behaved particle from the neighborhood as the optimal solution.** The selection of neighborhood optimal solution should satisfy two principles: it can guarantee the convergence of the algorithm;Can maintain a balance between searching Pareto solution set and maintaining population diversity. including:

- **Wheel method.** According to a rule, each candidate particle in the neighborhood is given a weight, and then it is selected randomly by the method of rotating wheel to maintain the diversity of the population.
- **Quantitative method.** In other words, the neighborhood optimal solution is selected by the specific selection method and the definite value is obtained.

4. Particle Swarm Optimization Algorithm Simulation Verification

4.1 Experiment of PSO Algorithm Under Different Iteration Steps

With the iteration of PSO algorithm, the result is closer to the optimal value of the expected function. This experiment is used to explore the influence of different iteration steps on the performance of PSO algorithm.

4.1.1 Experimental Result

CHere, the standard PSO algorithm is adopted to test Ackley function. In the case of population size N=20 and dimension D=30, the experimental results are shown in the table1

Table 1. The minimum value of objective function under different iteration steps

Iteration steps	100	1000	10000
Fitness	2.64E-01	2.41E-01	2.56E-01

4.1.2 Experiment Conclusion

As can be seen from the table, the number of iterative steps is not necessarily proportional to the accuracy of the obtained solution, that is, the greater the number of iterative steps, the higher the accuracy of the obtained solution is not necessarily. This is because PSO is a random algorithm, and the same parameters will produce different results. In particular, without any improvement, the standard PSO often appears the phenomenon of "prematurity", that is, the algorithm does not update after falling into the local optimal value, which is also one of the reasons why the convergence accuracy is no longer improved as the number of iteration steps increases.

4.2 Experiment of PSO at Different Particle Swarm Scales

PSO searches the surrounding region for the optimal solution for experimental particles with different particle swarm sizes. Therefore, theoretically, the larger the particle size is, the higher the particle density in the feasible domain is, and the higher the possibility of finding the optimal solution is, and the higher the solution accuracy is. This experiment is used to explore the influence of different population sizes on the performance of PSO algorithm.

4.2.1 Experimental Result

In this experiment, the standard PSO algorithm was used to test Ackley function under the global search structure, with the maximum number of iterations T=100 and dimension D=30.The experimental results are shown in the table below2

Table 2. The minimum value of objective function under different population size

pop size	10	100	1000
Fitness	6.62E-01	2.10E-02	1.10E-04

4.2.2 Experiment Conclusion

As can be seen from the table, the larger the population size, the higher the solution accuracy. However, as PSO algorithm is a random algorithm, there are also situations of small population size but high precision, or large population size but low precision. In particular, the standard particle swarm optimization algorithm is insensitive to the size of the population, that is, the decrease of the population size has little impact on the optimization ability of the algorithm.

4.3 Simulation Experiment of Objective Function Minimum Value SPSO and its Improved Algorithm Under Different Population Size

4.3.1 Test Function

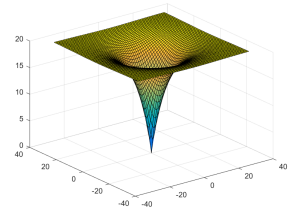
In order to evaluate the convergence speed, global search capability and solution accuracy of PSO algorithm, 8 typical benchmark functions are adopted for comparative analysis, among which $f_1, f_2, f_3 and f_4$ are multi-peak functions, which are mainly used to test the global search capability of the algorithm; f_5, f_6, f_7, f_8Is unimodal function, It is mainly used to test the convergence speed and solution accuracy of the algorithm. The name of the test function and the value range of the variable are listed in table 3.

4.3.2 Algorithm and Parameter Setting

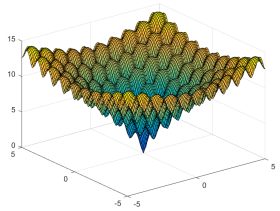
This paper selects the standard PSO algorithm based on global search (Global PSO,GPSO) and five classical improved PSO algorithms are compared. These algorithms include improved algorithms based on parameters, such as linear weight decline LDIW-PSO (Linear Decreasing Intertia Weight PSO) algorithm[41], adaptive weight adjustment APSO (Adaptation PSO) algorithm[44]; an improved backbone particle swarm optimization algorithm based on learning strategy (Bare-bones PSO,BBPSO) [56][57]; hybrid particle swarm optimization algorithm with crossover and subpopulation (Breed-PSO)[58]; an improved comprehensive learning strategy algorithm based on neighborhood topology (Comprehensive Learning PSO,CLPSO)[59].

