Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4568906, 22 pages
https://doi.org/10.1155/2020/4568906

Research Article

Modified Atom Search Optimization Based on Immunologic Mechanism and Reinforcement Learning

Yanming Fu,[1] Zhuohang Li,[1] Chiwen Qu,[2] and Haiqiang Chen[1]

[1] *Computer and Electronic Information College, Guangxi University, Nanning 530004, China*
[2] *School of Information Engineering, Baise University, Baise 533000, China*

Correspondence should be addressed to Chiwen Qu; quchiwen@163.com

Received 9 September 2019; Revised 30 November 2019; Accepted 4 December 2019; Published 14 January 2020

Academic Editor: Eric Florentin

Atom search optimization algorithm has good searching ability and has been successfully applied to calculate hydrogeological parameters and groundwater dispersion coefficient. Since the atom search optimization algorithm is only based on the atom force motion model in molecular dynamics, it has some shortcomings such as slow search speed and low precision during the later stage of iteration. A modified atom search optimization based on the immunologic mechanism and reinforcement learning is proposed to overcome the abovementioned shortcomings in this paper. The proposed algorithm introduces a vaccine operator to better utilize the dominant position in the current atom population so that the speed, accuracy, and domain search ability of the atom search optimization algorithm can be strengthened. The reinforcement learning operator is applied to dynamically adjust the vaccination probability to balance the global exploration ability and local exploitation ability. The test results of 21 benchmark functions confirm that the performance of the proposed algorithm is superior to seven contrast algorithms in search accuracy, convergence speed, and robustness. The proposed algorithm is used to optimize the permutation flow shop scheduling problem. The experimental results indicate that the proposed algorithm can achieve better optimization results than the seven comparative algorithms, so the proposed algorithm has good practical application value.

1. Introduction

Optimization has been a hot topic in scientific research and engineering application [1] such as industry, society, economy, and management. The optimization methods include traditional exact solution method, constructive algorithm, and swarm intelligence algorithm [2]. The traditional exact solution method can obtain the exact solution, but it requires the objective function to be continuous and differentiable. The algorithm of traditional exact solution method is too complex, so it is suitable for solving small-scale problems. Meanwhile, the traditional exact solution method is not good at optimizing multipeak, nonlinear, and dynamic problems. The constructive algorithm [3] can quickly obtain the solution, but the solution quality is poor and difficult to meet the actual engineering needs. The swarm intelligence algorithm simulates the swarm behavior of social organisms to optimize the given objectives and embodies the characteristics of randomness, parallel, and distribution. The swarm intelligence algorithm provides a solution to complex problems without centralized control and without providing the global mathematical model. Compared with traditional optimization methods, the swarm intelligence algorithm has simple principle and fewer parameters and does not need gradient information of the problem. At present, the common swarm intelligence algorithms include particle swarm optimization algorithm [4], differential evolution algorithm [5], cuckoo search algorithm [6], flower pollination algorithm [7], sine-cosine search algorithm [8–10], and artificial bee colony algorithm [11]. They are widely used in function optimization [12–14], combinatorial optimization [15], image segmentation [16], and flow shop scheduling [17–19].

The atom search optimization (ASO) was proposed by Zhao et al. in 2018 based on the force motion model of atoms in molecular dynamics [20]. ASO is a type of physics-inspired swarm intelligence algorithm. The algorithm needs fewer parameters and has good global exploration ability in the optimization process, so it has been successfully applied to calculate hydrogeological parameters and groundwater dispersion coefficient [21, 22]. Experimental results have proved that the ASO is significantly better than particle swarm optimization and genetic algorithm [20]. Since ASO is proposed only based on the forced motion model of atoms in molecular dynamics, it has the shortcomings of low search accuracy and slow convergence speed as other swarm intelligence algorithms.

The particles distribute approximately evenly in nature and constitute a complex group. The individuals of particles exchange locally acquired information and interact in group. Inspired by the above facts, the modified atom search optimization algorithm based on immunologic mechanism and reinforcement learning (MASO) is proposed to enhance the ASO. The immunologic mechanism can make better use of the dominant positions in the current atomic population to promote the speed and accuracy of the MASO. The reinforcement learning is applied to dynamically adjust the vaccination probability to balance the global exploration ability and local exploitation ability of the MASO.

2. Atom Search Optimization Algorithm

Matter is composed of molecules. Atoms are connected by covalent bonds to form molecules. Atoms have mass and volume. The interaction forces between atoms are shown as repulsions or attractions according to the different distances between atoms. In the repulsive region, the repulsion forces between atoms increase sharply as the distances decrease. In the attractive region, when the distances increase to a certain extent, the attraction forces reach maxima. As the distances continue to increase, the attraction forces gradually decrease to zero. When two atoms are at an equilibrium distance, the interaction force between them is zero. In molecules, atomic forces also need to consider the effects of geometric constraints and atoms' internal motions. The geometric constraints and atoms' internal motions are called binding force. Therefore, the constrained atomic motion equation can be simply written as follows:

$$F_i + G_i = m_i a_i, \tag{1}$$

where F_i is the resultant force of the interaction on atom i, G_i is the resultant force of the constraining force on atom i, m_i is the mass, and a_i is the acceleration of atom i.

In ASO, each atom represents a feasible solution in the search space. The mass of an atom represents the quality of a feasible solution. The better the solution is, the higher the mass of the atom will be, and vice versa. All atoms in a population attract or repel each other according to their distances, which cause lighter atoms to move toward heavier atoms. Heavier atoms have less accelerations, which allow them to exploit local space better. Lighter atoms have more accelerations, which allow them to explore a wider search space. The mass $m_i t$ of the atom i at the iteration t can be simply measured by the fitness function. The expression of $m_i t$ is shown in equation (3):

$$M_i(t) = e^{-\left(\mathrm{Fit}_i(t) - \mathrm{Fit}_{\mathrm{best}}(t)/\mathrm{Fit}_{\mathrm{worst}}(t) - \mathrm{Fit}_{\mathrm{best}}(t)\right)}, \tag{2}$$

$$m_i(t) = \frac{M_i(t)}{\sum_{j=1}^{N} M_j(t)}, \tag{3}$$

where N is the total number of atoms, $\mathrm{Fit}_i(t)$ is the fitness function value of the atom i at the iteration t, and $\mathrm{Fit}_{\mathrm{worst}}(t)$ and $\mathrm{Fit}_{\mathrm{best}}(t)$ are the fitness function values of the worst and best atoms at iteration t, respectively.

In ASO, Kbest is defined as a set of the first K atoms that have the best fitness values in the atomic population. The set is also called the K-nearest neighbors of atom i. The atom i needs to interact with its K-nearest neighbors as much as possible to strengthen exploration at the early iteration stage. The atom i needs to interact with its K neighbor as little as possible to enhance the exploitation at the late iteration stage. Therefore, K decreases as the number of iterations increases. When the total number of iterations is T, K can be calculated as follows:

$$\mathrm{K}(t) = N - (N-2) \times \sqrt{\frac{t}{T}}. \tag{4}$$

Since ASO needs to increase the attraction force and reduce the repulsive force with the increase of iteration times, Zhao et al. [20] proposed a simplified model of the interaction F_{ij}. The expression of F_{ij} is as follows:

$$F_{ij} = -\eta(t)\left[2\left(h_{ij}(t)\right)^{13} - \left(h_{ij}(t)\right)^{7}\right], \tag{5}$$

$$\eta(t) = \alpha\left(1 - \frac{t-1}{T}\right)^{3} e^{-(20t/T)}, \tag{6}$$

$$h_{ij}(t) = \begin{cases} h_{\min}, & \dfrac{r_{ij}(t)}{\sigma(t)} < h_{\min}, \\[2ex] \dfrac{r_{ij}(t)}{\sigma(t)}, & h_{\min} \le \dfrac{r_{ij}(t)}{\sigma(t)} \le h_{\max}, \\[2ex] h_{\max}, & \dfrac{r_{ij}(t)}{\sigma(t) > h_{\max}}, \end{cases} \tag{7}$$

where ηt is a depth function used to adjust the repulsion region and the attraction region. ηt is defined in equation (6), where α is the depth weight. According to equation (7), the distance function $h_{ij}(t)$ is updated, where $h_{\min} = 1.1$ and $h_{\max} = 1.4$ are the upper and lower limits of h, respectively. $r_{ij} t$ is the Euclidean distance between atom i and atom j in the iteration t. σt is the length, which is defined as follows:

$$\sigma(t) = \left\| X_{ij}(t), \frac{\sum_{\epsilon \mathrm{Kbest}} X_{ij}(t)}{K(t)} \right\|_2, \tag{8}$$

where X_{ij} is the position component of atom i in the dimension j of the search space. The dynamically changed $h_{\min}$ and $h_{\max}$ are defined in the following equation:

$$\begin{cases} h_{\min} = g_0 + g(t), \\ h_{\max} = u, \end{cases} \tag{9}$$

where g is the drift component and is defined as follows:

$$g(t) = 0.1 \times \sin\left(\frac{\pi}{2} \times \frac{t}{T}\right). \tag{10}$$

The resultant force of the interaction force on atom i is expressed in the following equation, where rand_j is a random number that obeys the uniform distribution of 0-1:

$$F_i^d(t) = \sum_{j \in \text{Kbest}} \text{rand}_j F_{ij}^d(t). \tag{11}$$

Geometric constraints in molecular dynamics theory play an important role in atomic motion. Suppose that every atom has covalent bonds with every atom in Kbest, every atom is constrained by Kbest. The constraining force of the atom i can be described by the following equation:

$$G_i^d(t) = \lambda(t)\left(X_{\text{best}}^d(t) - \right) X_i^d(t), \tag{12}$$

$$\lambda t = \beta e^{-(20t/T)}, \tag{13}$$

where λ is the Lagrange multiplier, β is the multiplier weight, and $X_{\text{best}}^d(t)$ is the position component in dimension d of the best atom at iteration t. According to equations (1), (5), and (12), the updated equation of acceleration can be deduced as follows:

$$\begin{aligned} a_i^d(t) &= \frac{F_i^d(t)}{m_i^d(t)} + \frac{G_i^d(t)}{m_i^d(t)} \\ &= \alpha\left(1 - \frac{t-1}{T}\right)^3 e^{-(20t/T)\Sigma_{j \in \text{Kbest}}\left(\text{rand}_j \left[2 \times \left(h_{ij}(t)\right)^{13} - \left(h_{ij}\right)^7\right]/m_i(t)\right) \times \left(\left(X_j^d(t) - x_i^d(t)\right)/\left\|X_i(t), X_j(t)\right\|_2\right)} \\ &\quad + \beta e^{-(20t/T)\left(X_{\text{best}}^d(t) - X_i^d(t)/m_i(t)\right)}. \end{aligned} \tag{14}$$

According to the above equations, the velocity and position update equations of atom i at iteration $t+1$ are equations (15) and (16), respectively:

$$v_i^d(t+1) = \text{rand}_i^d v_i^d(t) + a_i^d(t), \tag{15}$$

$$X_i^d(t+1) = X_i^d(t) + v_i^d(t+1). \tag{16}$$

The pseudocode of ASO is given in Algorithm 1.

3. Modified Atom Search Optimization Based on Immunologic Mechanism and Reinforcement Learning

3.1. Chaos. Chaos is a nonlinear phenomenon widely existing in nature. It has the characteristics of randomness, ergodicity and inherent regularity [23–25]. Randomness means that the chaotic behavior has the disorderly features similar to random variable. Ergodicity means that chaos can traverse all states according to its own laws without repetition. Regularity means that chaos is generated by deterministic iteration. Therefore, the chaos strategy is used to initialize the swarm intelligence algorithm and avoid the algorithm falling into the local optimum.

