*International Journal of Innovative Computing, Information and Control*
Volume **16**, Number **4**, August **2020**

ICIC International ©2020 ISSN 1349-4198
pp. 1203–1223

# TWO STRATEGY COOPERATIVE PARTICLE SWARM OPTIMIZATION ALGORITHM WITH INDEPENDENT PARAMETER ADJUSTMENT AND ITS APPLICATION

Qiwen Zhang[1], Yachen Wei[1] and Weiyang Song[2]

[1]School of Computer and Communication Technology
Lanzhou University of Technology
No. 287, Langongping Road, Qilihe District, Lanzhou 730050, P. R. China
tdyy2010@126.com; 278316262@qq.com

[2]The High School Attached to Northwest Normal University
No. 21, Shilidian South Street, Anning District, Lanzhou 730070, P. R. China
8230270@qq.com

Received February 2020; revised June 2020

Abstract. *Aiming at the premature convergence of particle swarm optimization (PSO) in solving complex multimodal problems, a two strategy cooperative particle swarm optimization algorithm (TSPSO) with independent parameter adjustment is proposed. In the proposed algorithm, the variance of population fitness, evolution ability and evolution rate of particles are defined firstly, and then the inertia weight and learning factor of each particle are adjusted adaptively, which effectively balances the exploitation and exploration ability of the algorithm. Secondly, according to the fitness value and evolution ability of particles in each generation, the population is divided dynamically. The inferior subgroup uses reconstruction strategy to generate new particles by learning from the particles in the superior subgroup, so as to speed up the convergence of the algorithm; the superior subgroup uses differential mutation to avoid the premature convergence of the algorithm, and maintain the diversity of the population. Finally, a large number of experiments are carried out on the CEC2013 standard test function set and flexible job shop scheduling problem, and the experimental results verify that TSPSO has high efficiency. The convergence analysis shows the effectiveness of the algorithm.*

**Keywords:** Particle swarm optimization, Parameter adjustment, Reconstruction, Differential mutation, Flexible job shop scheduling

## 1. Introduction.

Particle swarm optimization algorithm is an intelligent optimization algorithm proposed by Eberhart and Kennedy in 1995 to imitate the behavior of bird population [1]. There are many factors that affect the convergence performance of PSO, such as population size, neighborhood structure, synchronous and asynchronous update mode, especially the control of parameters. The performance of PSO mainly depends on the balance between exploitation and exploration. Over exploration will waste computing resources, and over exploitation will directly lead to premature convergence. In PSO, the reasonable selection of parameters plays an important role in balancing exploitation and exploration of PSO. Typical parameter adjustments include: inertia weight, learning factor, maximum speed, shrinkage factor, etc. For instance, in [2], a shrinkage factor cooperative particle swarm optimization algorithm based on simulated annealing is proposed, which improved the global optimization ability and convergence speed of the algorithm. In [3], particles are distinguished according to the distance from each particle to optimal population, and learning factor and inertia weight are dynamically adjusted according to

DOI: 10.24507/ijicic.16.04.1203

the distance. Experiments show that the algorithm is effective and accurate in the process of optimization. [4] proposed an adaptive method to change the inertia weight between linear and nonlinear, so that the algorithm has a better ability to balance exploitation and exploration. According to the evolutionary characteristics of each particle, [5] adjusts the value of parameters, and uses restart strategy to update the corresponding particles which has advantages in convergence, robustness and scalability. In order to improve the convergence accuracy and speed of the algorithm in multi-dimensional complex functions, an improved particle swarm optimization algorithm based on dynamic learning factor is proposed in [6], and the learning factors change with the change of the search phase. A new adaptive inertia weight adjustment strategy is proposed in [7]. The algorithm uses Bayesian technology to adjust the inertia weight according to the particle position, and uses Cauchy mutation to find a better solution. In addition, the inertia weight is determined according to the particle's performance and the distance from the best position [8], or it is set as a linear function which introduces a time factor [9], and the learning factors are adjusted according to the change of inertia weight adaptively.

The improved PSO algorithm has many advantages, and the parameter setting mechanism of most algorithms can improve the performance of the algorithm. It is difficult to balance the global search ability and local search ability of the algorithm only by adjusting the parameters, especially when dealing with high-dimensional complex optimization problems, particle swarm optimization algorithm is easy to fall into local optimization, and the convergence speed of the algorithm in the late iteration will be very slow. Therefore, [10] proposed a distance based local information particle swarm optimization algorithm, which guides particles to find the optimal solution in the search space through the historical optimal position of adjacent particles, and improves the local search ability of the algorithm. A comprehensive learning strategy is proposed in [11] which uses the historical best position of particles to update the speed of particles, maintain the diversity of the population, and prevent the algorithm from premature convergence. Self regulating particle swarm optimization algorithm adopts two learning strategies: self regulating the inertia weight of the optimal particle and the perception of the rest particles to the global optimal position to determine the search direction [12], so as to improve the convergence ability of the algorithm. [13] proposed a double differential mutation particle swarm optimization algorithm, which divides the population into two subgroups for different differential mutation operations, and the global search ability and efficiency of the algorithm are improved obviously.

PSO is studied from different perspectives, and the ultimate goal is to solve practical optimization problems [14-22]. The research content of optimization problem is to select the optimal solution in a wide range of solutions [23,24]. Flexible job shop scheduling problem (FJSP) [25] is a highly important research topic in the field of combinatorial optimization and production processing, which meets the actual requirements of workshop production scheduling. In recent years, intelligent algorithm has become the main method to solve FJSP problems, such as genetic algorithms (GA) [26,27], particle swarm optimization (PSO) [28,29], artificial bee colony algorithm (ABC) [30], tabu search (TS) [31], migrating birds optimization (MBO) [32], differential evolution (DE) [33-36], harmony search algorithm (HS) [37], and procured preferable effect. PSO is suitable for scientific research and engineering application due to its advantages of few parameters, easy to implement and mature theoretical background. In order to overcome the shortcomings of traditional algorithms in solving the FJSP problem, [38] proposed an improved particle swarm optimization algorithm with decreasing perturbation index, and used neighborhood search as a local search method to solve multi-objective job shop scheduling problem. [39] proposed the quantum particle swarm optimization algorithm, which uses mutation operator to avoid the algorithm falling into the local optimum. [40] proposed a hybrid algorithm for solving multi-objective FJSP by combining particle swarm optimization algorithm with tabu search algorithm. [41] proposed a hierarchical method to solve the two sub problems of FJSP with particle swarm optimization algorithm and random restart mountain climbing algorithm. The global search ability and local optimization ability of the hybrid algorithm have been effectively improved, and the advantages are obvious in solving FJSP problems.

In view of the premature convergence of particle swarm optimization algorithm and the lack of dynamic adaptability in the evolution process, a two strategy cooperative particle swarm optimization algorithm (TSPSO) is proposed in this paper. In TSPSO, the evolution rate of each particle is calculated according to the variance of population fitness and the evolution ability of independent particles. In each iteration, the evolution parameters of particles are adjusted according to the dynamic change of evolution rate, and the exploration stage and exploitation stage are effectively balanced. At the same time, according to the fitness value and evolution ability of particles, the population is divided into dynamic subgroups, and different strategies are adopted to help the evolution of different subgroups, so as to effectively avoid the algorithm falling into local optimum, enhance the diversity of population, and improve the convergence rate of the algorithm.

The rest of the paper is organized as follows. Section 2 reviews the background of PSO. In Section 3, we illustrate the proposed algorithm. In Section 4, we apply the algorithm to solving the flexible job shop scheduling problem. After that, Section 5 gives the experimental results and analysis of the algorithm in continuous and discrete problems. Section 6 is the summary of the paper, and points out the direction of the next step.

## 2. Particle Swarm Optimization.

In 1998, Shi and Eberhart [42] introduced inertia weight into traditional particle swarm optimization algorithm. In PSO, every individual without mass and volume is regarded as a particle. In the search space of D-dimensional, the fitness value of each particle is calculated according to the objective function, which is used to measure the position of particles. The motion direction and distance of particles are determined by the velocity vector $V$. The flight speed of particles is dynamically adjusted by the historical experience of particles and population. The formula for updating the speed and position of particles is as follows:

$$V_i^{t+1} = \omega * V_i^t + c_1 * r_1 * \left(P_{best} - X_i^t\right) + c_2 * r_2 * \left(G_{best} - X_i^t\right) \tag{1}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2}$$

where $t$ is the current number of iterations, and $r_1$, $r_2$ are the random numbers between $[0, 1]$, which are used to maintain the diversity of the population. $P_{best}$ is the historical optimal position of particle, and $G_{best}$ is the historical optimal position of population. $c_1$, $c_2$ are the learning factors, indicating the influence of particle history experience and population history experience on particle motion behavior. $\omega$ is called inertia weight and generally takes the number between $(0, 1)$. It is used to express the influence of the speed of particles in the previous generation on the current particle motion behavior. Currently, the setting method of inertia weight with more usage is that it decreases linearly with the increase of the number of iterations:

$$\omega = \omega_{init} - (\omega_{init} - \omega_{end}) * (t/T) \tag{3}$$

where $\omega_{init}$ is the initial weight, $\omega_{end}$ is the final weight, and $T$ is the maximum number of iterations. When $\omega$ is large, the speed of particle is fast, which is similar to the global search method, and it can always find new areas, showing a divergent state. Of course, the algorithm will need more iterations to find the global optimum. When $\omega$ is small, the particle will conduct fine search near the optimal solution. If the global optimal solution happens to be within the search range, the particle will find the global optimal solution quickly, otherwise it will not be able to find the global optimal solution. Therefore, it is generally set as $\omega_{init} = 0.9$, $\omega_{end} = 0.4$ to ensure that the value of $\omega$ is between 0.4-0.9. When $t = 1$, $\omega_{init} = 0.9$; when $t = T$, $\omega_{end} = 0.4$. $\omega$ decreases linearly with the increase of the number of iterations, which makes the algorithm have a larger value of $\omega$ tend to global search at the beginning of the iteration. After traversing the whole search space rapidly, the local search ability is constantly enhanced, and the convergence performance of the algorithm is improved. At the same time, it avoids the trouble that too small $\omega$ will cause particles to fall into local optimum.