Table 3. The eight test functions used in this article

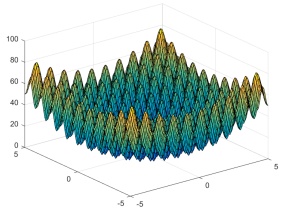
function name	test function	searching space	target value
Ackley	$f_1 = 20 - 20e^{-0.2\sqrt{\frac{1}{n}*\sum x_i^2}} + e - e^{\frac{1}{n}\sum cos(2\pi x_i)}$	(-32,32)	0
Rastrigin	$f_2 = \sum(x_i^2 + 10 - 10cos(2\pi x_i))$	(-5.21,5.21)	0
Griewank	$f_3 = \frac{1}{4000}\sum x_i^2 - \prod cos(\frac{x_i}{\sqrt{i}}) + 1$	(-600,600)	0
Alpine	$f_4 = \sum \lvert x_i sin(x_i) + 0.1x_i \rvert$	(-10,10)	0
Sphere	$f_5 = \sum x_i^2$	(-100,100)	0
Rosenbrock	$f_6 = \sum[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	(-30,30)	0
Schwefel P2.22	$f_7 = \sum \lvert x_i \rvert + \prod \lvert x_i \rvert$	(-10,10)	0
Sum of Different Power	$f_8 = \sum \lvert x_i \rvert^{i+1}$	[-1,1]	0



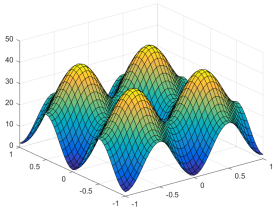
(a) Ackley in searching the space



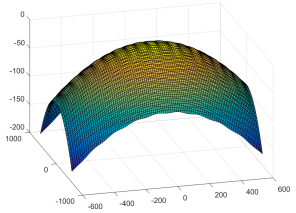
(b) Ackley near the extremum



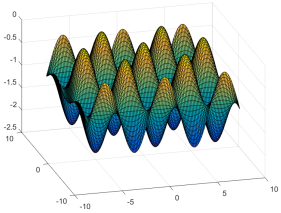
(c) Rastrigin in searching the space



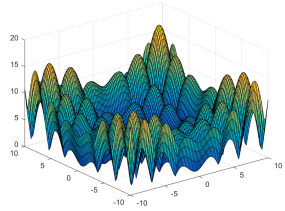
(d) Rastrigin near the extremum



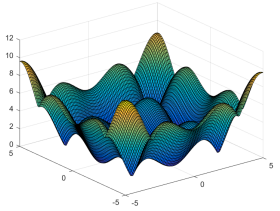
(e) Griewank in searching the space



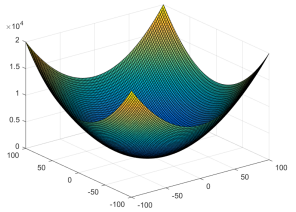
(f) Griewank near the extremum



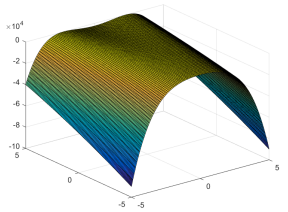
(g) Alpine in searching the space



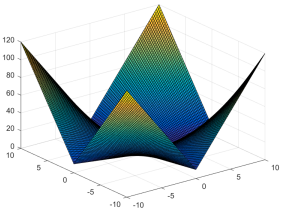
(h) Alpine near the extremum



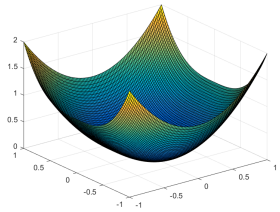
(i) Sphere in searching the space



(j) Rosenbrock near the extremum



(k) Schwefel P2.22 in searching the space



(l) Sum of Different Power in searching the space

Figure 6. Test function in its search space and near the extreme point image

The parameters of the algorithm in the experiment are shown in the table4:

Table 4. Experimental parameters of each comparison algorithm are set

Algorithm	Parameter Setting
PSO	ω=0.8,c_1=c_2=1.49445
LDIW-PSO	ω_{max}=0.9,ω_{min}=0.4,c_1=c_2=1.49445
APSO	ω_{max}=0.9,ω_{min}=0.4,c_1=c_2=1.49445
Breed-PSO	ω_{max}=0.9,ω_{min}=0.4,c_1=c_2=1.49445
BBPSO	No input parameters
CLPSO	ω_{max}=0.9,ω_{min}=0.4,c=1.49445,m=7

4.3.3 Algorithm Test Results

To verify the performance of the algorithm, the experiment was divided into two groups:

- **Group 1**Dimension of function variable D=10, population size N=20, maximum number of iterations T=1000
- **Group 2**Dimension of function variable D=30, population size N=80, maximum number of iterations T=1000

In order to reduce the error of experimental test and statistics, each group of experiments was repeated for 50 times in the experimental environment of MATLAB 2016a. The Mean of optimal fitness and standard variance of fitness were taken as the evaluation criteria.