There are defects in random initialization of the ASO. These defects can be well solved by the chaotic strategy. In MASO, nine different chaotic maps are used to initialize the atomic population. These nine kinds of mapping will be called circularly to ensure that different problems can be properly initialized. Table 1 displays the basic information about the nine maps. In this paper, the chaos strategy used to initialize population is called the chaos initialization operator.

3.2. Artificial Immune Algorithm and Vaccine Operator. Artificial immune system is an intelligent method inspired by the biological immune system. It is a new information processing system based on the theory of the human immune system. The artificial immune system provides evolutionary learning mechanisms such as noise tolerance, nonteacher learning, no negative examples, and self-organization. The artificial immune system provides a new way to solve the optimization problem of high dimension [26, 27].

The artificial immune algorithm is essentially a group of operators. The artificial immune algorithm can generate and maintain the diversity of population and has the ability of self-regulation, so it is suitable for optimizing and improving the swarm intelligent algorithm.

The ASO does not make full use of the dominant position in the population, so the accuracy and speed of convergence are not outstanding enough. These defects can be improved by introducing the vaccine operator [28–31]. The vaccine operator consists of three parts: vaccine extraction operator, vaccination operator, and immune detection operator.

(1) *Vaccine Extraction Operator.* In the artificial immune algorithm, vaccine is the prior knowledge of the solving problem. The vaccine extraction operator uses the iteration results of ASO as the prior knowledge and extracts the best current position of atoms as vaccine. The formula of vaccine extraction operator is shown below:

$$\text{vaccine} = \begin{cases} X_i(t), & \text{Fit}_i(t) \leq \text{Fit}_{\text{best}}, \\ X_i(t-1), & \text{Fit}_i(t) > \text{Fit}_{\text{best}}. \end{cases} \tag{17}$$

```
Randomly initialize a set of atoms $X$ (solutions) and their velocity $v$, and $\mathrm{Fit}_{\mathrm{Best}} = \mathrm{Inf}$.
While the stop criterion is not satisfied do
  For each atom $X_i$ do
    Calculate the fitness value Fit;
    If $\mathrm{Fit}_i < \mathrm{Fit}_{\mathrm{Best}}$ then
      $\mathrm{Fit}_{\mathrm{Best}} = \mathrm{Fit}_i$;
      $X_{\mathrm{Best}} = X_i$;
    End If.
    Calculate the massing using equations (2) and (3);
    Determine its K neighbors using equation (4);
    Calculate the interaction force $F_i$ and the constraint force $G_i$ using equations (11) and (12), respectively;
    Calculate the acceleration using equation (14);
    Update the velocity using equation (15);
    Update the position using equation (16);
  End For.
End While.
Find the best solution so far $X_{\mathrm{Best}}$
```

Algorithm 1: Pseudocode of the atom search optimization algorithm.

Table 1: Chaotic mapping transform kernel and parameter range.

The name of the map	Transform core	Parameter range
Circle map	$x_{n+1} = x_n + b - (a/2\pi)\sin(2\pi/x_n)$	$a, b \in R$
Gauss/mouse map	$x_{n+1} = e^{-\alpha x_n^2} + \beta$	$\alpha, \beta \in R$
Sine map	$x_{n+1} = \sin(\pi x_n)$	
Logistic map	$x_{n+1} = \mu x_n(1 - x_n)$	$\mu \in (0, 4]$
Iterative map	$x_{n+1} = \sin(\alpha\pi/x_n)$	$\propto \in (0, 1]$
Singer map	$x_{n+1} = u(7.86x_n - 23.31x_n^2) + 28.75x_n^3 - 13.302875x_n^4$	$u \in (0, 4]$
Tent map	$x_{n+1} = \begin{cases} (x_n/q), & 0 < x_n \leq q \\ (1 - x_n/1 - q), & q < x_n < 1 \end{cases}$	$q \in (0, 1)$
Sinusoidal map	$x_{n+1} = 2.3x_n^2 \sin(\pi x_n)$	
Chebyshev map	$x_{n+1} = \cos(\mu a \cos(x_n))$	$\mu \in [0, 1]$

$\mathrm{Fit}_i(t)$ expressed the fitness function value of X_i in the t iteration, and $\mathrm{Fit}_{\mathrm{best}}$ represents the fitness function value of the best position obtained by the iteration so far.

(2) *Vaccination Operator*. Vaccination uses prior knowledge to locally adjust the solutions so that the qualities of the candidate solutions are significantly improved. The vaccination operator will be used to inoculate atomic population. Firstly, a part of the individuals is selected from the atomic population, and then the position of the selected atomic individuals is modified by the following equation:

$$X_{\mathrm{new}_{ij}} = \begin{cases} \mathrm{Vaccine}_{ij}, & \mathrm{Pv} \geq r, \\ X_{ij}, & \mathrm{Pv} < r, \end{cases} \tag{18}$$

where X_{new} is the atomic population after vaccination. r is a random number that obeys the uniform distribution of 0-1.

(3) *Immunoassay Operator*. After the operation of the vaccination operator, the immune detection operator compares the fitness values of the vaccinated individuals and the original individuals. If the fitness values of vaccinated individuals are not as good as the original individuals, it means that there is degradation in the process of inoculation. Then the original individuals become the parents of the next generation directly according to equation (19). Otherwise, the vaccinated individuals become the parents of next generation according to the following equation:

$$X_i(t+1) = \begin{cases} X_{\mathrm{new}_i}(t), & \mathrm{Fit}_i(t) > \mathrm{Newfit}_i(t), \\ X_i(t), & \mathrm{Fit}_i(t) \leq \mathrm{Newfit}_i(t). \end{cases} \tag{19}$$

The process of using the immune mechanism to enhance ASO is shown in Figure 1.

3.3. Reinforcement Learning Mechanism Updates Vaccination Probability

3.3.1. Reinforcement Learning.

3.3.1. Reinforcement Learning. Machine learning includes supervised learning, unsupervised learning, and reinforcement learning. Reinforcement learning takes environmental

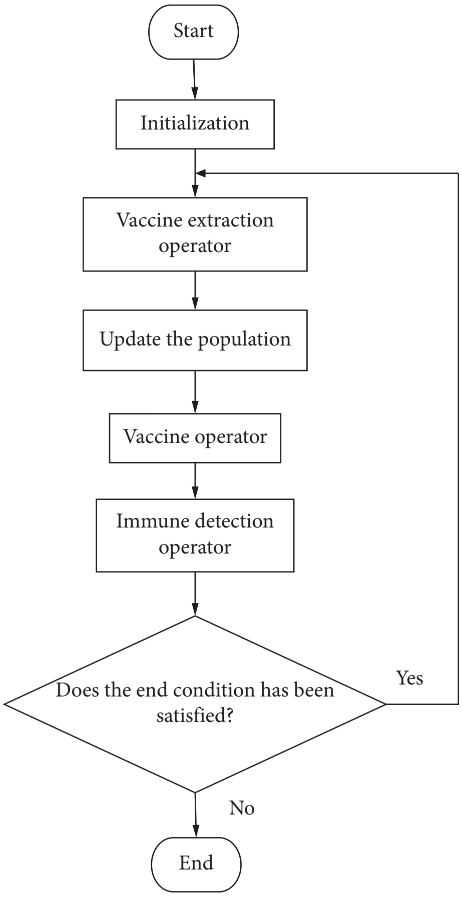
Start
Initialization
Vaccine extraction operator
Update the population
Vaccine operator
Immune detection operator
Does the end condition has been satisfied?
Yes
No
End

Figure 1: Immune mechanism improves ASO.

feedback as input. This method is different from the supervised learning and unsupervised learning. The supervised learning and unsupervised learning tell the Agent what behavior to take through positive and negative examples, but reinforcement learning finds the optimal behavior strategy through trial and error [32].

The principle of reinforcement learning is shown in Figure 2. Reinforcement learning regards learning as a test evaluation process. The Agent chooses an action for the environment, and the environment produces a reinforcement signal (award or punish) feedback to the Agent after its state changes. Then the Agent selects the next action according to the reinforcement signal and the current state of the environment. The principle of selection is to increase the probability of receiving positive reinforcement (award). The action of selection affects not only the immediate reinforcement value but also the state of the environment at the next moment and the final reinforcement value [33, 34].

3.3.2. Adjust Vaccination Probability Dynamically. In the MASO, the vaccination probability Pv is determined according to the actual situation. The reinforcement learning method is used to determine the vaccination probability dynamically. In this paper, this method is collectively called the reinforcement learning operator.

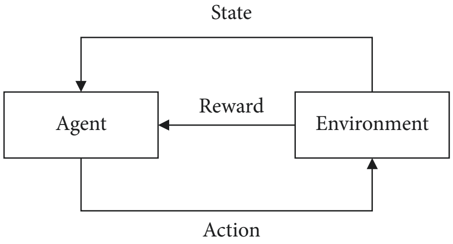

Figure 2: Principle of reinforcement learning.

The reinforcement learning method used here is based on snap-drift neural network [35–37]. The reinforcement learning method switches between snap mode and drift mode [38]. In the snap mode, as Pv decreases, global exploration increases, thus avoiding premature convergence. In the drift mode, as Pv decreases, local exploitation increases, thus improving convergence speed. The Pv value is updated with equation (20). Agent (MASO) accepts the state (snap or drift) and reward value (Pm) at time t and then takes an action (increase Pv or decrease Pv) to convert to a new state. The Pm value is updated with the following equation:

$$\mathrm{Pv} = \begin{cases} \max(0, \mathrm{Pv} - \omega)\,(\text{decrease rule}), & \mu = \text{snap}, \\ \min(1, \mathrm{Pv} + \omega)\,(\text{increase rule}), & \mu = \text{drift}, \end{cases} \tag{20}$$

$$\mathrm{Pm} = \frac{\mathrm{Se}}{N}, \tag{21}$$

where Se is the number of atomic individuals updated in this iteration. Pm is the conversion probability. The value of Pm is determined by the ratio of updated individuals in the population. When Pm is less than 50%, the snap mode is used; when Pm is more than 50%, the drive mode is used. ω is the step size of Pv for each change. Figure 3 illustrates how reinforcement learning operator works.

3.4. Algorithm Flow. After summarizing the above improved methods of ASO, the pseudocode of MASO is given in Algorithm 2.

4. Simulation Experiment

The performance of MASO will be verified by experiments in three aspects: (1) The MASO will compare with seven swarm intelligence algorithms using 21 benchmark functions [39–41] of 30 dimensions. (2) The significance of MASO improvement will be analyzed by the T test. (3) The significance of MASO improved by individual strategies will be analyzed with the Wilcoxon rank sum test.