Particle swarm optimization (PSO) has become a research hotspot of scientific theory rapidly due to its advantages of few parameters, easy to implement, and has been widely used to solve various engineering problems. However, PSO still has the problems of premature convergence and slow convergence speed, especially in solving complex multimodal function and large-scale function optimization problems, the stability and accuracy of the algorithm cannot be guaranteed. There are many reasons for the shortcomings of PSO algorithm, mainly due to the improper selection of parameters and optimization strategies. At present, most particle swarm optimization algorithms are based on population when adjusting parameters, and do not consider the differences between particles and the evolution of population. In this case, it is difficult to maintain the population diversity in the later stage of the iteration, and the particles are basically incapable of continuous optimization. Therefore, particles search in a small range near the local optimum, and it is difficult to jump out of the search area, so the algorithm is prone to premature convergence. Furthermore, a single evolutionary strategy cannot be applied to all particles with different performances in the population. In view of the fact that the diversity of particle swarm optimization is difficult to maintain, it is easy to be precocious, and poor robustness, this paper proposes a two strategy cooperative PSO algorithm with independent adjustment of parameters.

## 3. The Proposed TSPSO Algorithm.

### 3.1. Determination of population evolution state.

In the process of evolution, with the increase of the number of iterations, the particles gradually show a “gathering” state. The basic reason that the algorithm is easy to fall into the local optimal is that the difference between individuals is getting smaller and smaller. Therefore, this paper introduces the variance of population fitness to monitor the evolutionary state of each generation in the evolutionary process. The idea of this method benefits from the concept of variance in mathematical knowledge. Variance is used to measure the degree of deviation between random variables and their mathematical expectations. The variance of population fitness is the average of the sum of squares of the difference between the fitness value of each particle and the average fitness value of the particle in the population, which reflects the aggregation degree of the particles in the population, so as to determine the current evolution state of the population.

**Definition 3.1.** *The total number of particles in the population is $N$, $f(X_i)$ is the fitness value of the i-th particle, $f_a(X)$ is the average of the current fitness value of the population, and the calculation formula of the current fitness variance of the population is as follows:*

$$\delta^2 = \frac{1}{N}\sum_{i=1}^{N}(f(X_i) - f_a(X))^2 \tag{4}$$

**Theorem 3.1.** *If the algorithm converges to the global optimum or falls into the local optimal, then $\delta^2 = 0$.*

From Theorem 3.1, when $\delta^2 = 0$, it can be judged that PSO is in the state of convergence or precocity. When $\delta^2 > 0$, if the value of $\delta^2$ is far greater than 0, it means that the aggregation degree of particles in the population is not obvious, and the evolution state of the population is in the stage of random exploration, that is to say, it is more inclined to do global search. If $\delta^2$ is close to 0, the particles in the population are close to each other. In this evolutionary state, the population has a strong local search ability, but its global search ability is greatly reduced, and it is more prone to premature convergence.

3.2. **The evolutionary ability of particles.** In the evolutionary process, no matter the evolutionary state of the population is in the state of aggregation or dispersion, the evolutionary ability of different particles in the population is different from each other. Each particle has a varied position in the search space, and their ability to approach the global optimal solution is different.

**Definition 3.2.** *In the process of evolution, the evolutionary ability of particles is defined as the ability to find the optimal solution of individuals. The evolutionary ability of particle $i$ in generation $t$ is calculated as follows:*

$$E_i(t) = |f_{pi}(t) - f_{pi}(t-1)| \tag{5}$$

*where $f_{pi}(t)$ represents the individual optimal fitness value of the $i$-th particle in the $t$ generation. The value of $E_i(t)$ is always greater than or equal to zero. When $E_i(t) = 0$, it means that the particle cannot find a better solution than the current individual optimal value after the $t$-th iteration, and the individual optimal value has not been updated. On the contrary, it indicates that the particles have the ability to evolve in the $t$-th iteration, and the individual optimal value is updated, at this time, $E_i(t) > 0$. From Equation (5), the larger the difference between $f_{pi}(t)$ and $f_{pi}(t-1)$, the greater the value of $E_i(t)$, and the stronger the evolutionary ability of particles.*

3.3. **Parameter adjustment.** In PSO, the convergence performance of the algorithm depends on whether it can effectively balance the global and local search capabilities. The value of the parameter directly affects the global search and local search abilities of the algorithm. This paper redefines the concept of particle evolution rate according to the evolutionary state of population and particle evolution ability, and adjusts the parameters according to the change of evolution rate. This method of adaptive parameter adjustment can effectively improve the particle search efficiency.

**Definition 3.3.** *In the process of evolution, the evolution rate of particles is defined as the degree of change of particle evolution in the population. The evolution rate of particle $i$ in the $t+1$ generation is calculated as follows:*

$$s_i(t+1) = \frac{1}{\sqrt{\delta^2 + (E_i(t))^2 + 1}} \tag{6}$$

As shown in Equation (6), if the evolution state of the population is dispersion, the evolution rate of particles in the population is generally small, which can better maintain the evolution state of particles in the previous generation; if the evolution state of the population is aggregation, the evolution rate of particles in the population is large, and the evolution state of particles has great changes. When $E_i(t) = 0$, the evolution rate of particle $i$ is higher than that of other particles in the population, and the evolution state of particle has a great change at $t+1$ iteration. In the $t+1$ iteration, the evolution rate is inversely proportional to the evolution ability of particle $i$ in the $t$ iteration. The stronger the evolution ability, the smaller the evolution rate in the next iteration. According to the particle evolution rate, inertia weight and learning factor of each particle in this paper are adjusted as follows:

$$\omega_i(t+1) = \omega_{init} - (\omega_{init} - \omega_{end}) * s_i(t+1) \tag{7}$$

$$c_{1i}(t+1) = c_{1\max} - \frac{c_{1\max} * \sin\left(s_i(t+1) * \frac{\pi}{2}\right) * t}{T} \tag{8}$$

$$c_{2i}(t+1) = c_{2\min} - \frac{c_{2\min} * \sin\left(s_i(t+1) * \frac{\pi}{2}\right) * t}{T} \tag{9}$$

where $c_{1\max}$ is the maximum value of learning factor $c_1$; $c_{2\min}$ is the minimum value of learning factor $c_2$. If the value of $c_1$ is too large and $c_2$ is too small, the particle will be more dependent on the individual's historical experience, and the information sharing between particles can be ignored. At this time, the particle is wandering in the local range, and it is difficult to find the optimal solution. However, when the value of $c_2$ is large, but the value of $c_1$ is small, particles lacking self-cognition will converge to the local optimal advantage prematurely despite the interactive learning between particles. Therefore, this paper sets $c_{1\max} = 2$, $c_{2\min} = 0.8$ to limit the value range of learning factors. The adjusted learning factor can reduce the interference of local optimum and accelerate the convergence speed of the algorithm.

If the current evolution state of the population is determined to be decentralized, the diversity of the population is good enough, and the particles in the population have strong random exploration ability. In the next iteration, the particles are affected by a smaller evolution rate and keep a larger $\omega$ value to continue the global search. On the contrary, if the evolution state of population changes to aggregation state, $\omega$ will decrease with the influence of larger evolution rate, and change from global search to local search. In addition, if the particle has a strong evolutionary ability, the evolution rate of the particle in the next iteration will be small, and the value of $\omega$ will be slightly affected by the evolution rate compared with those of other particles in the same generation. Meanwhile, the larger $c_1$ and smaller $c_2$ are more favorable for the particles to do global search. If the evolution ability of the particle is weak, the value of $\omega$ is slightly affected by the larger evolution rate compared with that of other particles in the population, and smaller $c_1$ and larger $c_2$ make the particle more inclined to do local search. In a word, the evolution state of the current population determines the search range of the particles in the population, and the evolution ability of the particles is the key to adjust the parameters independently.

3.4. **The mechanism of two strategy cooperation.** In order to keep the excellent genes of the particles in the population, ensure the diversity of the population, and make the particles have the ability to jump out of the local optimum, this paper proposes a two strategy coprocess evolution mechanism, and its main operations are as follows.

1) Dynamic division of population

The particles in the population are sorted according to the order of fitness value from small to large and the order of evolution ability from strong to weak. The sorted sequence is called array1 and array2. The particles existing in both the first half of array1 and array2 are put into the superior subgroup1. These particles have good fitness value and strong evolutionary ability. Similarly, the particles existing in both the second half of array1 and array2 are put into subgroup2, which have poor fitness and weak evolutionary ability. Finally, the remaining particles in the population are put into the common subgroup3.

As shown in Figure 1, assume that the total number of particles in the population is 10, and the corresponding numbers of particles are 1 to 10. The 10 particles are sequenced according to fitness and evolution ability, and the results of sequence are as follows: array1

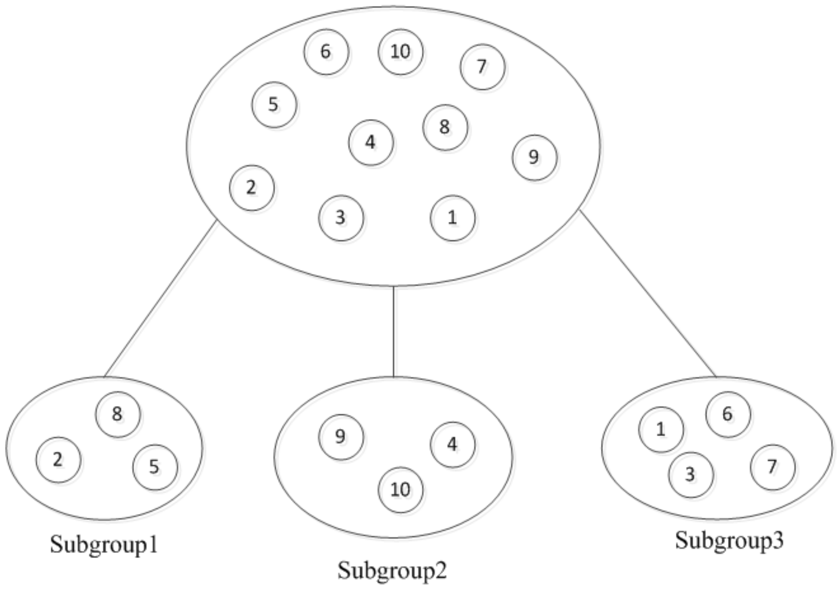


Figure 1. Dynamic population division diagram

$= [8, 2, 5, 7, 1, 6, 10, 4, 3, 9]$, array2 $= [3, 5, 6, 8, 2, 7, 1, 10, 9, 4]$. The particles 2, 8, 5 with better fitness value and strong evolutionary ability are put into subgroup1; the particles 4, 9, 10 with poor fitness value and weak evolutionary ability are put into subgroup2, and the remaining particles are put into subgroup3.