Experimental results of the first group (dimension of function variable D=10, population size N=20, maximum number of iterations T=1000) are shown in table 5. The experimental results of the second group (dimension of function variable D=30, population size N=80, maximum number of iterations T=1000) are shown in table **??**. The algorithm with the best performance under each function has been shown in bold:

4.3.4 Results Analyse

The convergence images of the above experiments are shown in figure 6 and figure 7. Compared with the single peak function, the global optimal solution search of multi-peak function is relatively difficult. If the population diversity is poor, or the global optimization ability is insufficient, the algorithm is easy to fall into a local extreme point in the search space and the phenomenon of "premature" occurs.

In addition, it can be seen from the experiment that BBPSO algorithm based on topology change performs best in most cases. In particular, it performs extremely well when dealing with such unimodal functions as Sphere, Schwefel P2.22 and Sum of Different Power, and can quickly converge to a high degree. BBPSO was put forward by Kendy in 2003, [56]. The algorithm was proposed on the basis of analyzing the running trajectory of particles in standard PSO. The velocity term was eliminated in the algorithm, and the position of particles was directly obtained by random sampling that obeys gaussian distribution. The simple collaborative probabilistic search method of BBPSO can improve the search efficiency and accuracy of the algorithm, and avoid the complex parameter adjustment of the standard PSO algorithm. It has been successfully applied to constrained optimization problem[56] [57], data mining[60], power system regulation,[61] and many other fields. BBPSO shows good efficiency in dealing with the single peak problem, but in some multi-peak functions, the effect of BBPSO is not very ideal. In particular, the update mode of particles in BBPSO has a very intuitive physical meaning, which enables us to easily adjust the focus search range of particles. Therefore, BBPSO is a promising algorithm.

The unimodal function f_6-rosenbrock is a kind of non-convex ill function, also known as banana function. The global optimum of this function lies in a narrow valley. Many PSO algorithms do not perform satisfactorily on this function, and ldiw-PSO, which performs best on this function, is not satisfactory.

The particle swarm optimization (CLPSO) based on Comprehensive Learning PSO (Comprehensive Learning PSO) mainly learns pBest instead of gBest to update particle speed, which avoids the prematurity caused by gBest falling into local optimum and has a strong global search ability. Although the performance of CLPSO in the above tests is flat, it has a high probability that CLPSO will eventually converge to the global optimal and avoid falling into the local extreme point under the condition of sufficient iteration times, which has a strong robustness.

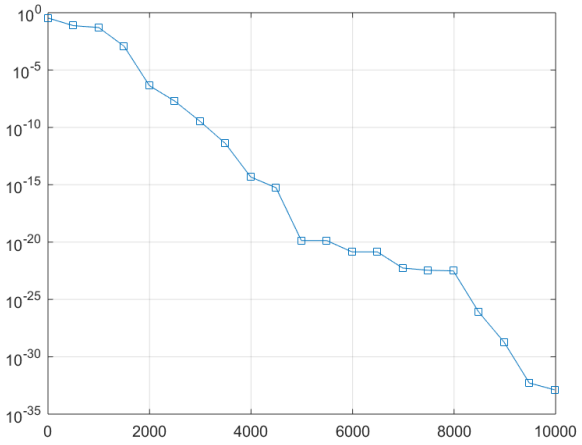


Figure 8. CLPSO iterates 10,000 results under the SDP function

5. Analysis and Summary

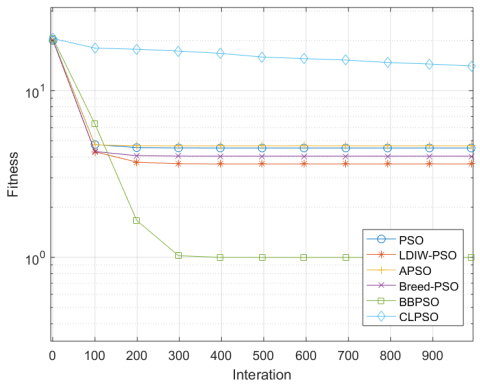
In this paper, the principle, process, social behavior analysis and algorithm components of particle swarm optimization are described in detail, and the development, research content, characteristics and application of particle swarm optimization and its improved algorithm are reviewed. Particle swarm optimization (PSO) is a stochastic search algorithm that simulates biological activity and swarm intelligence in nature. The position of each particle in the population represents a candidate solution in