4.1. Test Platform and Benchmark Functions

4.1.1. Test Platform. The server model is the dawning 5000 A server. The server configures Xeon X5620 CPU (4 cores) $*$ 2, 24 GB memory, and 300 GB SAS hard disk. The server is

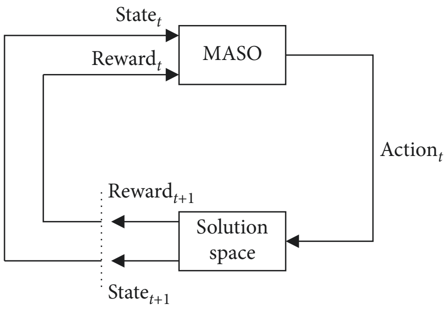

Figure 3: Principle of the reinforcement learning operator.

```
Chaotic initialize a set of atoms $X$ (solutions) and randomly initialize their velocity $v$, evaluate the fitness values by using the
objective function, and then calculate $X_{\mathrm{Best}}$ and $F_{G\mathrm{best}}$. Determine the parameters.
While the stop criterion is not satisfied do
    For each atom $X_i$ do
        Calculate the massing using equations (2) and (3);
        Determine its K neighbors using equation (4);
        Calculate the interaction force $F_i$ and the constraint force $G_i$ using equations (11) and (12), respectively;
        Calculate the acceleration using equation (14);
        Update the velocity using equation (15);
        Update the position using equation (16);
        Check out, update the fitness values;
        Invoke the reinforcement learning operator by equations (20) and (21);
        Invoke the vaccination operator by equations (17) and (18);
        Invoke the immune detection operator by equation (19);
    End For.
End While
Find the best solution so far $X_{\mathrm{Best}}$
```

Algorithm 2: Pseudocode of the modified atom search optimization based on the immunologic mechanism and reinforcement learning.

equipped with Rhel5.6 operating system. The programming tool is MATLAB 2012a (for Linux).

4.1.2. Benchmark Functions. 21 benchmark functions are selected to evaluate the algorithm performance. This set of benchmark functions has widely been used to evaluate the ability of the swarm intelligence algorithm [38]. The number, range, type, and formulation of functions are listed in Table 2. The 21 benchmark functions can be divided into three categories. F1~F6 are the unimodal functions (U) and are applied to investigate the search convergence accuracy. F7–F17 are the complex multimodal functions (M) with multiple local extremum points. These functions are employed to test the global search capability and the ability to avoid prematurity. F18–F21 are the rotating multimodal functions (R). These functions have more extreme points, thus increasing the difficulty of searching. F18–F21 can be adopted to further detect the overall performance of the algorithm.

4.2. Experimental Results Analysis of MASO and Comparison Algorithms. In this experiment, the parameters setting of the comparison algorithms are displayed in Table 3. For each benchmark function, each algorithm runs 20 times independently. The population number is 100. The evaluation times of MASO and CS are $2 * 100$ in each iteration, and evaluation times of other algorithms are 100 in each iteration. For each benchmark function, MASO and CS will iterate 2500 times and other algorithms will iterate 5000 times. So all algorithms will be compared under the evaluations of $100 * 5000$ in each run. The various performances of algorithms are measured by the four indexes of optimal value, average value, worst value, and standard deviation. The experimental statistical results are shown in Table 4. Take 500 points to draw the convergence curve. MASO and CS take one point every 50 iterations, and other algorithms take one point every 100 iterations. Figures 4–24 are the convergence curves of each function successively.

The capacities of MASO in three kinds of functions will be analyzed based on the results of Table 4 and Figures 4–24.

The MASO has a strong overall performance for single mode functions F1–F6. The single mode functions are relatively smooth, with only one extreme point at the origin, so the MASO is easy to converge to global optimum. In F1, F2, F3, and F5, the minimum, average, maximum, and variance of MASO are the best among the all algorithms. In F4, except BAT, CS, and PSO, the minimum, average, maximum, and

Table 2: The benchmark functions.

No.	Range	Type	Formulation
F1	[−100, 100]	U	$f(x)=\sum_{i=1}^{d}x_i^2$
F2	[−10, 10]	U	$f(x)=\sum_{i=1}^{d}\lvert x_i\rvert+\prod_{i=1}^{d}\lvert x_i\rvert$
F3	[−10, 10]	U	$f(x)=\sum_{i=1}^{d}ix_i^2$
F4	[−100, 100]	U	$f(x)=\sum_{i=1}^{d}x_i+0.5^2$
F5	[−1.28, 0.64]	U	$f(x)=\sum_{i=1}^{d}ix_i^4+R$
F6	[−1.28, 1.28]	U	$f(x)=\sum_{i=1}^{d}i{x_i}^4$
F7	[−32, 32]	M	$f(x)=-20\exp[-0.2\sqrt{(1/d)\sum_{i=1}^{d}{x_i}^2}]-\exp[(1/d)\sum_{i=1}^{d}\cos(2\pi x_i)]+(20+e)$
F8	[−600, 600]	M	$f(x)=(1/4000)\cdot\sum_{i=1}^{d}x_i^2-\prod_{i=1}^{d}\cos(x_i/\sqrt{i})+1$
F9	[−5.12, 5.12]	M	$f(x)=10*(d)+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$
F10	[−500, 500]	M	$f(x)=418.9829*(d)+\sum_{i=1}^{d}[-x_i\sin(\sqrt{\lvert x_i\rvert})]$
F11	[−10, 10]	M	$f(x)=\sum_{i=1}^{d}\lvert x_i\cdot\sin(x_i)+0.1\cdot x_i\rvert$
F12	[−10, 10]	M	$f(x)=\sum_{i=1}^{d-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\sin^2(3\pi x_1)$
F13	[−100, 100]	M	$f(x)=0.5+(\sin^2(\sum_{i=1}^{d}x_i^2)-0.5/(1+0.001(\sum_{i=1}^{d}x_i^2)))^2+\lvert x_n-1\rvert[1+\sin^2(3\pi x_n)]$
F14	[−5.12, 5.12]	M	$f(x)=\sum_{i=1}^{d}[{y_i}^2-10\cos(2\pi y_i)+10], y_i=\begin{cases}x_i, & \lvert x_i\rvert<(1/2)\\ (\mathrm{round}(2x_i)/2), & \lvert x_i\rvert\geq(1/2)\end{cases}$
F15	[−5, 10]	M	$f(x)=\sum_{i=1}^{d-1}[(1-x_i)^2+100(x_{i+1}-{x_i}^2)^2]$
F16	[−50, 50]	M	$f(x)=(\pi/d)\left\{10\sin^2(\pi z_1)+\sum_{i=1}^{d-1}(z_i-1)^2[1+10\sin^2(\pi z_{i+1})]+(z_d-1)^2\right\}+$ $\sum_{i=1}^{d}u(x_i,10,100,4), z_i=1+(1/4)(x_i+1), u_{x_i,a,k,m}=\begin{cases}k(x_i-a)^m, & (x_i>\mathrm{a or} x_i<-a)\\ 0, & \text{otherwise}\end{cases}$
F17	[−50, 50]	M	$f(x)=$ $(1/d)\left\{\sin^2(\pi x_1)+\sum_{i=1}^{d-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_d-1)^2[1+\sin^2(2\pi x_{i+1})]\right\}+$ $\sum_{i=1}^{d}u(x_i,5,100,4)$
F18	[−32, 32]	R	$f(x)=-20\exp\left\{-0.2\sqrt{(1/d)\sum_{i=1}^{d}{y_i}^2}\right\}-\exp\left\{(1/d)\sum_{i=1}^{d}\cos(2\pi y_i)\right\}+20+e$
F19	[−5.12, 5.12]	R	$f(x)=10*(d)+\sum_{i=1}^{d}[y_i^2-10\cos(2\pi y_i)]$
F20	[−600, 600]	R	$f(x)=(1/4000)\cdot\sum_{i=1}^{d}y_i^2-\prod_{i=1}^{d}\cos(y_i/\sqrt{i})+1$
F21	[−500, 500]	R	$f(x)=418.9829*(d)+\sum_{i=1}^{d}[-y_i\sin(\sqrt{\lvert y_i\rvert})]$

U: unimodel, M: multimodel, and R: rotated.

variance of other algorithms get the optimums. In F6, only MASO converges to the best minimum compared with other algorithms.

In complex multimodal functions F7–F17, MASO generally performs better among the all algorithms. In F9, F10, F12, F13, F14, and F17, the minimum, average, maximum, and variance of MASO are the best among the all algorithms. In F7, F11, and F16, only MASO converges to the global optimum, compared with other algorithms. In F14, the performance of MASO is worse than ASO.

In the rotated multimodal functions F18–F21, the performance of MASO is not significantly superior to other comparison algorithms. There are too many extreme points in the rotating multimodal function, so MASO needs a strong global exploration ability to converge to the global optimum. On the contrary, MASO needs to consider both local development and global search, so it cannot unilaterally maximize the global search capability. This will affect the performance of MASO in optimizing rotating multimodal functions. In F18, F19, and F21, only MASO converges to the global optimum compared with other algorithms. In F20, MASO and ASO converge to the global optimum. The stability of MASO is the best in F19, but in F18, F20, and F21, the stability of MASO is not good enough.

4.3. Effectiveness Analysis of Improved Strategies. ASO is improved in the following three aspects to obtain MASO: (1) The atomic population is initialized by the chaotic operator to make the population location more uniform so that the MASO does not fall into local optimum easily. (2) The dominant positions in the population are used more fully by the vaccine operator to accelerate the MASO convergence. (3) The vaccination probability Pv will be determined

Table 3: The parameters set of contrast algorithms.

Algorithm	Parameters
MASO	$N = 100$; depth = 50; multiplier = 0.2; Pv = 0.1; w = 0.05.
ASO	$N = 100$; depth = 50; multiplier = 0.2.
FPA	$N = 100$; $P = 0.8$.
BAT	$N = 100$; $A = 0.5$; $r = 0.5$; $Q_{\min} = 0$; $Q_{\max} = 2$.
PSO	$N = 100$; $c1 = 1.49445$; $c2 = 1.49445$.
CS	$N = 100$; pa = 0.25.
ACOR	$N = 100$; nSample = 50; $q = 0.5$; zeta = 1.
CSA	$N = 100$; AP = 0.2; $fl = 2$.

Table 4: Test statistical results of functions $f_1 \sim f_{21}$.

Functions	Algorithms	Min	Mean	Max	STD
$f_1(x)$	**MASO**	**2.99E – 45**	**1.82E – 40**	**3.5E – 39**	**7.82E – 40**
	ASO	7.33E – 27	2.64E – 26	1.05E – 25	3.08E – 26
	FPA	0.006587	0.013937	0.023855	0.004844
	BAT	1.09E – 05	155.0339	891.9841	257.7465
	PSO	3.869551	20.12067	77.25433	16.40947
	CS	0.004142	173.506	1688.332	391.8062
	ACOR	6.98E – 13	2.01E – 12	3.85E – 12	7.74E – 13
	CSA	1.29E – 12	8.89E – 09	7.83E – 08	1.9E – 08
$f_2(x)$	**MASO**	**2.41E – 34**	**1.63E – 29**	**1.34E – 28**	**3.9E – 29**
	ASO	2.16E – 13	5.32E – 13	9.67E – 13	2.51E – 13
	FPA	0.677651	1.159856	1.89272	0.340836
	BAT	0.014345	42.11026	124.5488	48.56557
	PSO	6.26162	9.511573	16.34641	2.636531
	CS	0.024397	1.543266	8.754166	2.237715
	ACOR	3.46E – 07	0.000387	0.006348	0.001411
	CSA	9.9E – 07	4.2E – 05	0.000133	3.52E – 05
$f_3(x)$	**MASO**	**1.5E – 47**	**5.73E – 42**	**8.19E – 41**	**1.87E – 41**
	ASO	5.73E – 26	1.73E – 25	3.92E – 25	1.03E – 25
	FPA	0.000957	0.001794	0.002651	0.000518
	BAT	0.000105	0.000156	0.000193	2.85E – 05
	PSO	1.681533	8.073872	24.85504	6.662011
	CS	0.138672	63.08822	290.7739	80.09947
	ACOR	7.82E – 14	2.16E – 13	4.42E – 13	1.09E – 13
	CSA	1.27E – 12	1.63E – 09	1.89E – 08	4.14E – 09
$f_4(x)$	**MASO**	**0**	**0**	**0**	**0**
	ASO	**0**	**0**	**0**	**0**
	FPA	**0**	**0**	**0**	**0**
	BAT	4214	10180.7	16319	3108.576
	PSO	57	191.7	405	90.82435
	CS	9	1558.45	6328	1791.922
	ACOR	**0**	**0**	**0**	**0**
	CSA	**0**	**0**	**0**	**0**
$f_5(x)$	**MASO**	**2.34E – 06**	**1.47E – 05**	**3.73E – 05**	**9.24E – 06**
	ASO	0.003764	0.00772	0.011135	0.002189
	FPA	0.006508	0.020989	0.029649	0.005437
	BAT	0.218579	0.403253	0.531159	0.080489
	PSO	0.273095	0.62667	1.206052	0.285222
	CS	0.94235	4.144344	13.32052	2.939838
	ACOR	0.016274	0.032401	0.053185	0.009913
	CSA	3.99E – 06	5.29E – 05	0.000164	4.22E – 05

Table 4: Continued.