2) Reconstruction

The particles in subgroup2 not only fail to converge quickly, but also easily fall into local optimum and difficult to jump out. Therefore, reconstruction strategy proposed in this paper makes the particles in subgroup2 reconstruct new particles by learning from the particles in subgroup1. The generation of new particles increases the diversity of population and contributes to the rapid convergence of the population. All particles in subgroup1 are likely to be the examples that need to be learned during particle reconstruction in subgroup2. Before reconstruction, a random number ($p_{i,d}$) is generated for each dimension of the particle. If $p_{i,d}$ is greater than the learning probability $P_l$ ($P_l = 0.8$), then $x_{i,d}$ learns from the $d$-th dimension of the global optimal particle; if $p_{i,d}$ is less than the learning probability $P_l$ and greater than the adjustment probability $P_a$ ($P_a = 0.4$), then $x_{i,d}$ randomly selects a particle $k$ from subgroup1 to learn in its $d$-th dimension; if $p_{i,d}$ is less than the adjustment probability $P_a$, then $x_{i,d}$ remains unchanged. The reconstruction formula of particle position is as follows:

$$x_{i,d} = \begin{cases} x_{gbest,d}, & \text{if } p_{i,d} < P_l \\ x_{k,d}, & \text{if } P_a < p_{i,d} < P_l \\ x_{i,d}, & \text{if } p_{i,d} < P_a \end{cases} \tag{10}$$

It can be seen from Equation (10), particles are reconstructed by learning from superior particles, and the selection of superior particles depends on the random number $p_{i,d}$.

3) Differential mutation

Particles in subgroup2 are reconstructed by learning from particles in subgroup1, not only preserving the genes of some superior particles, but also greatly improving the performance of the inferior particles. For the sake of ensuring the convergence performance of the algorithm and avoiding falling into local optimal, the particles in subgroup1 will be operated by differential mutation.

The evolutionary goals of the algorithm are different in different evolutionary stages. In the early stage of the algorithm iteration, the diversity of the population should be maintained, the search range should be expanded, and more optimal solutions should be found as far as possible. In the late stage of the algorithm iteration, the convergence ability of the population should be increased and the accuracy of the population to find the optimal solution should be improved. Therefore, this paper divides two stages according to the number of iterations, and then adopts a two-stage differential mutation strategy:

$$x_i^{mut} = \begin{cases} x_{r_1} + F \cdot (x_{r_2} - x_{r_3}), & t \le \dfrac{T}{2} \\ x_i + F \cdot |x_{best} - x_i| + F \cdot |x_{r_1} - x_{r_2}|, & t > \dfrac{T}{2} \end{cases} \tag{11}$$

where $x_{best}$ represents the optimal particle in the current population, $r_1$, $r_2$, $r_3$ are the indexes of the selected particles. $F$ is the scale factor, usually taken between the values of $[0, 1]$, and its value directly affects the interference degree of the disturbance vector to the initial vector.

The difference of mutation strategy is reflected in the difference of initial vector and disturbance vector. As shown in Equation (11), when the number of evolutions does not exceed $T/2$, three different individuals are randomly selected in the population. The vector difference between two individuals is calculated, then multiplied by the scale factor, and finally added with another individual. Because all the vectors involved in mutation are randomly selected, it can search faster in the global scope and get more optimal solutions. When the evolution number is greater than $T/2$, find the current optimal particle to find global optimal solution. In the later stage of the iteration, particles gradually approach to the optimal value, and the diversity between particles decreases, resulting in smaller disturbance range, thus reducing the search range and achieving rapid population convergence. After the mutation individual $x_i^{mut}$ is generated, it will cross $x_i$. In this paper, the crossing method used in this paper is binomial crossing:

$$x_{i,j}^{cro} = \begin{cases} x_{i,j}^{mut}, & \text{if } rand \le CR \\ x_{i,j}, & \text{else} \end{cases} \tag{12}$$

where $CR$ is the cross factor, and the value is taken from $[0, 1]$. Cross process is a further change to the mutant individual. The new individuals are obtained by mixing the mutated individuals and the target individuals with cross factors. Eventually get a new individual.

After the process of crossover and mutation, in order to get more excellent particles and avoid the destruction of the original particles by the process of crossover and mutation, so compare the particles $x_i^{cro}$ and $x_i$ to select a better particle into the next generation:

$$x_i^{new} = \begin{cases} x_i^{cro}, & \text{if } f(x_i^{cro}) \le f(x_i) \\ x_i, & \text{else} \end{cases} \tag{13}$$

The new subgroup2 after reconstruction, the new subgroup1 after differential mutation and the common subgroup3 are combined to form a new population for the next iteration.

3.5. **Convergence analysis of the algorithm.** In order to make the following analysis of particle swarm optimization more convenient and intuitive, the dimension of particle velocity and position is simplified from $D$ to 1 without losing generality. Let $P_i$ be the best position of particle $i$ and $P_g$ for the best position of the whole population, $\varphi_1 = c_1 * r_1$, $\varphi_2 = c_2 * r_2$, and replace them with the corresponding vectors in Formulas (1) and (2) to obtain the velocity and position updating formula of the simplified PSO algorithm is as follows:

$$v(t+1) = \omega v(t) + \varphi_1(P_i - x(t)) + \varphi_2(P_g - x(t)) \tag{14}$$

$$x(t+1) = x(t) + v(t+1) \tag{15}$$

In order to further simplify the speed and position formula of PSO, let $\varphi = \varphi_1 + \varphi_2$, $p = (\varphi_1 P_i + \varphi_2 P_g)/\varphi$, $y(t) = x(t) - p$, and substitute them into Equations (14) and (15) to obtain the velocity and position formula as:

$$v(t+1) = \omega v(t) - \varphi y(t) \tag{16}$$

$$y(t+1) = \omega v(t) + (1-\varphi) y(t) \tag{17}$$

If $R(t) = \begin{bmatrix} v(t) \\ y(t) \end{bmatrix}$, $A = \begin{bmatrix} \omega & -\varphi \\ \omega & 1-\varphi \end{bmatrix}$, then $R(t+1) = A * R(t)$.

**Definition 3.4.** *If $R^*$ is defined as the equilibrium point of dynamic system $R(t+1) = A * R(t)$, then the equilibrium point $R^*$ must meet the following conditions: $\forall t \geq 0$, $R(t+1) = R(t)$.*

According to Lyapunov's stability, the necessary and sufficient condition for the system to converge to equilibrium is that all eigenvalues of $\boldsymbol{A}$ are less than or equal to 1. $|\boldsymbol{A} - \lambda E| = 0$ is called the characteristic equation of $\boldsymbol{A}$, then the eigenvalue of $\boldsymbol{A}$ is the solution of the characteristic equation $\lambda^2 - (\omega + 1 - \varphi)\lambda + \omega = 0$, so the two eigenvalues of the solution of the linear system $R$ are:

$$\begin{cases} \lambda_1 = \dfrac{\omega + 1 - \varphi + \sqrt{\Delta}}{2} \\ \lambda_2 = \dfrac{\omega + 1 - \varphi - \sqrt{\Delta}}{2} \end{cases} \tag{18}$$

where $\Delta = (\omega + 1 - \varphi^2) - 4\omega$, $\Delta = 0$, $\Delta > 0$, $\Delta < 0$ are discussed respectively, and the convergence domain of system $R(t)$ is obtained as follows:

$$\begin{cases} \omega < 1 \\ \varphi > 0 \\ 2\omega - \varphi + 2 > 0 \end{cases} \tag{19}$$

Therefore, the necessary and sufficient conditions for system convergence are:

$$\begin{cases} \omega < 1 \\ c_1 + c_2 > 0 \\ \omega > (c_1 + c_2)/2 - 1 \end{cases} \tag{20}$$

In this paper, according to the evolution state of population and the evolution ability of particles, each particle is set with independent parameters, in which the value of parameters is: $\omega_{init} = 0.9$, $\omega_{end} = 0.4$, $c_{1\max} = 2$, $c_{2\min} = 0.8$. According to Equations (5)-(9), it can be obtained that:

$$0 \leq s_i(t+1) \leq 1 \tag{21}$$

$$\omega_i(t+1) = 0.9 - 0.5 * s_i(t+1) \tag{22}$$

$$\frac{c_{1i}(t+1) + c_{2i}(t+1)}{2} - 1 = 0.4 - 0.6 * \sin\left(s_i(t+1) * \frac{\pi}{2}\right) * (t/T) \tag{23}$$

where $0.4 \leq \omega_i(t) \leq 0.9$, $0.8 \leq c_{1i}(t) \leq 2$, $0.8 \leq c_{2i}(t) \leq 2$. For each particle, $c_{1i}(t) > 0$, $c_{2i}(t) > 0$, $\omega_i(t) < 1$, $\omega_i(t) > (c_{1i}(t) + c_{2i}(t))/2 - 1$.