Table 5. Compare the optimization results of the algorithm test function(D=10,N=20)

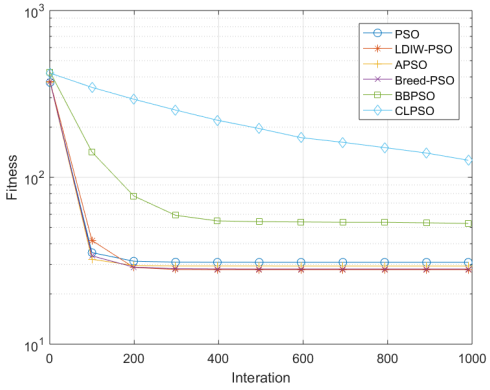
Test Function	Assessment Criteria	PSO	LDIW-PSO	APSO	Breed-PSO	BBPSO	CLPSO
Ackley	Mean	3.73E+00	2.05E+00	2.92E+00	2.79E+00	**5.28E-01**	1.04E+01
	Std. Dev	1.08E+00	1.06E+00	1.32E+00	**1.01E+00**	2.81E+00	2.51E+00
Rastrigin	Mean	1.21E+01	1.22E+01	1.34E+01	1.14E+01	**8.74E+00**	1.03E+01
	Std. Dev	5.22E+00	5.66E+00	6.49E+00	4.92E+00	**3.97E+00**	3.99E+00
Griewank	Mean	5.14E-01	2.20E-01	4.50E-01	3.45E-01	**8.62E-02**	5.94E+01
	Std. Dev	2.65E-01	1.05E-01	2.59E-01	1.98E-01	**4.74E-02**	1.79E+01
Alpine	Mean	6.38E-01	2.53E-01	6.47E-01	4.16E-01	**1.17E-14**	3.01E-01
	Std. Dev	6.19E-01	3.18E-01	8.06E-01	5.59E-01	**2.73E-14**	2.48E-01
Sphere	Mean	4.84E+00	4.52E-02	5.06E+00	3.70E-01	**5.25E-69**	1.08E+03
	Std. Dev	6.46E+00	1.04E-01	9.95E+00	6.18E-01	**3.36E-68**	7.37E+02
Rosenbrock	Mean	1.37E+02	**4.58E+01**	1.92E+02	7.33E+01	3.73E+03	5.05E+04
	Std. Dev	2.29E+02	**7.65E+01**	3.19E+02	1.78E+02	1.76E+04	7.60E+04
Schwefel	Mean	3.54E-01	1.88E-01	8.39E-01	6.41E-01	**1.23E-43**	2.19E+00
	Std. Dev	2.85E-01	3.19E-01	6.24E-01	4.99E-01	**3.73E-43**	2.20E+00
SDP	Mean	7.80E-07	7.03E-08	4.28E-07	4.49E-08	**4.336E-120**	3.89E-05
	Std. Dev	1.50E-06	2.91E-07	1.41E-06	1.01E-07	**3.015E-119**	9.68E-05

Table 6. Compare the optimization results of the algorithm test function(D=30,N=80)

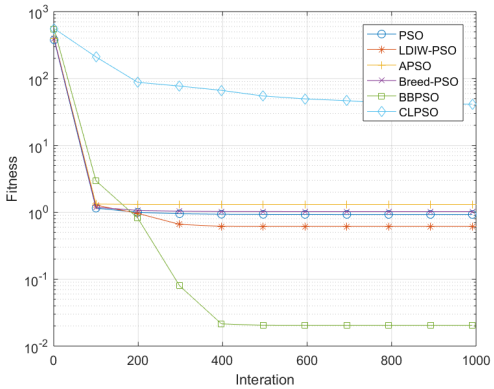
Test Function	Assessment Criteria	PSO	LDIW-PSO	APSO	Breed-PSO	BBPSO	CLPSO
Ackley	Mean	4.59E+00	3.76E+00	4.69E+00	4.19E+00	**7.92E-01**	1.40E+01
	Std. Dev	8.53E-01	7.44E-01	7.51E-01	**6.78E-01**	3.88E+00	1.03E+00
Rastrigin	Mean	3.86E+01	**2.78E+01**	3.09E+01	3.22E+01	4.89E+01	1.23E+02
	Std. Dev	9.83E+00	**9.01E+00**	9.67E+00	9.30E+00	1.85E+01	1.48E+01
Griewank	Mean	1.41E+00	6.02E-01	1.31E+00	9.58E-01	**1.66E-02**	4.56E+01
	Std. Dev	3.20E-01	2.22E-01	1.32E-01	1.62E-01	**1.63E-02**	8.02E+00
Alpine	Mean	2.29E+00	1.22E+00	1.69E+00	1.78E+00	**7.99E-01**	9.90E+00
	Std. Dev	1.66E+00	**1.22E+00**	1.44E+00	1.25E+00	1.71E+00	1.28E+00
Sphere	Mean	3.97E+01	8.86E-01	3.47E+01	5.92E+00	**1.72E-18**	2.44E+03
	Std. Dev	2.68E+01	6.98E-01	2.01E+01	3.88E+00	**6.88E-18**	6.58E+02
Rosenbrock	Mean	9.69E+02	**1.54E+02**	7.29E+02	2.69E+02	7.76E+03	1.21E+06
	Std. Dev	6.73E+02	**1.83E+02**	5.99E+02	2.08E+02	2.43E+04	4.89E+05
Schwefel	Mean	3.83E+00	**2.16E+00**	4.89E+00	3.20E+00	5.60E+00	3.07E+01
	Std. Dev	1.85E+00	1.36E+00	1.46E+00	**1.10E+00**	8.04E+00	4.68E+00
SDP	Mean	1.50E-08	2.42E-10	3.45E-10	1.64E-11	**3.01E-32**	1.16E-03
	Std. Dev	2.94E-08	4.85E-10	1.05E-09	5.24E-11	**1.89E-31**	1.11E-03