Functions	Algorithms	Min	Mean	Max	STD
$f_6(x)$	**MASO**	**2.41E − 60**	6.82E − 51	1.26E − 49	2.82E − 50
	ASO	2.19E − 52	**1.46E − 51**	**5.24E − 51**	**1.51E − 51**
	FPA	3.28E − 11	3.39E − 10	1.43E − 09	3.33E − 10
	BAT	6.29E − 11	1.02E − 10	1.45E − 10	2.17E − 11
	PSO	0.001601	0.010088	0.038612	0.010173
	CS	0.000168	0.34018	2.726353	0.655415
	ACOR	2.22E − 14	2.49E − 13	1.82E − 12	3.95E − 13
	CSA	3.56E − 31	1.06E − 23	1.02E − 22	2.42E − 23
$f_7(x)$	**MASO**	**2.22E − 14**	3.83E − 13	5.36E − 12	1.18E − 12
	ASO	4.35E − 14	**7.41E − 14**	**1.47E − 13**	**2.97E − 14**
	FPA	0.17994	0.618946	1.286978	0.260178
	BAT	10.99074	13.41295	19.95889	1.88331
	PSO	3.213753	5.385464	7.818361	1.282747
	CS	8.631765	13.19331	16.18969	1.882678
	ACOR	2.3E − 07	4.2E − 07	5.56E − 07	1.07E − 07
	CSA	2.84E − 07	1.95E − 05	4.98E − 05	1.54E − 05
$f_8(x)$	**MASO**	1.11E − 16	1.65E − 15	5.66E − 15	1.65E − 15
	ASO	**0**	**0**	**0**	**0**
	FPA	0.086888	0.136963	0.230776	0.044916
	BAT	19.19178	53.02061	94.33961	21.77808
	PSO	0.939324	1.178857	1.343306	0.102008
	CS	0.610131	4.621765	33.27042	8.201352
	ACOR	4.88E − 12	0.022291	0.247193	0.069061
	CSA	3.57E − 10	4.25E − 08	4.24E − 07	9.87E − 08
$f_9(x)$	**MASO**	**0**	**4.97E − 13**	**3.18E − 12**	**9.06E − 13**
	ASO	9.949591	14.5264	20.89413	3.356089
	FPA	53.45329	72.74809	84.86339	8.314181
	BAT	38.80531	79.29986	169.1439	35.18297
	PSO	75.51275	110.7482	169.454	25.42386
	CS	30.82795	51.57561	84.51111	14.55878
	ACOR	183.2672	214.6805	237.272	12.35372
	CSA	1.76E − 12	2.82E − 09	7.92E − 09	2.7E − 09
$f_{10}(x)$	**MASO**	**0.000382**	**0.000382**	**0.000382**	**1.35E − 11**
	ASO	4106.132	4650.915	5626.077	521.8557
	FPA	3244.947	3698.782	4219.299	237.132
	BAT	5972.237	7099.381	8123.466	576.3557
	PSO	3897.536	5375.416	6743.524	814.7786
	CS	1848.715	2793.954	4026.17	610.2232
	ACOR	1478.361	5880.244	7607.56	360.132
	CSA	**0.000382**	**0.000382**	**0.000382**	7E − 08
$f_{11}(x)$	**MASO**	**3.44E − 33**	1.83E − 05	9.49E − 05	3.11E − 05
	ASO	4.23E − 14	8.76E − 14	1.97E − 13	4.72E − 14
	FPA	3.227243	6.669646	9.146383	1.526006
	BAT	0.834363	3.926399	9.459001	2.576305
	PSO	2.748819	6.186047	10.35792	2.432746
	CS	0.039447	0.854302	2.652989	0.811748
	ACOR	0.034574	0.302129	5.21918	1.157372
	CSA	7.64E − 08	**5.4E − 06**	**2.31E − 05**	**5.86E − 06**
$f_{12}(x)$	**MASO**	**1.15E − 31**	**6.29E − 28**	**1.18E − 26**	**2.63E − 27**
	ASO	2.78E − 27	6.39E − 27	2.1E − 26	5.59E − 27
	FPA	1.055809	2.341216	3.399438	0.776936
	BAT	3.166887	8.757063	15.72329	3.538368
	PSO	1.089431	7.019646	16.18996	3.659522
	CS	9.73E − 09	0.005653	0.104223	0.02323
	ACOR	1.69E − 11	9.04E − 11	4.43E − 10	9.82E − 11
	CSA	1.24E − 13	5.21E − 11	4.53E − 10	1.13E − 10

Table 4: Continued.

Functions	Algorithms	Min	Mean	Max	STD
$f_{13}(x)$	**MASO**	**2.22*E* − 16**	**9.1*E* − 16**	**5.33*E* − 15**	**1.14*E* − 15**
	ASO	0.111929	0.171566	0.247934	0.04255
	FPA	0.164183	0.23541	0.288289	0.035616
	BAT	0.493711	0.498347	0.499585	0.001342
	PSO	0.213232	0.336072	0.442089	0.05312
	CS	0.470226	0.493116	0.498482	0.006452
	ACOR	0.018867	0.036172	0.046726	0.006468
	CSA	1.33*E* − 15	4.72*E* − 12	2.33*E* − 11	6.56*E* − 12
$f_{14}(x)$	**MASO**	**1.78*E* − 15**	**1.44*E* − 14**	**4.09*E* − 14**	**1.04*E* − 14**
	ASO	12	18.70114	35.0005	6.254622
	FPA	58.21314	75.72416	92.27162	10.98064
	BAT	142.0001	225.425	320.25	43.94807
	PSO	51.61432	97.72042	154.1477	27.68505
	CS	22.03245	65.54915	124.1758	23.27628
	ACOR	159.1282	193.0334	215.8543	15.42857
	CSA	5.74*E* − 11	5.9*E* − 09	2.73*E* − 08	8.01*E* − 09
$f_{15}(x)$	**MASO**	**9.9*E* − 13**	7.77***E*** − 09	8.32***E*** − 08	2***E*** − 08
	ASO	13.22012	14.72793	15.51416	0.63515
	FPA	23.19234	26.9907	29.18676	1.618782
	BAT	0.095172	18.58866	73.64575	30.2613
	PSO	305.1689	574.6557	1477.057	316.5058
	CS	92.87089	4700.287	44298.9	10461.76
	ACOR	24.81962	25.10222	25.36799	0.158517
	CSA	2.24*E* − 11	**1.35*E* − 09**	**5.65*E* − 09**	**1.97*E* − 09**
$f_{16}(x)$	**MASO**	**3.12*E* − 32**	**1.12*E* − 29**	2.02*E* − 28	4.51*E* − 29
	ASO	3.53*E* − 29	8.02*E* − 29	**1.31*E* − 28**	**3.4*E* − 29**
	FPA	0.795144	1.715761	2.908752	0.550818
	BAT	0.519122	11.97827	28.55629	9.510974
	PSO	3.03539	7.040329	11.93878	2.719544
	CS	7.55*E* − 08	0.019757	0.366427	0.081707
	ACOR	9.06*E* − 09	2.61*E* − 07	2.1*E* − 06	4.61*E* − 07
	CSA	2.13*E* − 14	3.58*E* − 11	1.54*E* − 10	4.63*E* − 11
$f_{17}(x)$	**MASO**	**1.26*E* − 32**	**2.26*E* − 31**	**1.86*E* − 30**	**4.61*E* − 31**
	ASO	6.55*E* − 28	2.14*E* − 27	5.54*E* − 27	1.53*E* − 27
	FPA	0.075634	0.159202	0.383942	0.076028
	BAT	28.02352	59.78051	75.72239	10.54314
	PSO	3.843779	71.81837	922.9239	200.6675
	CS	3.36*E* − 22	8.9*E* − 22	1.77*E* − 21	4.31*E* − 22
	ACOR	3.17*E* − 08	8.81*E* − 07	6.2*E* − 06	1.69*E* − 06
	CSA	9.23*E* − 13	2.67*E* − 10	8.43*E* − 10	2.34*E* − 10
$f_{18}(x)$	**MASO**	**2.22*E* − 14**	1.37*E* − 13	8.96*E* − 13	2.02*E* − 13
	ASO	4.71*E* − 14	**9.18*E* − 14**	**1.75*E* − 13**	**3.96*E* − 14**
	FPA	1.169421	1.510639	2.147189	0.28967
	BAT	12.84255	14.03792	16.05812	0.884501
	PSO	3.803351	5.659908	7.923137	0.986783
	CS	14.20253	16.00982	17.58788	1.084051
	ACOR	3.56*E* − 07	8.04*E* − 07	1.54*E* − 06	3.01*E* − 07
	CSA	6.75*E* − 07	2.1*E* − 05	5.99*E* − 05	1.93*E* − 05
$f_{19}(x)$	**MASO**	**0**	**5*E* − 13**	**4.66*E* − 12**	**1.05*E* − 12**
	ASO	8.954632	16.01884	19.89918	3.492605
	FPA	59.36055	76.74372	100.1136	10.82332
	BAT	31.84028	84.16432	198.9923	34.74113
	PSO	68.84252	109.8572	159.7136	22.65396
	CS	65.6753	109.6626	154.4972	25.50229
	ACOR	202.1726	219.1054	230.5434	8.423366
	CSA	6.31*E* − 12	5.12*E* − 09	3.79*E* − 08	1.01*E* − 08

Table 4: Continued.

Functions	Algorithms	Min	Mean	Max	STD
$f_{20}(x)$	**MASO**	**0**	$1.46E-15$	$8.99E-15$	$1.99E-15$
	ASO	**0**	**0**	**0**	**0**
	FPA	0.053602	0.127282	0.226275	0.05216
	BAT	30.91076	58.58257	119.8216	20.73163
	PSO	0.744758	1.095692	1.310903	0.116757
	CS	0.868257	3.707799	19.43772	4.708682
	ACOR	$8.4E-10$	0.00311	0.062197	0.013908
	CSA	$2.6E-13$	$3.35E-08$	$1.31E-07$	$4.47E-08$
$f_{21}(x)$	**MASO**	**1140.688**	**2142.009**	**2596.788**	363.9108
	ASO	4046.881	4920.749	5321.667	391.9594
	FPA	2480.206	2843.365	3076.285	**173.4922**
	BAT	3444.441	5378.728	7371.65	998.61
	PSO	1906.464	4896.28	6944.173	1242.374
	CS	3032.338	4125.281	5258.021	555.419
	ACOR	2691.551	3918.742	5687.103	463.1189
	CSA	7233.943	8295.476	9227.791	487.6725

The best results are shown in bold.