It can be concluded that the parameter setting method in this algorithm satisfies the convergence condition of the algorithm, and it is proved that the algorithm has the ability of converging to the local optimal or the global optimal in the search space.

3.6. **The description of the TSPSO algorithm.** The pseudo-code of TSPSO is described in Algorithm 1.

**Algorithm 1** The proposed TSPSO algorithm.

```
1  Inputs: The object function $f(x)$.
2  Initialize: Population size $N$; Initialization parameters $\omega_{init}$, $\omega_{end}$, $c_{1\max}$, $c_{2\min}$; Maximum number of iterations $T$; Dimension size $D$.
3  for $i = 1$ to $N$
4  Initialize position vector of particle $i$ randomly selected in the domain $[x_{\min}, x_{\max}]^D$.
5  Initialize velocity vector of particle $i$ randomly selected in the domain $[-v_{\min}, v_{\max}]^D$.
6  Calculate the fitness value of particles, find out the individual optimal $P_{best}$ and the global optimal $G_{best}$;
7  End for
8  Calculate the variance of population average fitness according to Equation (4).
9  $t = 1$
10 while $t < T$ do
11   for $i = 1$ to $N$
12     Update the velocity and position of particles according to Equations (1) and (2).
13     Calculate the fitness value of particles, and update the individual optimal and global optimal.
14     if $t > 1$
15       Calculate the evolution ability of particles according to Equation (5).
16       Calculate the evolution rate of particle $i$ in the next generation according to Equation (6);
17       Calculate the inertia weight and learning factor of particle $i$ in the next generation according to Equations (7)-(9).
18       According to the fitness value and evolution ability, the particles are sorted and dynamically divided into three subgroups.
19       The particles in inferior subgroup are reconstructed according to Equation (10).
20       The particles in superior subgroup are operated by differential mutation.
21       Combine three subgroups.
22     End if
23   End for
24 t++
25 End while
26 Output: The optimal solution.
```

## 4. **Application of TSPSO in Flexible Job Shop Scheduling Problem.**

4.1. **Description of FJSP problem.** The research of FJSP is that $n$ jobs $J = (J_1, J_2, \ldots, J_n)$ are processed on $m$ machines $M = (M_1, M_2, \ldots, M_m)$. Each job contains one or more processes. Each process can be processed on at least one machine. The processing time of each process varies with the different processing machines. The purpose of scheduling is to reasonably arrange the processing sequence of the process and select the appropriate processing machine for it to meet certain performance indicators. Therefore, FJSP contains two problems: the processing sequence of the process and the processing machine of the process. The main constraints of FJSP are as follows.

1) The constraint of job processing sequence, which constrains the processing sequence of each job.

$$s_{ih} + x_{kih} \times p_{kih} \leq c_{ih} \tag{24}$$

$$c_{ih} \leq s_i(h+1) \tag{25}$$

where $i$ represents process index ($i = 1, 2, \ldots, n$); $h$ represents total number of job $J_i$ ($h = 1, 2, \ldots, h_j$); $k$ represents machine index ($k = 1, 2, \ldots, m$); $p_{kih}$ represents processing time of the $h$-th process of job $J_i$ on machine $k$.

2) The constraint of the job completion time, which restricts the completion time of each job no more than the total completion time.

$$c_{ih_j} \leq C_{\max} \tag{26}$$

where $C_{\max}$ represents maximum completion time.

3) Machine constraint, which means that the same process can only be processed by one machine at the same time.

$$\sum_{i=1}^{m_{i,h}} x_{kih} = 1 \tag{27}$$

where $m_{i,h}$ represents the number of machines that can be selected in the $h$-th process of job $J_i$; $x_{kih}$ indicates whether machine $k$ is selected for processing.

4) Parameter constraints, each parametric variable must be a positive number.

$$s_{ih} \geq 0 \tag{28}$$

$$c_{ih} \geq 0 \tag{29}$$

where $s_{ih}$ represents the start time of the $h$-th process of $J_i$; $c_{ih}$ represents the end time of the $h$-th process of $J_i$.

When solving FJSP problems, the scheduling scheme is judged by different optimization objectives. The evaluation index adopted in this paper is to minimize the maximum completion time, which refers to the completion time of all processes allocated by each machine. The maximum completion time reflects the production efficiency of the scheduling system, which is the most direct basis for judging the performance of the scheduling system. Therefore, it has been one of the most widely used evaluation indexes in the application research of FJSP, which can be expressed as follows:

$$f = \min\left(\max_{1\leq i\leq n} C_i\right) \tag{30}$$

4.2. **Encoding and decoding.** The particle encoding consists of two parts: one is based on the process encoding, which is used to determine the procedure processing sequence of the job; the other is based on the machine encoding, which is used to assign processing machines to each procedure. In this paper, two vectors of $L$ ($L$ is the total number of procedures for the job) dimension are used to represent a particle, and they are procedure vector $X_P[L]$ and machine vector $X_M[L]$, which constitute a scheduling scheme.

As shown in Figure 2, it is a feasible code. The first line represents the process vector, each element is an integer at $(1, 2, \ldots, n)$, $n$ is the number of jobs. The first element 2 represents job $J_2$, and the second element 1 represents job $J_1$. The first occurrence of 2 indicates the first process of job $J_2$, and the second occurrence of 2 indicates the second process of job $J_2$. The second line is the machine vector. The elements in the vector correspond to the serial number of the machine in the machine set where the process is executed. The value range is $(1, 2, \ldots, m)$, $m$ is the number of machines. The first element 1 indicates that the first process of job $J_2$ is processed on the first machine in the executable machine set.

| $X_P[L]$ | 2 | 1 | 3 | 2 | 1 | 1 | 2 | 3 | 3 |
|---|---|---|---|---|---|---|---|---|---|
| $X_M[L]$ | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 2 |

FIGURE 2. Code of particles

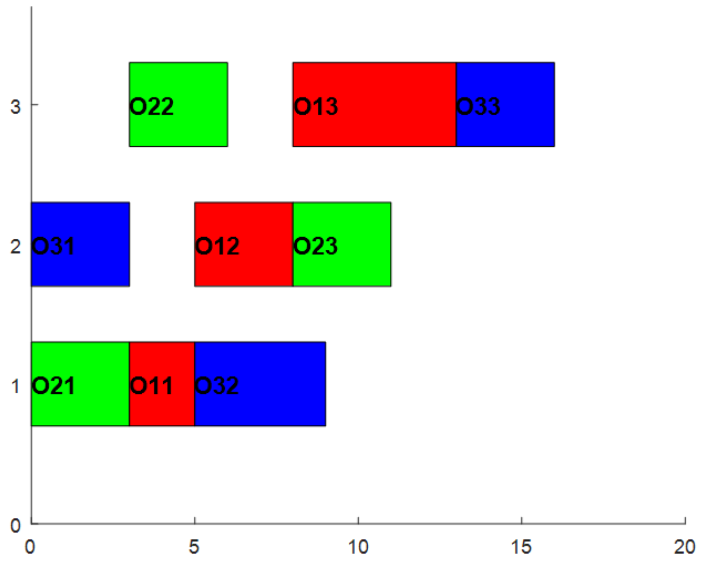


FIGURE 3. Gantt chart

Decoding is the process of converting these two vectors into a specific scheduling scheme. First, determine the execution machine of each process according to the machine vector, and then determine the sequence of the processes on each machine according to the process vector, so as to determine the specific scheduling scheme. Combined with the process vector and the machine vector, the corresponding process on machine 1 after decoding is $O_{21} \rightarrow O_{11} \rightarrow O_{32}$, the corresponding process on machine 2 is: $O_{31} \rightarrow O_{12} \rightarrow O_{23}$, and the corresponding process on machine 3 is: $O_{22} \rightarrow O_{13} \rightarrow O_{33}$. The Gantt chart is shown in Figure 3.

4.3. **Initialization.** Population initialization is a key part of evolutionary algorithm. The quality of initialization particles directly affects the speed and accuracy of the algorithm. First, an initial process sequence array is generated according to the number of jobs and the number of processes of jobs, for example, array $= [1, 1, 1, 2, 2, 2, 3, 3, 3]$. Then, according to the population size $N$, the spatial dimension (i.e., the total number of processes $h$), all particles are randomly generated. The initial sequence of processes corresponds to each particle one by one, and according to the ascending order of the particle dimensions to make corresponding changes, thus a process sequence is generated. Most of the literature in solving FJSP problems are using the method of random initialization, random initialization produces individuals with obvious randomness, and the location of particles cannot be estimated. In this paper, the inverse particles are considered at the same time when initializing. Both particles and the inverse particles have the same possibility of approaching the optimal solution. Then, if a better particle is selected as the initial particle, the convergence speed of the algorithm will be guaranteed.

4.4. **Update mechanism of position and velocity vector.** The velocity vector of each particle is expressed as $V_P[L]$ and $V_M[L]$, calculated according to Equation (1); the position vector is expressed as $X_P[L]$ and $X_M[L]$, calculated according to Equation (2).

After the update iteration, particles may not be a feasible scheduling scheme. Therefore, some corrections need to be made during the whole update process, as follows.