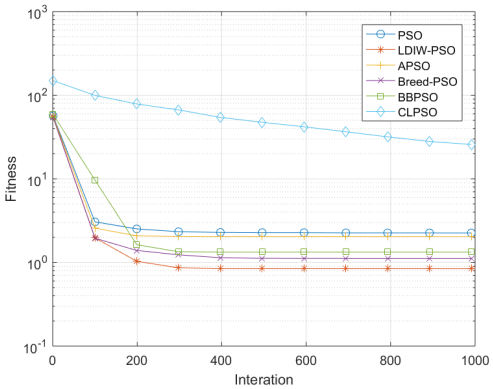
(a) Ackley comparison of convergence results



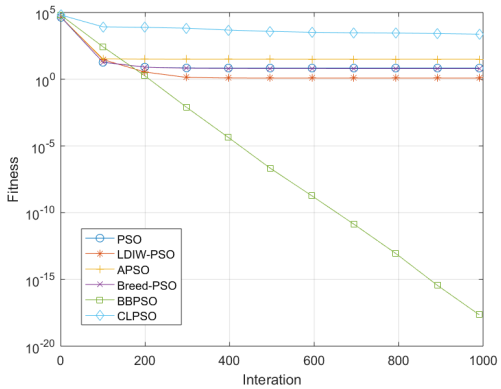
(b) Rastrigin comparison of convergence results



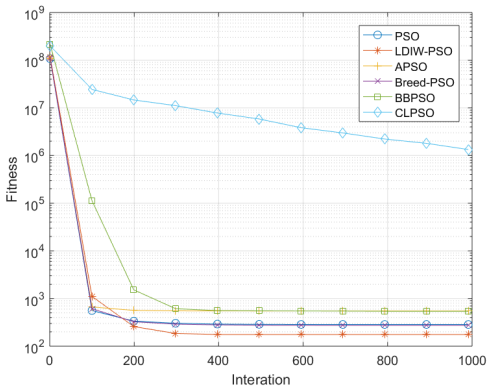
(c) Griewank comparison of convergence results



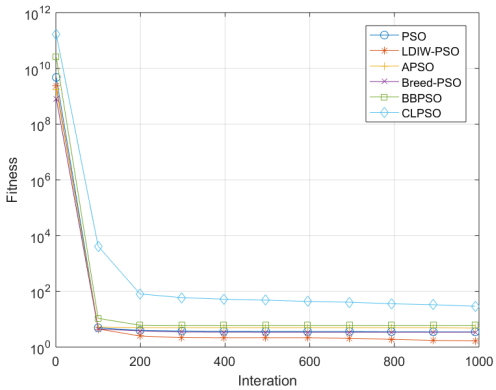
(d) Alpine comparison of convergence results



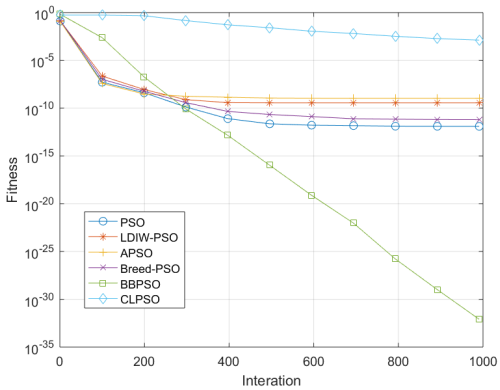
(e) Sphere comparison of convergence results