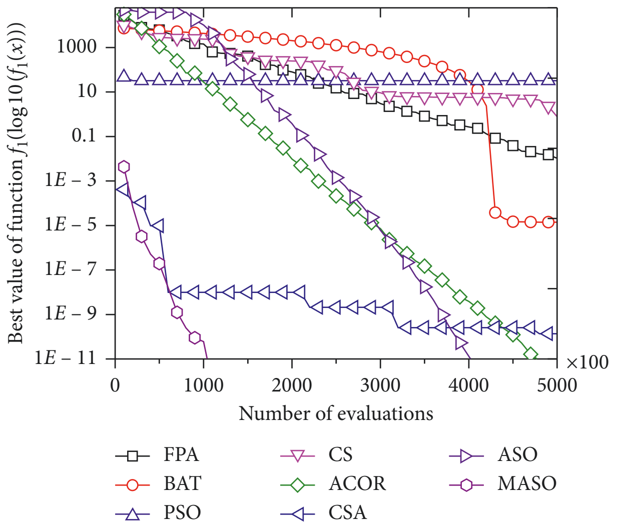

Figure 4: Convergence rates for F1.

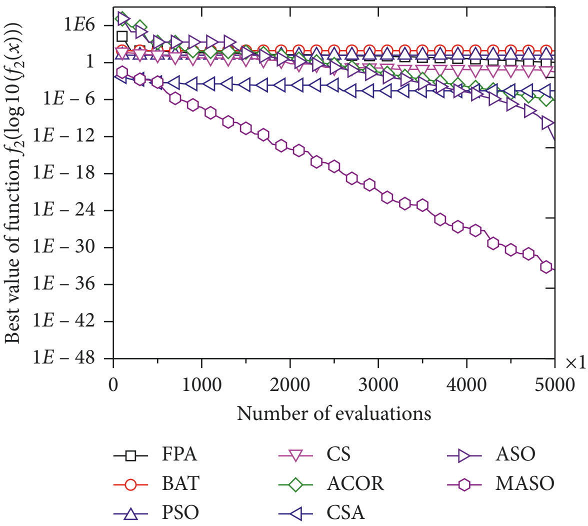

Figure 5: Convergence rates for F2.

dynamically by the reinforcement learning mechanism so that Pv adapts the problem automatically in a more scientific and reasonable way. The MASO improved from the above three aspects has high search accuracy and convergence speed. The improvement effectiveness of MASO will be analyzed by the T test and rank sum test, respectively.

4.3.1. The T Test Analysis between MASO and Comparative Algorithms. Table 5 lists the results of T test, where "+" means that MASO has obvious advantages when compared with other algorithms, "=" indicates that the T test results of MASO equals other algorithms, and "≈" means that the T test results of MASO are approximately equal to other algorithms.

In F5, F7, F9, F10, F11, F12, F13, F14, F18, F19, and F21, MASO is more prominent than the most comparative algorithms. In F1, F3, F4, F6, F8, F15, F17, and F20, although performance of MASO is not outstanding, but its advantages are still better than most comparative algorithms. In F2 and F16, the effect of MASO is almost the same as the corresponding comparison algorithm. The improvement of MASO is not significant enough in F2 and F16 because most algorithms can optimize the two functions easily. Except F2 and F16, the effect of MASO is obviously better than most other algorithms, so the improvement of MASO is evidently effective. Therefore, MASO is a successful improvement to ASO.

4.3.2. Wilcoxon Rank Sum Test for Improved Strategies. MASO is obtained by using the chaotic initialization operator, vaccine operator, and reinforcement learning operator to ameliorate ASO. The ASO enhanced by the chaotic initialization operator is named ASO_chaos. The ASO enhanced by the vaccine operator is named ASO_vaccine. The ASO enhanced by the reinforcement learning operator is named ASO_rl.

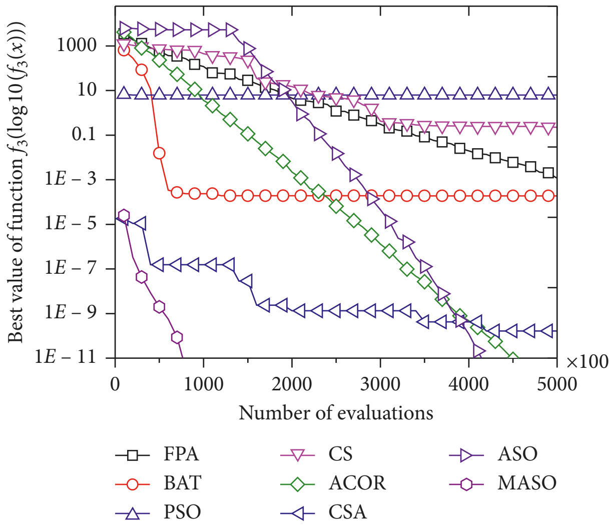

Figure 6: Convergence rates for F3.

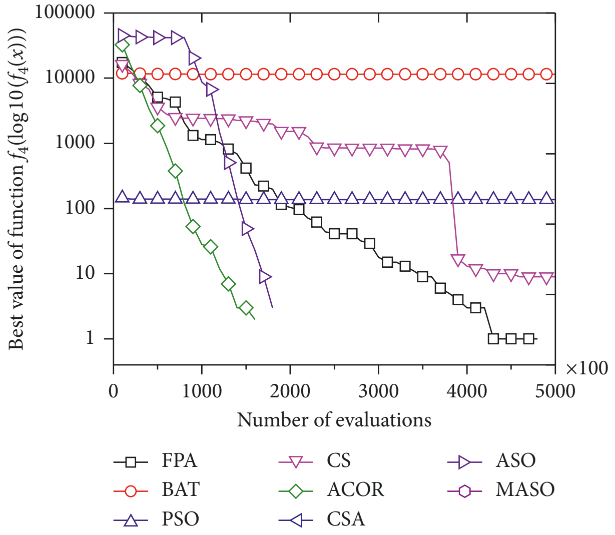

Figure 7: Convergence rates for F4.

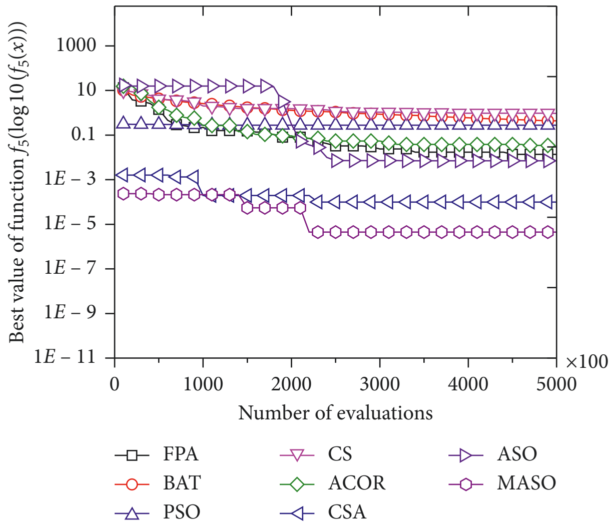

Figure 8: Convergence rates for F5.

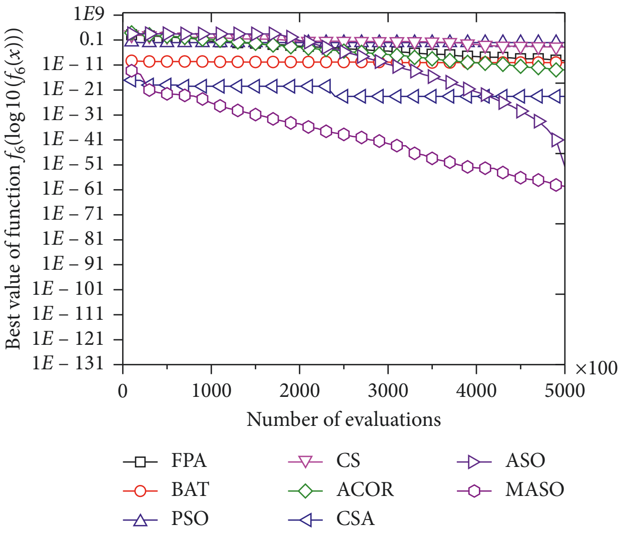

Figure 9: Convergence rates for F6.

In this experiment, each algorithm runs 20 times independently in each benchmark function. The population number is 100. The evaluation times of ASO_vaccine and ASO_rl are 2 ∗ 100 in each iteration, and evaluation times of ASO and ASO_chaos are 100 in each iteration. For each benchmark function, ASO_vaccine and ASO_rl will iterate 2500 times, and ASO and ASO_chaos will iterate 5000 times. So the four algorithms will be compared under the evaluation of 100 ∗ 5000 in each run.

Table 6 lists the statistical results with different strategies. Among the three improved methods, the improvement effect of ASO_chaos is relatively general, and its stability is not good too. ASO_vaccine and ASO_rl are slightly improved compared with the ASO, but the improvement effect is not significant. Generally, the minimums of ASO_vaccine and ASO_rl are at most several orders of magnitude lower than the minimum of ASO. Only the MASO algorithm, that is, enhanced by three operators, can achieve remarkable results.

Table 7 displays the statistical results of the Wilcoxon rank sum test, where "+" means that MASO has obvious advantages when compared with other algorithms and "≈" means that the Wilcoxon rank sum test results of MASO approximately equal to other algorithms. In F5, F9, F10, F13, F14, F19, and F21, ASO_chaos, ASO_vaccine, and ASO_rl are more effective than the ASO. The ASO_chaos has better performance than ASO in F15. The improved effect of ASO_chaos, ASO_vaccine, and ASO_rl is not significant in functions F1–F4, F6–F8, F11–F12, F16–F18, and F20. The effect of MASO is significantly better than ASO and comparative algorithms in most benchmark functions. Therefore, the remarkable effect of MASO can only be achieved by using three operators together.

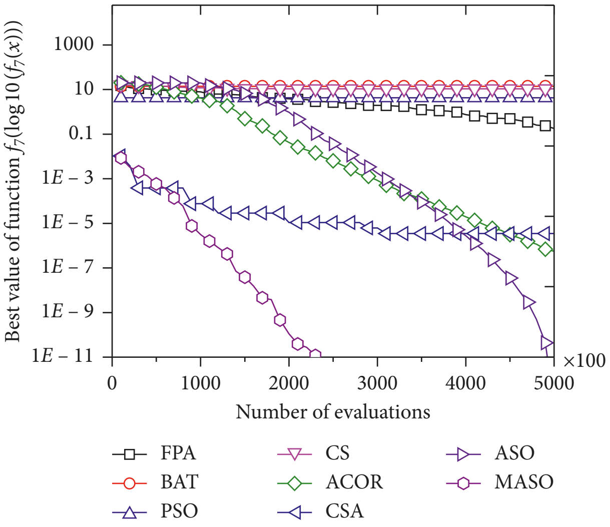

Figure 10: Convergence rates for F7.