1) The element value range of vector $X_M[L]$ is $[1, m]$. The result in Figure 4 is obtained after the vector is updated according to Equation (2). If the value is no longer a feasible solution, the upward rounding method is used to correct it, and if the corrected value exceeds the value interval of $[1, m]$ or if the process cannot be executed on the machine, an executable machine is randomly selected to execute the process, as shown in the bold body in the figure.

| **Initial value** | **1** | **1** | **2** | **3** | **2** | **3** | **2** | **1** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| **Updated value** | 1.2 | 1.8 | 2.5 | 3.2 | 2.6 | 0.8 | 2.3 | 1.6 | 3.2 |
| $\boldsymbol{X_M[L]}$ | 2 | 2 | 3 | **1** | 3 | 1 | 3 | 2 | **2** |

Figure 4. Update process of machine vector

2) The process of updating vector $X_P[L]$ is as shown in Figure 5. Calculate the vector $X_P[L]$ according to Equation (2), sort the calculated results incrementally, and then the corresponding sequence will be sorted to get a new process sequence.

| **Initial value** | **2** | **1** | **3** | **2** | **1** | **1** | **2** | **3** | **3** |
|---|---|---|---|---|---|---|---|---|---|
| **Updated value** | 5.2 | 2.3 | 4.1 | 5.0 | 3.1 | 4.0 | 4.5 | 1.6 | 2.7 |
| **Updated array** | 9 | 2 | 6 | 8 | 4 | 5 | 7 | 1 | 3 |
| $\boldsymbol{X_P[L]}$ | 3 | 1 | 3 | 1 | 1 | 3 | 2 | 2 | 2 |

Figure 5. Update process of process vector

## 5. Experimental Analysis and Results.

### 5.1. Experiment based on CEC2013 standard test function set.

In this paper, 15 benchmark functions of CEC2013 standard test function set are selected to test the performance of TSPSO algorithm. The test set includes three unimodal functions, eight basic multimodal functions and four composition functions. The specific information of the test function is shown in Table 1. In order to verify the performance of TSPSO algorithm in solving complex problems, we compare TSPSO algorithm with IDE-PSO [8], CLPSO [11], SRPSO [12] and standard PSO [42]. In order to ensure the fairness of the test, the parameters of the algorithm are as follows: the population size is 40, the number of iterations is 1000, and the algorithm runs independently 30 times when $D = 10$, 30 and 50, respectively. The experimental environment is: Inter i5 CPU 2.50GHz, RAM 4GB, Windows7, MATLAB R2016a.

Tables 2-4 show the results of GPSO, CLPSO, SRPSO, IDE-PSO and TSPSO algorithm on the test function when $D = 10$, 30 and 50, respectively, including the average value and the standard deviation, and bold indicates the optimal result. Refer to the data statistics and analysis methods in [43], Wilcoxon rank test with significance level of 0.05 is used to judge the performance of the algorithm. Here "$+$", "$-$", "$\approx$" indicate that the result of TSPSO algorithm is superior to, inferior to and equivalent to the test result of corresponding algorithm.

From the results of Tables 2-4, it is concluded that TSPSO can find better results in unimodal function, multimodal function and composition function in both low-dimensional and high-dimensional than other algorithms. According to Wilcoxon's results in Table 5,

Table 1. CEC2013 standard test function

| | No. | Functions | $f_{i^*} = f_i(x^*)$ |
|---|---|---|---|
| Unimodal Functions | 1 | Rotated High Conditioned Elliptic Function | $-1300$ |
| | 2 | Rotated Bent Cigar Function | $-1200$ |
| | 3 | Rotated Discus Function | $-1100$ |
| Basic Multimodal Functions | 4 | Rotated Rosenbrock's Function | $-900$ |
| | 5 | Rotated Ackley's Function | $-800$ |
| | 6 | Rotated Ackley's Function | $-700$ |
| | 7 | Rotated Weierstrass Function | $-600$ |
| | 8 | Rotated Griewank's Function | $-500$ |
| | 9 | Lunacek Bi_Rastrigin Function | 300 |
| | 10 | Rotated Lunacek Bi_Rastrigin Function | 400 |
| | 11 | Expanded Griewank's plus Rosenbrock's Function | 500 |
| Composition Functions | 12 | Composition Function 1 ($n = 5$, Rotated) | 700 |
| | 13 | Composition Function 4 ($n = 3$, Rotated) | 1000 |
| | 14 | Composition Function 7 ($n = 5$, Rotated) | 1300 |
| | 15 | Composition Function 8 ($n = 5$, Rotated) | 1400 |

when $\alpha = 0.05$, the TSPSO algorithm has obvious advantages over the comparison algorithms in the test function. In particular, TSPSO algorithm has outstanding advantages in solving high-dimensional problems. In addition, the convergence of the algorithm on some functions is shown in Figures 6(a)-6(i) respectively. It can be clearly observed from the figure that the convergence speed and accuracy of TSPSO algorithm are significantly improved. Although TSPSO algorithm has its own shortcomings, when the dimension increases, the stability of the algorithm is slightly less than other algorithms, but in general, compared with other algorithms, TSPSO algorithm has a great improvement in the optimization results.

5.2. **The experiment of solving FJSP problem.** For the sake of verifying the feasibility and effectiveness of TSPSO algorithm proposed in this paper in solving the FJSP problem, the benchmark problem example proposed by Brandimarte is used for testing. The data set (BRdata) is 10 benchmark instances of flexible job shop scheduling problem with the number of jobs in the range of $[10, 20]$, and the number of machines in the range of $[4, 15]$ (the instance data can be downloaded from http://www.dsia.ch/monaldo). In this paper, TSPSO algorithm is compared with other algorithms used to solve FJSP problem. The parameters of the algorithm are as follows: population number $N = 100$, iteration number $T = 200$.

Table 6 shows the comparison between hybrid PSO algorithm [17], migratory bird algorithm [32], QPSO [39], hybrid genetic algorithm [27] and TSPSO algorithm in solving the FJSP problem. The relative deviation is calculated by the formula $dev = [100 * (C_{\max} - C^*_{\max})/C^*_{\max}]$. $C^*_{\max}$ is the optimization result of the algorithm in this paper. $C_{\max}$ is the optimization result of the comparison algorithm. The relative deviation value shows the deviation degree between the optimization result of this algorithm and that of the comparison algorithm. If the result is 0, it means that the optimal solution of TSPSO algorithm is the same as that obtained by comparison algorithm. The positive result indicates that the optimization result of TSPSO algorithm is better than the comparison algorithm. The larger the value is, the better the quality of solution obtained by TSPSO algorithm is. The negative result shows that the optimal solution of TSPSO algorithm is

Table 2. The optimized results of comparison algorithm on the test functions ($D = 10$)

| Fun. | Criterion | GPSO | CLPSO | SRPSO | IDE-PSO | TSPSO |
|---|---|---|---|---|---|---|
| $f_1$ | Mean | 5.00E+06 | 3.98E+06 | 3.76E+06 | 2.41E+06 | **4.38E+03** |
| | Std. | 2.57E+06 | 5.31E+06 | 5.58E+06 | 1.00E+07 | **6.53E+03** |
| $f_2$ | Mean | 1.16E+09 | 5.50E+09 | 5.88E+09 | 7.58E+08 | **3.03E+04** |
| | Std. | 4.41E+08 | 9.41E+09 | 7.07E+09 | 2.22E+09 | **1.08E+05** |
| $f_3$ | Mean | 9.75E+03 | 1.99E+04 | 2.08E+04 | 5.26E+03 | **1.30E+01** |
| | Std. | 4.42E+03 | 1.06E+04 | 1.26E+04 | 4.62E+03 | **1.52E+01** |
| $f_4$ | Mean | 6.98E+01 | 4.95E+01 | 4.63E+01 | 2.64E+01 | **7.73E+00** |
| | Std. | 1.88E+01 | 5.97E+01 | 4.33E+01 | 3.56E+01 | **1.47E+01** |
| $f_5$ | Mean | 5.46E+01 | 7.91E+01 | 7.87E+01 | 3.39E+01 | **1.43E+00** |
| | Std. | 2.21E+01 | 5.45E+01 | 4.26E+01 | 1.46E+01 | **3.66E+00** |
| $f_6$ | Mean | 2.05E+01 | 2.05E+01 | 2.04E+01 | 2.04E+01 | **2.04E+01** |
| | Std. | 8.86E-02 | **7.28E-02** | 1.06E-01 | 8.06E-02 | 9.77E-02 |
| $f_7$ | Mean | 7.89E+00 | 6.45E+00 | 5.49E+00 | 5.42E+00 | **1.91E+00** |
| | Std. | 8.36E-01 | 1.39E+00 | 1.61E+00 | 1.45E+00 | **9.08E-01** |
| $f_8$ | Mean | 9.46E+01 | 1.15E+02 | 8.53E+01 | 2.56E+01 | **1.32E-01** |
| | Std. | 2.42E+01 | 1.92E+02 | 9.92E+01 | 4.22E+01 | **1.02E-01** |
| $f_9$ | Mean | 1.17E+02 | 3.18E+01 | 3.16E+01 | 3.03E+01 | **1.78E+01** |
| | Std. | 1.51E+01 | 9.60E+00 | 1.73E+01 | 1.25E+01 | **7.66E+00** |
| $f_{10}$ | Mean | 1.21E+02 | 4.56E+01 | 4.95E+01 | 3.16E+01 | **2.62E+01** |
| | Std. | 1.45E+01 | 8.91E+00 | 1.26E+01 | 1.35E+01 | **8.81E+00** |
| $f_{11}$ | Mean | 1.26E+01 | 5.24E+01 | 3.18E+01 | 1.85E+00 | **7.89E-01** |
| | Std. | 2.63E+00 | 9.85E+01 | 1.04E+02 | 1.31E+00 | **3.51E-01** |
| $f_{12}$ | Mean | 4.72E+02 | 3.99E+02 | **3.70E+02** | 3.87E+02 | **3.70E+02** |
| | Std. | 5.28E+01 | 8.11E+01 | 7.03E+01 | 5.08E+01 | **4.03E+01** |
| $f_{13}$ | Mean | 2.20E+02 | 2.23E+02 | 2.21E+02 | 2.17E+02 | **2.09E+02** |
| | Std. | 1.05E+01 | 5.07E+00 | 5.34E+00 | **4.09E+00** | 5.44E+00 |
| $f_{14}$ | Mean | 6.18E+02 | 5.85E+02 | 5.97E+02 | 5.02E+02 | **3.46E+02** |
| | Std. | **2.73E+01** | 6.00E+01 | 3.15E+01 | 8.45E+01 | 7.88E+01 |
| $f_{15}$ | Mean | 7.71E+02 | 4.09E+02 | 4.16E+02 | 3.93E+02 | **3.13E+02** |
| | Std. | 1.50E+02 | 1.92E+02 | 2.28E+02 | 1.63E+02 | **7.22E+01** |

inferior to that of comparison algorithm, and the higher the value is, the worse the quality of solution obtained by TSPSO algorithm is.