(f) Rosenbrock comparison of convergence results



(g) Schwefel P2.22 comparison of convergence results



(h) Sum of Different Power comparison of convergence results

Figure 7. The test function converges in D=30 dimensions

the search space. Individual particles adjust their search direction through the communication and learning of location information. Particle swarm optimization because of its algorithm is simple, easy to implement, without gradient information, less parameters etc in continuous optimization problems, and discrete optimization problems are showed good result, especially because of its natural characteristics of real number encoding is suitable for processing the optimization problem, in recent years become a popular in research of intelligent optimization in the world. As an important optimization tool, particle swarm optimization (PSO) algorithm has been successfully used in the fields of objective function optimization, neural network training, fuzzy control system, pattern recognition, signal processing, image processing, statistical learning model parameter optimization, machine learning model parameter optimization, etc.

In addition, this article also introduces several main algorithm to improve the direction, including improved method based on parameter adjustment, with contraction factor improvement methods, for discrete optimization problems of two typical version of particle swarm optimization algorithm, based on the genetic improvement strategy of ideas and gradient information, and the algorithm in constrained optimization and multi-objective optimization solutions of two classes of complex environment.

Finally, the factors influencing the particle swarm optimization algorithm are explored through experiments. The influence of iteration number and population size on the convergence precision of the algorithm was studied by changing the parameters. Through the simulation of 8 classical test functions (Ackley, Rastrigin, Griewank, Alpine, Sphere, Rosenbrock, Schwefel P2.22, Sum of Different Power), the standard particle swarm optimization algorithm and its improved algorithms (ldiw-PSO, APSO, breed-PSO, BBPSO, CLPSO) were compared, and the performance of Different algorithms in Different functions was statistically calculated. The characteristics of particle swarm optimization (PSO) in parameter improvement, algorithm mixing, learning operator improvement and topology improvement are obtained.

Acknowledgments

This paper is final project of my Professional English course. I'd like to give my gratitude to teacher Xiaojie Li, one of the most dedicated teachers i've ever met. Also, this paper was first finished during my last winter vacation and written in Chinese, which is the first time to do a review study by myself, highly promoting my research ability. Thanks to all researchers working on PSO, devoting themselves to such a elegant algorithm.

References

[1] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Mhs95 Sixth International Symposium on Micro Machine & Human Science*, 2002.

[2] Russell C Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. In *Congress on Evolutionary Computation*, 2002.

[3] Jiawei Han, Pei Jian, and Yiwen Yin. Mining frequent patterns without candidate generation. 2000.

[4] Umair Farooq Siddiqi, Yoichi Shiraishi, and Sadiq M. Sait. Multi-constrained route optimization for electric vehicles (evs) using particle swarm optimization (pso). In *International Conference on Intelligent Systems Design & Applications*, 2012.

[5] Chia Feng Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society*, 34(2):997–1006, 2004.

[6] Jiansheng Wu, Long Jin, and Mingzhe Liu. Evolving rbf neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm. *Neurocomputing*, 148(2):136–142, 2015.

[7] A. S. Elwer, S. A. Wahsh, M. O. Khalil, and A. M. Nur-Eldeen. Intelligent fuzzy controller using particle swarm optimization for control of permanent magnet synchronous motor for electric vehicle. In *Conference of the IEEE Industrial Electronics Society*, 2003.

[8] K. Kumarasamy and R. Raghavan. Particle swarm optimization algorithm for voltage stability improvement using multiple statcom. In *International Conference on Emerging Trends in Electrical Engineering & Energy Management*, 2013.

[9] Rudra Pratap Singh, V. Mukherjee, and S. P. Ghoshal. Particle swarm optimization with an aging leader and challengers algorithm for optimal power flow problem with facts devices. *International Journal of Electrical Power & Energy Systems*, 64:1185–1196, 2015.

[10] M. A. Abido. Multiobjective particle swarm optimization for optimal power flow problem. In *Power System Conference*, 2008.

[11] A. B Kusumaningtyas, M. N Hidayat, and F Ronilaya. Implementation of particle swarm optimization method for voltage stability analysis in 150 kv sub system grati – paiton east java. In *Materials Science & Engineering Conference Series*, 2018.

[12] Y Fukuyama and H Yoshida. A particle swarm optimization for reactive power and voltage control in electric power systems. In *Conference on Genetic & Evolutionary Computation*, 1999.

[13] Liu Jia, Li Yang, and Liqun Gao. Particle swarm optimization algorithm in power line overhaul system control. In *Control & Decision Conference*, 2008.