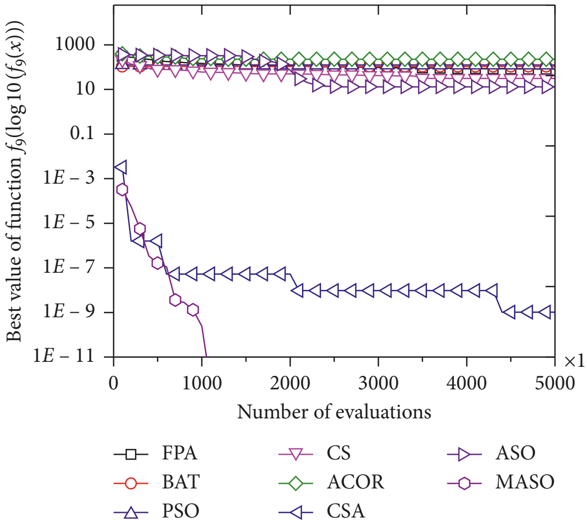

Figure 12: Convergence rates for F9.

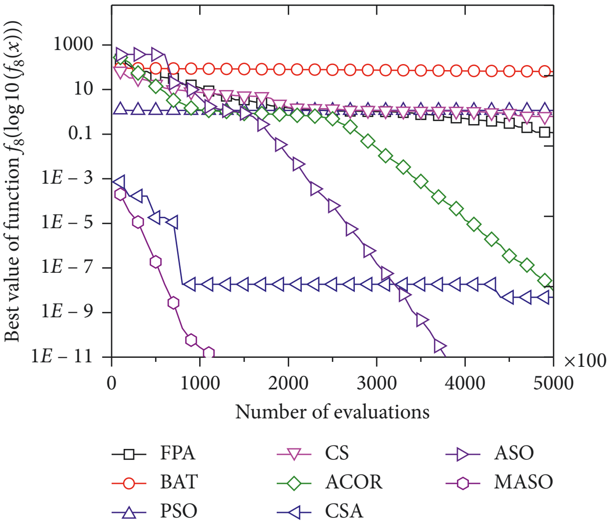

Figure 11: Convergence rates for F8.

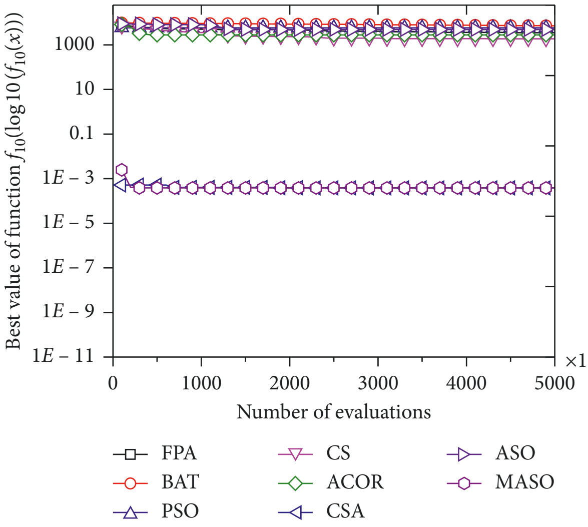

Figure 13: Convergence rates for F10.

5. Application of MASO in Optimization of Permutation Flow Shop Scheduling Problem

In order to verify the optimization capability of MASO for practical engineering problems, MASO will be used to optimize the permutation flow shop scheduling problem.

5.1. Permutation Flow Shop Scheduling Problem. The flow shop scheduling problem studies the process of n jobs on m machines. According to different constraints, flow shop scheduling can be divided into the following different problems: permutation flow shop scheduling problem, no-wait flow shop scheduling problem, no-idle flow shop scheduling problem, blocking flow shop scheduling problem, flow shop scheduling problem with limited buffers, lot-streaming flow shop scheduling problem, and hybrid flow shop scheduling problem [42].

The solution of the permutation flow shop scheduling problem is to find a scheduling sequence so that a certain production index can reach the optimal value. The optimization objective is the minimum makespan. The minimum makespan equals to the completion time of the last workpiece on the last machine.

5.2. Model Assumptions. Some assumptions are made to establish the mathematical model of permutation flow shop scheduling problem:

(1) N workpieces are processed on M machines. Each workpiece must be processed in sequence on different machines without interruption.

(2) There is an infinite buffer area between machines.

(3) In each process, a workpiece can only be processed on one machine without interruption and a machine can only process one workpiece at the same time.

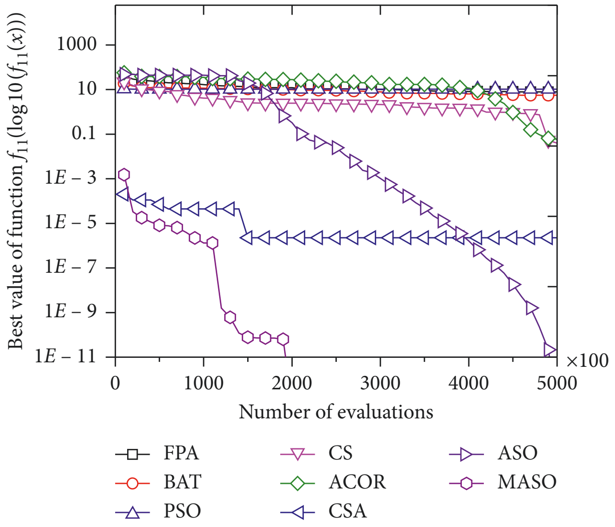

Figure 14: Convergence rates for F11.

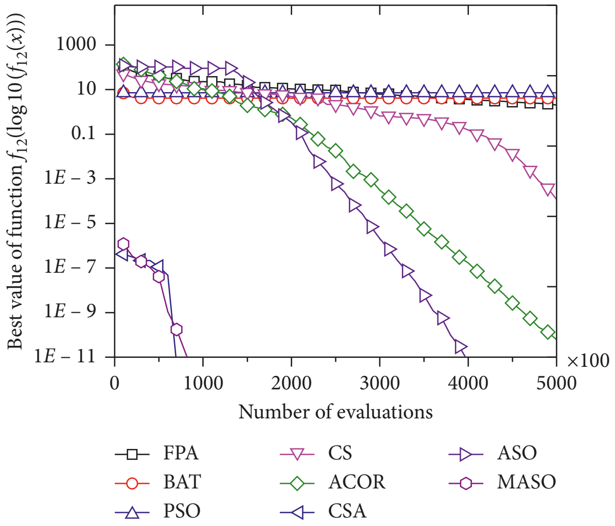

Figure 15: Convergence rates for F12.

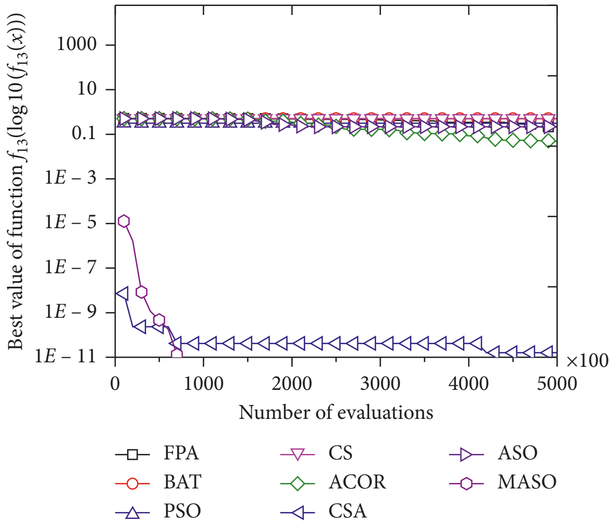

Figure 16: Convergence rates for F13.

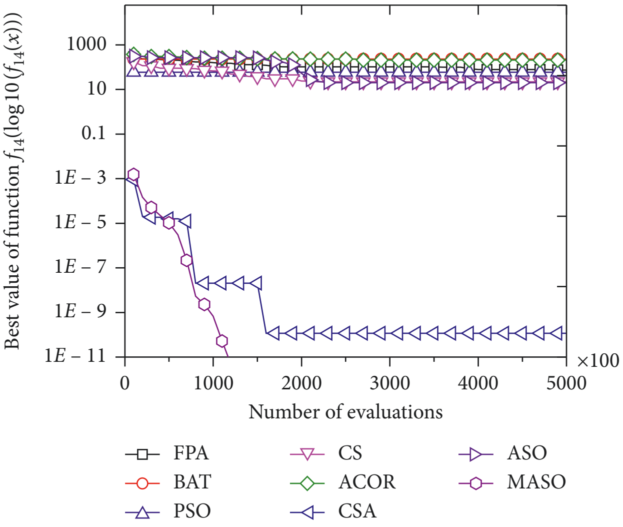

Figure 17: Convergence rates for F14.

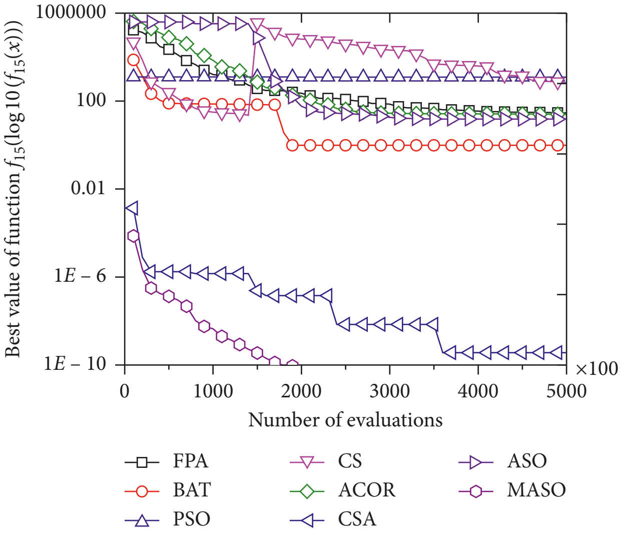

Figure 18: Convergence rates for F15.

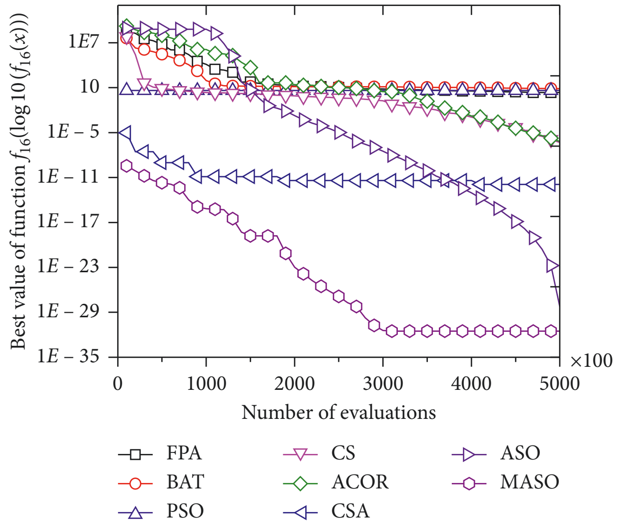

Figure 19: Convergence rates for F16.

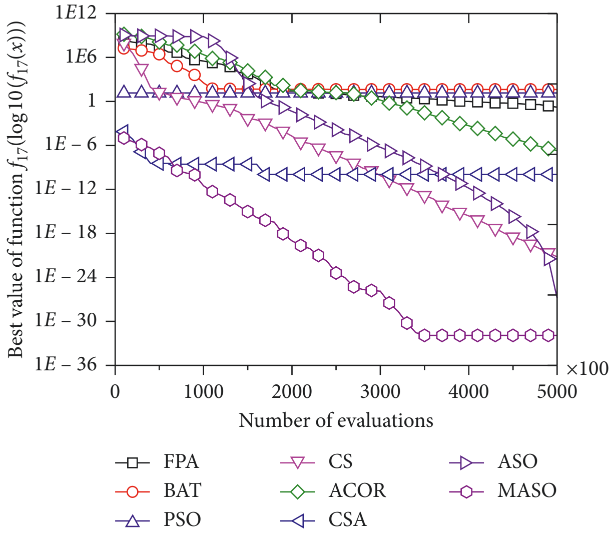

Figure 20: Convergence rates for F17.