In Table 6, the optimal solution is firstly evaluated. The smaller the optimal solution is, the better the operation result is, and the better solution is identified in bold. It can be seen from the table that the particle swarm optimization algorithm with self-adaptive parameter adjustment can optimize the scheduling scheme to a large extent, and add two strategies to assist the evolution. TSPSO algorithm obtains better optimal values in 7 of 10 standard test instances. Although the optimization results of TSPSO on the examples Mk06, Mk09 and Mk10 are not good, the solution scheme of this algorithm to be more closer to the optimal solution, in general, it is better than the comparison algorithm. Therefore, TSPSO algorithm has better reliability and stability. Figure 7 shows a set of scheduling Gantt chart of TSPSO algorithm on example Mk01, $C^*_{\max} = 40$.

TABLE 3. The optimized results of comparison algorithm on the test functions ($D = 30$)

| Fun. | Criterion | GPSO | CLPSO | SRPSO | IDE-PSO | TSPSO |
|---|---|---|---|---|---|---|
| $f_1$ | Mean | 1.52E+08 | 5.82E+07 | 7.11E+07 | 3.14E+07 | **3.40E+06** |
| | Std. | 4.01E+07 | 3.69E+06 | 7.80E+07 | 1.52E+07 | **2.21E+06** |
| $f_2$ | Mean | 2.60E+10 | 3.80E+11 | 1.01E+12 | 3.71E+10 | **2.36E+08** |
| | Std. | 6.12E+09 | 3.23E+11 | 4.27E+12 | 1.99E+10 | **3.24E+08** |
| $f_3$ | Mean | 6.37E+04 | 6.80E+04 | 6.01E+04 | 6.0E+04 | **5.94E+03** |
| | Std. | 1.73E+04 | 5.62E+03 | 2.27E+04 | 2.67E+04 | **2.33E+03** |
| $f_4$ | Mean | 7.10E+02 | 1.02E+03 | 3.79E+02 | 2.56E+02 | **5.72E+01** |
| | Std. | 9.29E+01 | 4.01E+01 | 3.23E+02 | 1.35E+02 | **2.51E+01** |
| $f_5$ | Mean | 1.55E+02 | 7.29E+02 | 2.61E+02 | 1.74E+02 | **4.57E+01** |
| | Std. | 2.40E+01 | 3.06E+02 | 1.69E+02 | 6.64E+01 | **2.13E+01** |
| $f_6$ | Mean | 2.10E+01 | 2.10E+01 | 2.10E+01 | 2.10E+01 | **2.10E+01** |
| | Std. | 7.64E-02 | 5.34E-02 | 4.59E-02 | 6.00E-02 | **3.06E-02** |
| $f_7$ | Mean | 3.99E+01 | 3.24E+01 | 2.90E+01 | 3.13E+01 | **1.93E+01** |
| | Std. | **1.62E+00** | 3.23E+00 | 3.83E+00 | 2.64E+00 | 2.99E+00 |
| $f_8$ | Mean | 1.46E+03 | 1.59E+03 | 1.36E+03 | 2.84E+02 | **2.08E+0** |
| | Std. | 2.18E+02 | 4.59E+01 | 1.17E+03 | 1.72E+02 | **1.49E+0** |
| $f_9$ | Mean | 7.06E+02 | 2.54E+02 | 2.44E+02 | 2.51E+02 | **9.36E+01** |
| | Std. | 4.92E+01 | **1.27E+01** | 1.34E+02 | 6.01E+01 | 2.46E+01 |
| $f_{10}$ | Mean | 6.99E+02 | 3.26E+02 | 2.31E+02 | 3.02E+02 | **1.99E+02** |
| | Std. | 5.48E+01 | **2.49E+01** | 1.23E+02 | 2.26E+02 | 3.60E+01 |
| $f_{11}$ | Mean | 1.45E+03 | 1.58E+04 | 1.81E+04 | 5.37E+02 | **4.75E+0** |
| | Std. | 5.36E+02 | 1.53E+02 | 6.70E+03 | 1.61E+03 | **1.53E+0** |
| $f_{12}$ | Mean | 1.96E+03 | 1.55E+03 | 8.19E+02 | 5.16E+02 | **3.31E+02** |
| | Std. | 2.46E+02 | 8.97E+01 | 5.41E+02 | 2.49E+02 | **8.80E+01** |
| $f_{13}$ | Mean | 3.08E+02 | 3.55E+02 | 2.90E+02 | 2.90E+02 | **2.52E+02** |
| | Std. | **4.20E+00** | 1.04E+01 | 9.62E+00 | 1.31E+01 | 1.30E+01 |
| $f_{14}$ | Mean | 1.30E+03 | 1.19E+03 | 1.12E+03 | 1.07E+03 | **7.99E+02** |
| | Std. | 4.23E+01 | **4.00E+01** | 8.21E+01 | 8.06E+01 | 7.56E+01 |
| $f_{15}$ | Mean | 2.63E+03 | 2.41E+03 | 2.52E+03 | 2.18E+03 | **3.15E+02** |
| | Std. | **1.78E+02** | 1.79E+02 | 6.81E+02 | 8.58E+02 | 2.31E+02 |

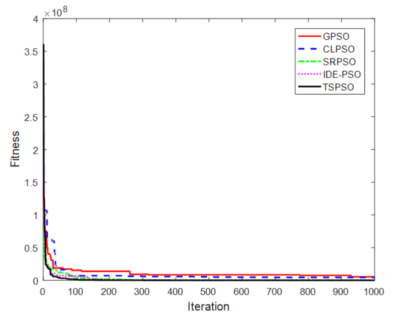
With the improvement of national comprehensive strength, the development of national economy is changing from stable growth stage to high-quality development stage. Manufacturing industry is the basis of economic development, which provides an important guarantee for the national economy, people's livelihood and national defense strength. Optimization of production scheduling is an effective way for manufacturing enterprises to maximize the satisfaction of the target demand in the manufacturing process. Under the condition of adapting to the dynamic changes of the market, it is the key for the survival and development of manufacturing enterprises to reasonably utilize the limited production resources and produce high-quality products at low cost. In this paper, the minimum maximum completion time is taken as the evaluation index, which reflects the production efficiency of the scheduling system, and is the most direct basis for judging the scheduling scheme of FJSP. To sum up, the improved methods proposed in this paper are conducive to improving the ability of the algorithm to solve FJSP, and the algorithm proposed in this paper has certain effectiveness and practical significance in solving FJSP.

Table 4. The optimized results of comparison algorithm on the test functions ($D = 50$)

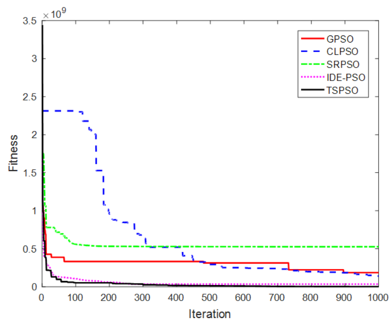
| Fun. | Criterion | GPSO | CLPSO | SRPSO | IDE-PSO | TSPSO |
|---|---|---|---|---|---|---|
| $f_1$ | Mean | 4.82E+08 | 2.15E+08 | 2.22E+08 | 1.21E+08 | **1.95E+07** |
| | Std. | 7.03E+07 | 3.37E+07 | 1.57E+08 | 1.01E+08 | **9.47E+06** |
| $f_2$ | Mean | 1.06E+11 | 7.15E+10 | 1.72E+11 | 8.63E+10 | **3.82E+09** |
| | Std. | 1.43E+10 | 8.07E+09 | 1.01E+11 | 6.68E+10 | **3.83E+09** |
| $f_3$ | Mean | 1.37E+05 | 8.30E+04 | 1.10E+05 | 9.19E+04 | **2.48E+04** |
| | Std. | 2.11E+04 | **6.17E+03** | 4.17E+04 | 2.35E+04 | 6.52E+03 |
| $f_4$ | Mean | 1.99E+03 | 9.75E+02 | 8.46E+02 | 5.49E+02 | **8.09E+01** |
| | Std. | 3.01E+02 | 7.16E+01 | 4.19E+02 | 6.06E+02 | **4.43E+01** |
| $f_5$ | Mean | 2.19E+02 | 1.73E+02 | 2.53E+02 | 1.67E+02 | **7.18E+01** |
| | Std. | 2.13E+01 | 1.94E+01 | 7.69E+01 | 4.03E+01 | **1.59E+01** |
| $f_6$ | Mean | 2.12E+01 | 2.12E+01 | 2.12E+01 | 2.12E+01 | **2.12E+01** |
| | Std. | 3.37E-02 | 4.27E-02 | 5.60E-02 | 1.90E-02 | **1.45E-02** |
| $f_7$ | Mean | 7.40E+01 | 6.64E+01 | 5.68E+01 | 5.84E+01 | **4.20E+01** |
| | Std. | **1.89E+00** | 2.07E+00 | 4.82E+00 | 3.57E+00 | 5.10E+00 |
| $f_8$ | Mean | 4.52E+03 | 2.09E+03 | 3.00E+03 | 9.91E+02 | **6.35E+01** |
| | Std. | 5.79E+02 | 1.84E+02 | 1.89E+03 | 6.53E+02 | **3.00E+01** |
| $f_9$ | Mean | 2.15E+03 | 3.48E+02 | 5.92E+02 | 6.81E+02 | **2.24E+02** |
| | Std. | 2.81E+02 | **2.46E+01** | 2.94E+02 | 1.10E+02 | 5.09E+01 |
| $f_{10}$ | Mean | 1.98E+03 | 7.01E+02 | 6.45E+02 | 7.04E+02 | **4.11E+02** |
| | Std. | 2.32E+02 | **3.93E+01** | 2.33E+02 | 1.36E+02 | 7.12E+01 |
| $f_{11}$ | Mean | 3.17E+04 | 2.79E+04 | 1.08E+05 | 7.84E+03 | **1.99E+01** |
| | Std. | 1.06E+04 | 3.24E+04 | 2.47E+05 | 2.43E+04 | **5.78E+00** |
| $f_{12}$ | Mean | 3.96E+03 | 3.21E+03 | 2.05E+03 | 2.04E+03 | **8.71E+02** |
| | Std. | 8.14E+02 | **6.13E+01** | 7.09E+02 | 9.86E+02 | 3.59E+02 |
| $f_{13}$ | Mean | 3.93E+02 | 4.91E+02 | 3.73E+02 | 3.74E+02 | **3.18E+02** |
| | Std. | **8.38E+00** | 1.71E+01 | 1.42E+01 | 1.41E+01 | 1.27E+01 |
| $f_{14}$ | Mean | 2.19E+03 | 2.32E+03 | 1.84E+03 | 1.89E+03 | **1.40E+03** |
| | Std. | **5.59E+01** | 6.47E+01 | 1.28E+02 | 1.15E+02 | 1.28E+02 |
| $f_{15}$ | Mean | 5.02E+03 | 6.97E+03 | 3.91E+03 | 2.73E+03 | **9.46E+02** |
| | Std. | 1.49E+03 | **2.41E+02** | 1.50E+03 | 1.74E+03 | 1.24E+03 |