[14] Fang Yao, Zhao Yang Dong, Ke Meng, Zhao Xu, Ho Ching Iu, and Kit Po Wong. Quantum-inspired particle swarm optimization for power system operations considering wind power uncertainty and carbon tax in australia. *IEEE Transactions on Industrial Informatics*, 8(4):880–888, 2012.

[15] Pinaki Mitra and Ganesh K. Venayagamoorthy. A dstatcom controller tuned by particle swarm optimization for an electric ship power system. In *Power & Energy Society General Meeting-conversion & Delivery of Electrical Energy in the Century*, 2013.

[16] Jianli Shi, Zhang Jin, Kun Wang, and Fang Xin. Particle swarm optimization for split delivery vehicle routing problem. *Asia-Pacific Journal of Operational Research*, (49):1840006, 2018.

[17] Y. E Demirtas, E Ozdemir, and U Demirtas. A particle swarm optimization for the dynamic vehicle routing problem. In *International Conference on Modeling*, 2015.

[18] Ramadan Abdelaziz and Mauricio Zambrano-Bigiarini. Particle swarm optimization for inverse modeling of solute transport in fractured gneiss aquifer. *Journal of Contaminant Hydrology*, 164(4):285–298, 2014.

[19] Wang Hua, Xiangxu Meng, Li Shuai, and Xu Hong. A tree-based particle swarm optimization for multicast routing. *Computer Networks*, 54(15):2775–2786, 2010.

[20] Amin Zargar Nasrollahy and Hamid Haj Seyyed Javadi. Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target. In *Third Uksim European Symposium on Computer Modeling & Simulation*, 2009.

[21] Hongwei Mo and Lifang Xu. Research of biogeography particle swarm optimization for robot path planning. *Neurocomputing*, 148(148):91–99, 2015.

[22] Zheng Bo, Yanfeng Li, and Guozhong Fu. Discretization algorithm based on particle swarm optimization and its application in attributes reduction for fault data. *Journal of Shanghai Jiaotong University*, 23(4):1–5, 2018.

[23] Orlando Durán, Nibaldo Rodriguez, and Luiz Airton Consalter. Collaborative particle swarm optimization with a data mining technique for manufacturing cell design. *Expert Systems with Applications*, 37(2):1563–1567, 2010.

[24] Andres Iglesias, Karin Ljubič Fister, Salahuddin M Kamal, Iztok Fister, Iztok Fister, and Matjaž Perc. Particle swarm optimization for automatic creation of complex graphic characters. *Chaos Solitons & Fractals the Interdisciplinary Journal of Nonlinear Science & Nonequilibrium & Complex Phenomena*, 73(4):29–35, 2015.

[25] Hai Hao Li, Yu Wen Fu, Zhi Hui Zhan, and Jing Jing Li. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling. In *Evolutionary Computation*, 2015.

[26] M. Mutingi and C. Mbohwa. A fuzzy particle swarm optimization approach for task assignment in home health care. In *IEEE International Conference on Industrial Engineering & Engineering Management*, 2014.

[27] Wenzhong, Rongrong, Yuzhen, Guolong, and CHEN. Xgrouter: high-quality global router in x-architecture with particle swarm optimization. *Frontiers of Computer Science*, 9(4):576–594, 2015.

[28] Marciel Barros Pereira and Tarcisio Ferreira Maciel. Particle swarm optimization for base station placement. In *Telecommunications Symposium*, 2014.

[29] Nanbo Jin and Yahya Rahmat-Samii. Advances in particle swarm optimization for antenna designs: Real-number, binary, single-objective and multiobjective implementations. *IEEE Transactions on Antennas & Propagation*, 55(3):556–567, 2007.

[30] Edson R. Schlosser, Sabrina M. Tolfo, and Marcos V. T. Heckler. Particle swarm optimization for antenna arrays synthesis. In *Microwave & Optoelectronics Conference*, 2016.

[31] Amir Mahyar Khorasani, Mohsen Asadnia, and Pooneh Saadatkia. Modeling of tic-n thin film coating process on drills using particle swarm optimization algorithm. *Arabian Journal for Science & Engineering*, 38(6):1565–1571, 2013.

[32] Y. H. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998.

[33] Yanxia, Qi, Guoyuan, Wang, Zenghui, Van Wyk, Barend Jacobus, Hamam, and Yskandar. Chaotic particle swarm optimization. In *Congress on Evolutionary Computation*, 2002.

[34] Xiaojun Wu, Wang Ying, and Tiantian Zhang. An improved gapso hybrid programming algorithm. In *International Conference on Information Engineering & Computer Science*, 2009.