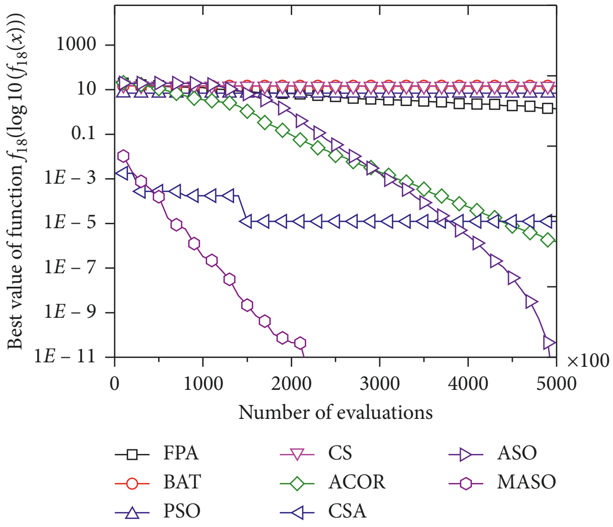

Figure 21: Convergence rates for F18.

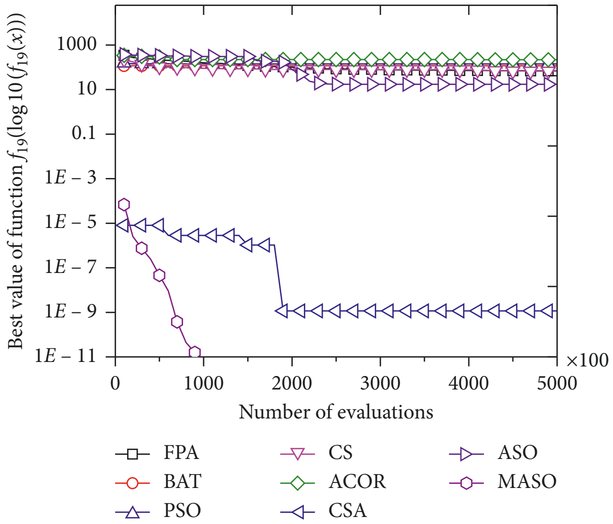

Figure 22: Convergence rates for F19.

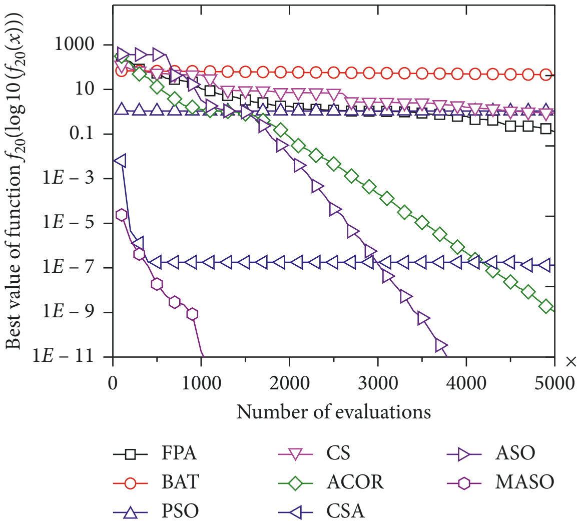

Figure 23: Convergence rates for F20.

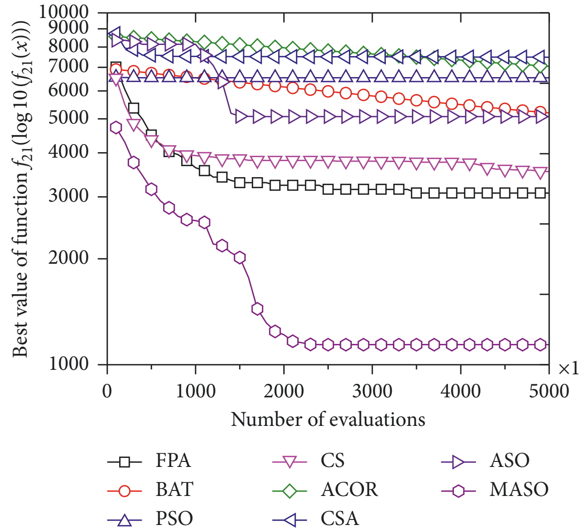

Figure 24: Convergence rates for F21.

(4) The processing time of the workpiece is known, and the preparation time of the workpiece and the start time of the machine are all included in the processing time.

The mathematical models of permutation flow shop scheduling problems are shown in equations (22), (23), and (24) [43].

(1) Objective function:

$$\text{objective} = \text{Min}\left[\text{Max}\left(C_{j,s}\right)\right], \quad (j = 1, \ldots, n). \tag{22}$$

(2) Constraint conditions:

$$C_{j,k} = S_{j,k} + P_{j,k}, \quad (j = 1, \ldots, n;\ k = 1, \ldots, s), \tag{23}$$

$$C_{j,k} \leq S_{j,k+1}, \quad (j = 1, \ldots, n;\ k = 1, \ldots, s). \tag{24}$$

Table 5: The comparisons of T test for f_1~f_{21}.

Algorithm		$f_1(x)$			$f_2(x)$			$f_3(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	1	0.016614	+	0	0.290266	≈	1	0.023023	+
MASO/BAT	1	$3.24E-09$	+	1	0	+	0	0.14638	≈
MASO/PSO	1	$9.2E-105$	+	1	$1.1E-302$	+	1	$1.4E-181$	+
MASO/CS	1	0.000179	+	1	$1.36E-07$	+	1	$1.7E-17$	+
MASO/ACOR	0	0.094316	≈	0	0.273106	≈	0	0.09149	≈
MASO/CSA	0	0.304726	≈	0	0.259636	≈	0	0.310844	≈
MASO/ASO	1	0.000526	+	0	0.481441	≈	1	0.003881	+
		$f_4(x)$			$f_5(x)$			$f_6(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	1	0.004565	+	0	0.050501	≈	1	0.049963	+
MASO/BAT	1	$8.7E-204$	+	1	$1.45E-09$	+	1	$5.04E-28$	+
MASO/PSO	1	$1.7E-183$	+	1	$2.5E-128$	+	1	$6.04E-80$	+
MASO/CS	1	$1.47E-09$	+	1	$7.87E-73$	+	1	0.001168	+
MASO/ACOR	0	0.089476	≈	1	0.029733	+	0	0.111095	≈
MASO/CSA	0	0.303911	≈	1	0.009487	+	0	0.318903	≈
MASO/ASO	N/A	N/A	=	1	$4.62E-05$	+	1	0.0 03612	+
		$f_7(x)$			$f_8(x)$			$f_9(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	1	$2.28E-11$	+	1	0.011808	+	1	$1.6E-30$	+
MASO/BAT	1	0	+	1	$4.66E-84$	+	1	0	+
MASO/PSO	1	$9.1E-295$	+	1	$1.1E-188$	+	1	$9.5E-210$	+
MASO/CS	1	$8.9E-104$	+	1	$5.45E-09$	+	1	$1.03E-47$	+
MASO/ACOR	1	0.001019	+	0	0.079069	≈	1	$1.21E-68$	+
MASO/CSA	0	0.193249	≈	0	0.249111	≈	1	$1.75E-09$	+
MASO/ASO	1	$4.02E-04$	+	0	0.134598	≈	1	0.005241	+
		$f_{10}(x)$			$f_{11}(x)$			$f_{12}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	1	$1.96E-56$	+	1	$1.91E-25$	+	1	$1.21E-07$	+
MASO/BAT	1	$7.53E-91$	+	1	$1.15E-40$	+	1	$9.57E-87$	+
MASO/PSO	1	$7E-250$	+	1	$2.3E-209$	+	1	$1.6E-269$	+
MASO/CS	1	$1.51E-46$	+	1	$3.31E-08$	+	1	$1.6E-278$	+
MASO/ACOR	1	0.003855	+	1	$7.06E-20$	+	1	0.006418	+
MASO/CSA	1	0.003995	+	0	0.128471	≈	0	0.278777	≈
MASO/ASO	1	$1.88E-05$	+	1	0.006584	+	1	0.001213	+
		$f_{13}(x)$			$f_{14}(x)$			$f_{15}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	1	$2.66E-57$	+	1	$6.46E-32$	+	1	0.041413	+
MASO/BAT	1	0	+	1	0	+	0	0.18106	≈
MASO/PSO	1	$7.1E-227$	+	1	$3E-290$	+	1	$5.2E-275$	+
MASO/CS	1	$9.9E-189$	+	1	$9.78E-36$	+	1	0.010623	+
MASO/ACOR	1	$1.34E-17$	+	1	$1.93E-69$	+	1	0.024626	+
MASO/CSA	1	0.001114	+	1	$1.65E-16$	+	0	0.165226	≈
MASO/ASO	1	0.009734	+	1	$3.09E-11$	+	1	$9.92E-53$	+
		$f_{16}(x)$			$f_{17}(x)$			$f_{18}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	0	0.128943	≈	1	0.048392	+	1	$4.89E-15$	+
MASO/BAT	0	0.272098	≈	1	0.045067	+	1	0	+
MASO/PSO	1	$6.8E-214$	+	1	$2.5E-252$	+	1	$2.6E-255$	+
MASO/CS	0	0.266098	≈	0	0.276366	≈	1	$5.9E-177$	+
MASO/ACOR	0	0.116175	≈	1	0.049686	+	1	0.001027	+
MASO/CSA	0	0.307291	≈	0	0.319772	≈	0	0.156388	≈
MASO/ASO	1	0.008508	+	1	$3.3E-04$	+	0	0.206133	+
		$f_{19}(x)$			$f_{20}(x)$			$f_{21}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
MASO/FPA	1	$5.92E-23$	+	1	0.01143	+	1	$5.7E-21$	+
MASO/BAT	1	0	+	1	$8.67E-78$	+	1	$3.82E-62$	+
MASO/PSO	1	$1.1E-178$	+	1	$1.8E-185$	+	1	$1.05E-80$	+
MASO/CS	1	$5.5E-54$	+	1	$2.14E-05$	+	1	$3.74E-28$	+
MASO/ACOR	1	$7.36E-78$	+	0	0.097076	≈	1	$7.62E-51$	+
MASO/CSA	1	$3.03E-52$	+	0	0.237911	≈	1	$2.72E-86$	+
MASO/ASO	1	0.000142	+	0	0.643876	≈	1	$2.28E-36$	+

Table 6: Test statistical results with different strategies.