Table 5. Rankings obtained through the Wilcoxon signed-rank test

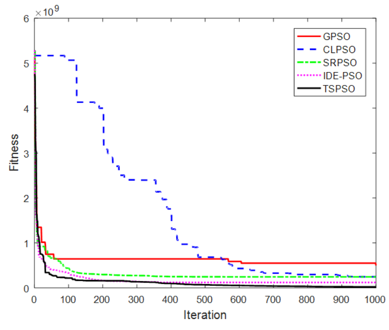
| D | TSPSO VS | $p$-value | + | ≈ | − | $\alpha = 0.05$ |
|---|---|---|---|---|---|---|
| 10 | GPSO | 0.005062 | 15 | 0 | 0 | **Yes** |
| | CLPSO | 0.005062 | 15 | 0 | 0 | **Yes** |
| | SRPSO | 0.007686 | 13 | 2 | 0 | **Yes** |
| | IDE-PSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| 30 | GPSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| | CLPSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| | SRPSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| | IDE-PSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| 50 | GPSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| | CLPSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| | SRPSO | 0.005062 | 14 | 1 | 0 | **Yes** |
| | IDE-PSO | 0.005062 | 14 | 1 | 0 | **Yes** |



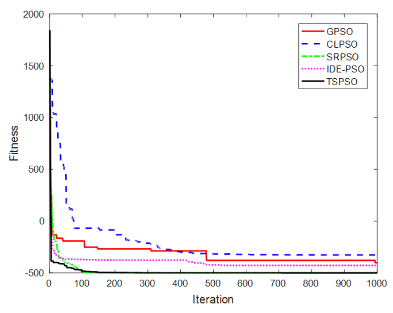
(a) $f_1$ $(D = 10)$



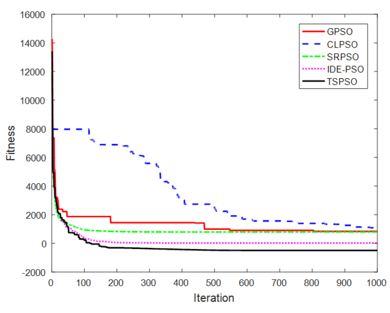
(b) $f_1$ $(D = 30)$



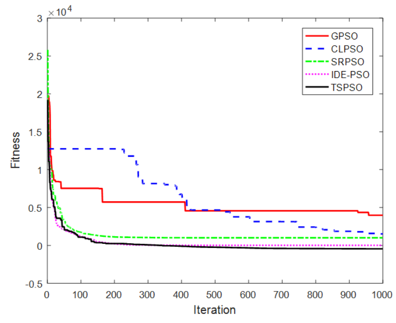
(c) $f_1$ $(D = 50)$



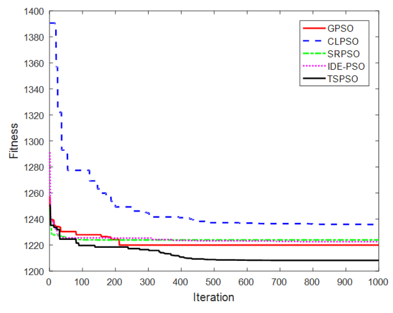
(d) $f_8$ $(D = 10)$



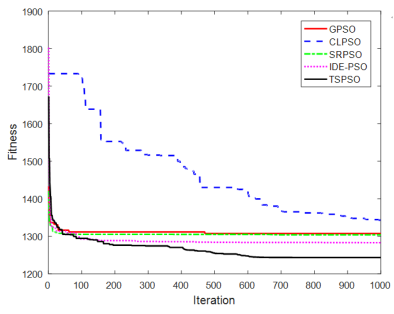
(e) $f_8$ $(D = 30)$



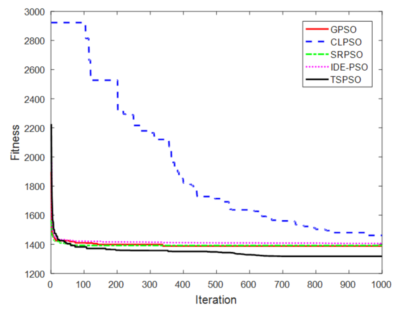
(f) $f_8$ $(D = 50)$



(g) $f_{13}$ $(D = 10)$



(h) $f_{13}$ $(D = 30)$



(i) $f_{13}$ $(D = 50)$

FIGURE 6. Convergence curve of some functions

TABLE 6. Comparison of scheduling results (BRdata)

| Case | $n$ | $m$ | $h$ | Hybrid PSO | | Migratory Bird | | QPSO | | Hybrid GA | | TSPSO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Best | dev/% | Best | dev/% | Best | dev/% | Best | dev/% | Best | Mean |
| Mk01 | 10 | 6 | 55 | **40** | 0 | **40** | 0 | 41 | 2.5 | **40** | 0 | **40** | 40.6 |
| Mk02 | 10 | 6 | 58 | **28** | 0 | 32 | 14.3 | **28** | 0 | **28** | 0 | **28** | 28 |
| Mk03 | 15 | 8 | 150 | **204** | 0 | 207 | 1.5 | 204 | 0 | **204** | 0 | **204** | 204 |
| Mk04 | 15 | 8 | 90 | 64 | 1.6 | 67 | 6.4 | 70 | 11.1 | 67 | 6.4 | **63** | 63.8 |
| Mk05 | 15 | 4 | 106 | 177 | 2.3 | 188 | 8.7 | 179 | 3.5 | 177 | 2.3 | **173** | 176.4 |
| Mk06 | 10 | 15 | 150 | **65** | −3.0 | 85 | 26.9 | 68 | 1.5 | 69 | 3.0 | 67 | 67.7 |
| Mk07 | 20 | 5 | 100 | 145 | 0.7 | 154 | 6.9 | 149 | 3.5 | 145 | 0.7 | **144** | 146.7 |
| Mk08 | 20 | 10 | 225 | **523** | 0 | **523** | **0** | **523** | 0 | **523** | 0 | **523** | 523 |
| Mk09 | 20 | 10 | 240 | 331 | 3.8 | 437 | 36.9 | 342 | 7.2 | **313** | −1.9 | 319 | 325.8 |
| Mk10 | 20 | 15 | 240 | **223** | −6.3 | 380 | 59.6 | 246 | 3.4 | 242 | 1.7 | 238 | 242.6 |

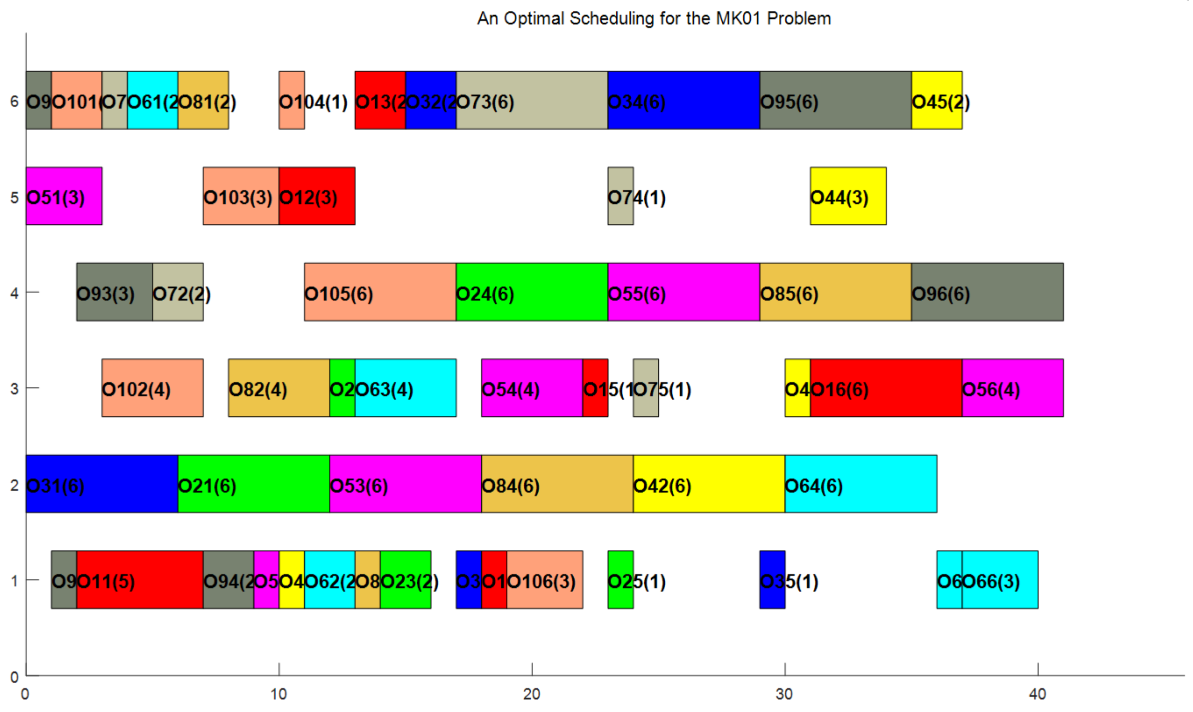


FIGURE 7. Gantt chart of Mk01

6. **Conclusions.** In this paper, a two strategy cooperative particle swarm optimization (TSPSO) algorithm with independent parameter adjustment is proposed. The algorithm adjusts the inertia weight and learning factors of particles according to the evolution rate of the particles. In addition, reconstruction strategy and differential mutation process are introduced to increase population diversity and avoid falling into local optimum, so as to improve the convergence performance of PSO. In the experiment, the TSPSO algorithm is compared with several improved algorithms in CEC2013 standard test function set and flexible job shop scheduling problem. The experimental results show that the TSPSO algorithm has significant improvement in convergence speed and accuracy, even in complex multimodal problems, it can effectively avoid falling into local optimal. The algorithm of this paper is to optimize the single-objective FJSP, solving the multi-objective FJSP problem will be the next research direction.