[35] Fei Han. An improved particle swarm optimization algorithm based on gradient search is proposed. *Journal of Nanjing University*, 2013.

[36] Cédric Leboucher, Patrick Siarry, Stéphane Le Ménec, Hyo Sang Shin, Rachid Chelouah, and Antonios Tsourdos. An enhanced particle swarm optimisation algorithm combined with neural networks to decrease computational time. In *International Conference on Swarm Intelligence Based Optimization Icsibo*, 2018.

[37] Y. J. Gong, J. J. Li, Y. Zhou, Y. Li, H. S. Chung, Y. H. Shi, and J. Zhang. Genetic learning particle swarm optimization. *IEEE Transactions on Cybernetics*, 46(10):2277–2290, 2017.

[38] Deming Lei. A pareto archive particle swarm optimization for multi-objective job shop scheduling. *Computers & Industrial Engineering*, 54(4):960–971, 2008.

[39] Ahmad Nickabadi, Mohammad Mehdi Ebadzadeh, and Reza Safabakhsh. A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing Journal*, 11(4):3658–3670, 2011.

[40] Jianghong Han. A adaptive particle swarm optimization and its simulation study. 2006.

[41] Jianbin Xin, Guimin Chen, and Yubao Hai. A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In *International Joint Conference on Computational Sciences & Optimization*, 2009.

[42] E. S. Peer, F. Van Den Bergh, and A. P. Engelbrecht. Using neighbourhoods with the guaranteed convergence pso. In *Swarm Intelligence Symposium*, 2003.

[43] Asanga Ratnaweera, Saman K Halgamuge, and Harry C Watson. *Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients*. 2004.

[44] M. Clerc and J. Kennedy. *The particle swarm - explosion, stability, and convergence in a multidimensional complex space*. 2002.

[45] Eberhart R C. Kennedy J. A discrete binary version of the particle swarm algorithm. *IEEE*, 2002.

[46] Eberhart R.C. Hu, X. Solving constrained nonlinear optimization problems with particle swarm optimization. *The 6th world multiconference on systemics,cybernetics and informatics*, 2002.

[47] P. J. Angeline. Using selection to improve particle swarm optimization. In *IEEE World Congress on IEEE International Conference on Evolutionary Computation*, 2002.

[48] N Higashi and H Iba. Particle swarm optimization with gaussian mutation. In *Swarm Intelligence Symposium*, 2013.

[49] Amaresh Sahu, Sushanta Kumar Panigrahi, Sabyasachi Pattnaik, J. Protcy, Adaptive Weight, Constriction Factor, and Particle Mean Dimension. Fast convergence particle swarm optimization for functions optimization. *Procedia Technology*, 4(11):319–324, 2012.

[50] Junwei Wang. Improved and application of particle swarm optimization algorithm. 2006.

[51] K. E Parsopoulos, V. P Plagianakos, G. D Magoulas, and M. N Vrahatis. *Improving the Particle Swarm Optimizer by Function "Stretching"*. 2001.

[52] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[53] A. El-Gallad, M. El-Hawary, A. Sallam, and A. Kalas. Enhancing the particle swarm optimizer via proper parameters selection. In *IEEE Ccece Canadian Conference on Electrical & Computer Engineering*, 2002.

[54] Emilio F. Campana, Giovanni Fasano, and Antonio Pinto. Dynamic analysis for the selection of parameters and initial population, in particle swarm optimization. *Journal of Global Optimization*, 48(3):347–397, 2010.

[55] QU, Y B., SUGANTHAN, N P., LIANG, and J J. Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, 16(5):601–614, 2012.

[56] James Kennedy. Bare bones particle swarms. In *Swarm Intelligence Symposium*, 2003.

[57] Mauro Campos and Renato A. Krohling. Hierarchical bare bones particle swarm for solving constrained optimization problems. In *Evolutionary Computation*, 2013.

[58] Steven J. Gunn. A hybrid particle swarm optimization (pso)-simplex algorithm for damage identification of delaminated beams. *Mathematical Problems in Engineering,2012,(2012-11-7)*, 2012(2012):1239–1257, 2012.

[59] V. L. Huang, P. N. Suganthan, and J. J. Liang. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. *International Journal of Intelligent Systems*, 21(2):209–226, 2006.

[60] Zhang Yong, Dunwei Gong, Hu Ying, and Wanqiu Zhang. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing*, 148(1):150–157, 2015.

[61] A. Koshti and S. C. Choube. Voltage stability constrained distributed generation planning using modified bare bones particle swarm optimization. *Journal of the Institution of Engineers*, 94(2):123–133, 2013.