**Acknowledgement.** This work was financially supported by the National Fund (6186 2041).

## REFERENCES

[1] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, *Proc. of the 6th International Symposium on Micro Machine and Human Science (MHS'95)*, pp.39-43, 1995.
[2] Z. Wu, S. Zhang and T. Wang, A cooperative particle swarm optimization with constriction factor based on simulated annealing, *Computing*, vol.100, no.8, pp.861-880, 2018.
[3] G. Ardizzon, G. Cavazzini and G. Pavesi, Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms, *Information Sciences*, vol.299, pp.337-378, 2015.
[4] D. Tian and Z. Shi, Modified particle swarm optimization and its applications, *Swarm and Evolutionary Computation*, vol.41, pp.49-68, 2018.
[5] J. Gou, Y. X. Lei, W. P. Guo, C. Wang, Y. Q. Cai and W. Luo, A novel improved particle swarm optimization algorithm based on individual difference evolution, *Applied Soft Computing*, vol.57, pp.468-481, 2017.

[6] Y. Ai, Y. X. Su and Y. Peng, Optimization of reactive power based on dynamic learning factor multi-objective particle swarm algorithm, *Applied Mechanics and Materials*, vol.556, pp.3984-3987, 2014.

[7] L. Zhang, Y. Tang, C. Hua and X. Guan, A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques, *Applied Soft Computing*, vol.28, pp.138-149, 2015.

[8] M. Taherkhani and R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, *Applied Soft Computing*, vol.38, pp.281-295, 2016.

[9] G. Ma, R. Li and L. Liu, Particle swarm optimization algorithm of learning factors and time factor adjusting to weights, *Application Research of Computers*, vol.31, no.11, pp.3291-3294, 2014.

[10] B. Y. Qu, P. N. Suganthan and S. Das, A distance-based locally informed particle swarm model for multimodal optimization, *IEEE Trans. Evolutionary Computation*, vol.17, no.3, pp.387-402, 2013.

[11] J. J. Liang, A. K. Qin and P. N. Suganthan, Comprehensive learning particle swarm optimiser for global optimisation of multimodal functions, *IEEE Trans. Evolutionary Computation*, vol.10, no.3, pp.281-295, 2006.

[12] M. R. Tanweer, S. Suresh and N. Sundararajan, Self regulating particle swarm optimization algorithm, *Information Sciences*, vol.294, pp.182-202, 2015.

[13] Y. Chen, L. Li, H. Peng, J. Xiao, Y. Yang and Y. Shi, Particle swarm optimizer with two differential mutation, *Applied Soft Computing*, vol.61, pp.314-330, 2017.

[14] M. R. Singh, M. Singh, S. S. Mahapatra and N. Jagadev, Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem, *The International Journal of Advanced Manufacturing Technology*, vol.85, nos.9-12, pp.2353-2366, 2016.

[15] Y. Q. Wu, Y. G. Wu and X. L. Liu, Couple-based particle swarm optimization for short-term hydrothermal scheduling, *Applied Soft Computing*, vol.74, pp.440-450, 2019.

[16] M. Eddaly, B. Jarboui and P. Siarry, Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem, *Journal of Computational Design and Engineering*, vol.3, no.4, pp.295-311, 2016.

[17] R. Zeng and Y. Wang, A chaotic simulated annealing and particle swarm improved artificial immune algorithm for flexible job shop scheduling problem, *EURASIP Journal on Wireless Communications and Networking*, vol.2018, no.1, pp.1-10, 2018.

[18] S. S. Rao, Optimization: Theory and applications, *IEEE Trans. Systems Man & Cybernetics*, vol.10, no.5, pp.280-280, 1979.

[19] P. M. Pardalos, H. E. Romeijn and H. Tuy, Recent developments and trends in global optimization, *Journal of Computational and Applied Mathematics*, vol.124, nos.1-2, pp.209-228, 2000.

[20] L. Dang, T. Cao and Y. Hoshino, Hybrid hardware-software architecture for neural networks trained by improved PSO algorithm, *ICIC Express Letters*, vol.11, no.3, pp.565-574, 2017.

[21] U. Younis, A. Khaliq and M. Stleem, Weights aggregated multi-objective particle swarm optimizer for optimal power flow considering the generation cost, emission, transmission loss and bus-voltage profile, *International Journal of Innovative Computing, Information and Control*, vol.14, no.4, pp.1423-1441, 2018.

[22] H. Xin, J. Zhu and T.-R. Tsai, Parameter estimation for the three-parameter Burr-XII distribution under accelerated life testing with type I censoring using particle swarm optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.14, no.5, pp.1959-1968, 2018.

[23] B. Tang, Z. Zhu and J. Luo, A convergence-guaranteed particle swarm optimization method for mobile robot global path planning, *Assembly Automation*, vol.37, no.1, pp.114-129, 2017.

[24] R. Singh, H. Chaudhary and A. K. Singh, A new hybrid teaching-learning particle swarm optimization algorithm for synthesis of linkages to generate path, *Sādhanā*, vol.42, no.11, pp.1-20, 2017.

[25] K. Gao, Z. Cao, L. Zhang, Z. H. Chen, Y. Y. Han and Q. K. Pan, A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems, *IEEE/CAA Journal of Automatica Sinica*, vol.6, no.4, pp.904-916, 2019.

[26] H. Kim, Parallel genetic algorithm with a knowledge base for a redundancy allocation problem considering the sequence of heterogeneous components, *Expert Systems with Applications*, vol.113, pp.328-338, 2018.

[27] X. Li and L. Gao, An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem, *International Journal of Production Economics*, vol.174, pp.93-110, 2016.

[28] E. R. Kato, G. D. de Aguiar Aranha and R. H. Tsunaki, A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and random-restart hill climbing, *Computers & Industrial Engineering*, vol.125, pp.178-189, 2018.

[29] S. Wu, P. Zhang, F. Li, F. Gu and Y. Pan, A hybrid discrete particle swarm optimization-genetic algorithm for multi-task scheduling problem in service oriented manufacturing systems, *Journal of Central South University*, vol.23, no.2, pp.421-429, 2016.

[30] Z. Cui and X. Gu, An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems, *Neurocomputing*, vol.148, pp.248-259, 2015.

[31] B. Peng, Z. Lü and T. Cheng, A tabu search/path relinking algorithm to solve the job shop scheduling problem, *Computers & Operations Research*, vol.53, pp.154-164, 2015.

[32] H. Zhu and B. He, Variable neighborhood search migratory bird optimization algorithm for flexible job shop scheduling, *Microelectronics and Computer*, vol.34, no.4, pp.28-32, 2017.

[33] W. L. Wang, L. X. Fan and X. L. Xu, Multi-objective differential evolution algorithm for flexible job-shop batch scheduling problem, *Computer Integrated Manufacturing Systems*, vol.19, no.10, pp.2481-2492, 2013.

[34] G. Balaraju, S. Venkatesh and B. S. Reddy, Multi-objective flexible job shop scheduling using hybrid differential evolution algorithm, *International Journal of Internet Manufacturing and Services*, vol.3, no.3, pp.226-243, 2014.

[35] Z. Z. Jiang, C. Xia, X. Chen and X. Meng, A discrete differential evolution algorithm for the multi-objective generalized assignment problem, *Journal of Computational and Theoretical Nanoscience*, vol.10, no.12, pp.2819-2825, 2013.

[36] A. Ponsich and C. A. Coello, A hybrid differential evolution – Tabu search algorithm for the solution of job-shop scheduling problems, *Applied Soft Computing*, vol.13, no.1, pp.462-474, 2013.

[37] K. Z. Gao, P. N. Suganthan, Q. K. Pan and T. J. Chua, Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives, *Journal of Intelligent Manufacturing*, vol.27, no.2, pp.363-374, 2016.

[38] F. Zhao, J. Tang and J. Wang, An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem, *Computers & Operations Research*, vol.45, pp.38-50, 2014.

[39] M. R. Singh and S. S. Mahapatra, A quantum behaved particle swarm optimization for flexible job shop scheduling, *Computers & Industrial Engineering*, vol.93, pp.36-44, 2016.

[40] G. Zhang, X. Shao, P. Li and L. Gao, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering*, vol.56, no.4, pp.1309-1318, 2009.

[41] X. Li, Z. Peng, B. Du, J. Guo, W. Xu and K. Zhuang, Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems, *Computers & Industrial Engineering*, vol.113, pp.10-26, 2017.

[42] Y. Shi and C. Eberhart, A modified particle swarm optimizer, *Proc. of the 1998 IEEE International Conference on Evolutionary Computation (ICEC'98)*, pp.69-73, 1998.

[43] J. Derrac, S. García, D. Molina and F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm and Evolutionary Computation*, vol.1, no.1, pp.3-18, 2011.