Functions	Algorithms	Min	Mean	Max	STD
$f_1(x)$	ASO	$7.33E-27$	$2.64E-26$	$1.05E-25$	$3.08E-26$
	ASO_chaos	$2.69E-27$	38220.86	134747.6	49900.47
	ASO_vaccine	$9.34E-27$	$3.35E-26$	$6.9E-26$	$2.51E-26$
	ASO_rl	$5.59E-27$	$4.05E-26$	$9.19E-26$	$3.97E-26$
$f_2(x)$	ASO	$2.16E-13$	$5.32E-13$	$9.67E-13$	$2.51E-13$
	ASO_chaos	$2.99E-13$	$1.58E+18$	$2.84E+19$	$6.35E+18$
	ASO_vaccine	$5.68E-13$	$7.88E-13$	$1.01E-12$	$1.6E-13$
	ASO_rl	$3.18E-13$	$1.25E-12$	$2.35E-12$	$7.73E-13$
$f_3(x)$	ASO	$5.73E-26$	$1.73E-25$	$3.92E-25$	$1.03E-25$
	ASO_chaos	$3.79E-26$	5651.508	17487.91	7310.399
	ASO_vaccine	$8.75E-26$	$2.01E-25$	$5.32E-25$	$1.86E-25$
	ASO_rl	$1.23E-25$	$3.62E-25$	$5.3E-25$	$1.63E-25$
$f_4(x)$	ASO	0	0	0	0
	ASO_chaos	0	42588.7	116706	53955.46
	ASO_vaccine	0	0	0	0
	ASO_rl	0	0	0	0
$f_5(x)$	ASO	0.003764	0.00772	0.011135	0.002189
	ASO_chaos	0.003324	72.91083	301.5848	100.3357
	ASO_vaccine	0.003663	0.005553	0.008711	0.00202
	ASO_rl	0.024793	0.032927	0.046203	0.010069
$f_6(x)$	ASO	$2.19E-52$	$1.46E-51$	$5.24E-51$	$1.51E-51$
	ASO_chaos	$2.34E-53$	94.61799	345.6983	134.9649
	ASO_vaccine	$1.02E-51$	$3.62E-51$	$8.42E-51$	$3.45E-51$
	ASO_rl	$5.22E-52$	$9.87E-51$	$3.93E-50$	$1.67E-50$
$f_7(x)$	ASO	$4.35E-14$	$7.41E-14$	$1.47E-13$	$2.97E-14$
	ASO_chaos	$5.42E-14$	8.462759	21.3434	10.63432
	ASO_vaccine	$6.84E-14$	$9.25E-14$	$1.29E-13$	$2.22E-14$
	ASO_rl	$7.55E-14$	$1.02E-13$	$1.82E-13$	$4.54E-14$
$f_8(x)$	ASO	0	0	0	0
	ASO_chaos	0	344.9339	956.8887	437.1594
	ASO_vaccine	0	0	0	0
	ASO_rl	0	0	0	0
$f_9(x)$	ASO	9.949591	14.5264	20.89413	3.356089
	ASO_chaos	7.959672	230.112	648.2703	274.7404
	ASO_vaccine	11.93951	16.11833	20.89413	4.065958
	ASO_rl	7.959672	12.33749	14.92438	2.68823
$f_{10}(x)$	ASO	4106.132	4650.915	5626.077	521.8557
	ASO_chaos	4312.996	5089.873	5718.976	566.3842
	ASO_vaccine	4500.827	5160.146	5625.985	471.8367
	ASO_rl	3928.42	4852.266	5823.46	748.2354
$f_{11}(x)$	ASO	$4.23E-14$	$8.76E-14$	$1.97E-13$	$4.72E-14$
	ASO_chaos	$2.55E-14$	37.10606	116.9176	47.21336
	ASO_vaccine	$7.52E-14$	$1.12E-13$	$1.6E-13$	$3.61E-14$
	ASO_rl	$7.19E-14$	$1.3E-13$	$2.7E-13$	$8.45E-14$
$f_{12}(x)$	ASO	$2.78E-27$	$6.39E-27$	$2.1E-26$	$5.59E-27$
	ASO_chaos	$2.46E-27$	172.4654	562.9697	222.5449
	ASO_vaccine	$2.06E-27$	$1.86E-26$	$6.71E-26$	$2.73E-26$
	ASO_rl	$3.9E-27$	$1.11E-26$	$1.95E-26$	$7.21E-27$
$f_{13}(x)$	ASO	0.111929	0.171566	0.247934	0.04255
	ASO_chaos	0.006224	0.221173	0.5	0.239022
	ASO_vaccine	0.18622	0.268834	0.353136	0.072799
	ASO_rl	0.094156	0.109595	0.122451	0.010779
$f_{14}(x)$	ASO	12	18.70114	35.0005	6.254622
	ASO_chaos	9	226.4739	622.2764	263.4471
	ASO_vaccine	16.08234	23.04624	27.001	4.654738
	ASO_rl	13	23	30	6.892024

Table 6: Continued.

Functions	Algorithms	Min	Mean	Max	STD
$f_{15}(x)$	ASO	13.22012	14.72793	15.51416	0.63515
	ASO_chaos	14.05125	1691798	5141408	2180894
	ASO_vaccine	15.5044	15.89732	16.17919	0.26884
	ASO_rl	15.80763	16.45844	17.09231	0.562882
$f_{16}(x)$	ASO	$3.53E-29$	$8.02E-29$	$1.31E-28$	$3.4E-29$
	ASO_chaos	$2.17E-29$	$2.2E+09$	$8.17E+09$	$2.87E+09$
	ASO_vaccine	$3.97E-29$	$1.23E-28$	$1.94E-28$	$6.21E-29$
	ASO_rl	$3.35E-29$	$1.49E-28$	$4.22E-28$	$1.59E-28$
	MASO	$3.53E-29$	$8.02E-29$	$1.31E-28$	$3.4E-29$
$f_{17}(x)$	ASO	$6.55E-28$	$2.14E-27$	$5.54E-27$	$1.53E-27$
	ASO_chaos	$6.04E-28$	$1.47E+09$	$5.17E+09$	$1.96E+09$
	ASO_vaccine	$7.69E-28$	$2.92E-27$	$7.94E-27$	$2.89E-27$
	ASO_rl	$7.23E-28$	$2.19E-27$	$5.26E-27$	$1.84E-27$
$f_{18}(x)$	ASO	$4.71E-14$	$9.18E-14$	$1.75E-13$	$3.96E-14$
	ASO_chaos	$5.06E-14$	8.510808	21.42481	10.69466
	ASO_vaccine	$6.13E-14$	$1.59E-13$	$4.34E-13$	$1.55E-13$
	ASO_rl	$1.04E-13$	$1.34E-13$	$1.82E-13$	$2.95E-14$
$f_{19}(x)$	ASO	8.954632	16.01884	19.89918	3.492605
	ASO_chaos	9.949591	226.8982	594.1118	268.2596
	ASO_vaccine	11.9395	22.28707	31.83867	7.629455
	ASO_rl	9.949591	14.5264	19.89917	4.750859
$f_{20}(x)$	ASO	0	0	0	0
	ASO_chaos	0	375.431	1145.34	482.6689
	ASO_vaccine	0	0	0	0
	ASO_rl	0	0	0	0
$f_{21}(x)$	ASO	4046.881	4920.749	5321.667	391.9594
	ASO_chaos	4561.110	5235.886	3421.026	339.6557
	ASO_vaccine	4817.261	5210.066	5626.202	326.6599
	ASO_rl	4540.61	5634.291	6731.833	945.2253

The number of workpieces is n, and a single workpiece is marked as j ($j = 1, \ldots, n$). The number of machines is s, and a single machine is marked k ($k = 1, \ldots, s$). $S_{j,k}$ is the start time of workpiece j in machine k. $P_{j,k}$ is the processing time of workpiece j on machine k. $C_{j,k}$ is the makespan of workpiece j in machine k.

5.3. Engineering Sample. An engineering example of permutation flow shop scheduling problem is given below. Suppose three jobs A, B, and C are processed on three machines in the order of A, B, and C, respectively. The starting time of the first job on the first machine is 0. The processing time of all jobs on each machine is listed in Table 8. The Gantt chart and the makespan are shown in Figure 25.

5.4. Experimental Design. The configuration data about the size and structure of the problem need to be obtained to systematically evaluate the performance of algorithm. The actual data in industry are difficult to obtain, so the test data will be generated randomly. These test data will represent the industrial data. The test is divided into nine problem combinations. Each problem can be identified in the form of "number of jobs-number of machines." For example, problems with 20 jobs and 4 machines can be expressed as "20-4." The factors of level and scope are given in Table 9.

The setting of experimental parameters is the same as in Section 3.2. But for each test problem, MASO is compared with ASO, FPA, BAT, PSO, CS, ACOR, and CSA by running ten times separately.

5.5. Analysis of Experimental Results. The above eight algorithms have been tested ten times on nine combinations of problems, and the test results are listed in Table 10. The minimum makespan and average are two important performance evaluation indexes. The minimum makespan represents the optimization ability of the algorithm. The average represents the stability of the algorithm.

In the "10-2" problem, the minimum makespan obtained by MASO is better than BAT, PSO, and ACOR and is equal to FPA and CS but is worse than the ASO. The stability of MASO is in the middle of all the test algorithms. Generally, the effect is not ideal. In "10-4," the search effect of MASO is general, but the average is the best, and the stability is good. In the "10-8" problem, the search ability of MASO is the best, but it is equal to CSA, ACOR, and BAT, respectively. And the average of MASO is the best in all algorithms. In summary, MASO has almost the same performance as other algorithms under the problem scale of 10 jobs because the scale of the problem is too small to reflect the good search capability of MASO.

Table 7: Test statistical results of the Wilcoxon rank sum test.

Algorithm		$f_1(x)$			$f_2(x)$			$f_3(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	0	0.314619	≈	0	0.324488	≈	0	0.328083	≈
ASO_vaccine/ASO	0	0.488314	≈	0	0.641407	≈	0	0.195942	≈
ASO_rl/ASO	0	0.701861	≈	0	0.925725	≈	0	0.920272	≈
		$f_4(x)$			$f_5(x)$			$f_6(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	0	0.244594	≈	0	0.062368	≈	0	0.701051	≈
ASO_vaccine/ASO	0	0.553674	≈	1	0.030511	+	0	0.315137	≈
ASO_rl/ASO	0	0.893699	≈	1	0.000289	+	0	0.706156	≈
		$f_7(x)$			$f_8(x)$			$f_9(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	0	0.321172	≈	0	0.307772	≈	1	0.000331	+
ASO_vaccine/ASO	0	0.41769	≈	0	0.457487	≈	1	0.004432	+
ASO_rl/ASO	0	0.801112	≈	0	0.636002	≈	1	0.03666	+
		$f_{10}(x)$			$f_{11}(x)$			$f_{12}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	1	$8.67E-08$	+	0	0.301404	≈	0	0.321245	≈
ASO_vaccine/ASO	1	$7.45E-07$	+	0	0.205611	≈	0	0.231387	≈
ASO_rl/ASO	1	$1.61E-07$	+	0	0.898369	≈	0	0.914785	≈
		$f_{13}(x)$			$f_{14}(x)$			$f_{15}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	1	$2.53E-18$	+	1	0.018586	+	1	0.02315	+
ASO_vaccine/ASO	1	$8.18E-05$	+	0	0.064058	≈	0	0.479393	≈
ASO_rl/ASO	1	0.000381	+	1	0.000728	+	0	0.075423	≈
		$f_{16}(x)$			$f_{17}(x)$			$f_{18}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	0	0.449624	≈	0	0.385929	≈	0	0.311159	≈
ASO_vaccine/ASO	0	0.462677	≈	0	0.458521	≈	0	0.555353	≈
ASO_rl/ASO	0	0.769354	≈	0	0.753607	≈	0	0.8118	≈
		$f_{19}(x)$			$f_{20}(x)$			$f_{21}(x)$	
	H	P	Sig	H	P	Sig	H	P	Sig
ASO_chaos/ASO	1	0.000516	+	0	0.389344	≈	1	$4.02E-07$	+
ASO_vaccine/ASO	1	0.003634	+	0	0.412808	≈	1	$1.92E-08$	+
ASO_rl/ASO	1	0.013797	+	0	0.636028	≈	1	$6.05E-07$	+

Table 8: The processing time of the workpiece on each machine.

			Workpieces	
		A	B	C
Machines	1	3	5	1
	2	4	2	2
	3	1	1	3

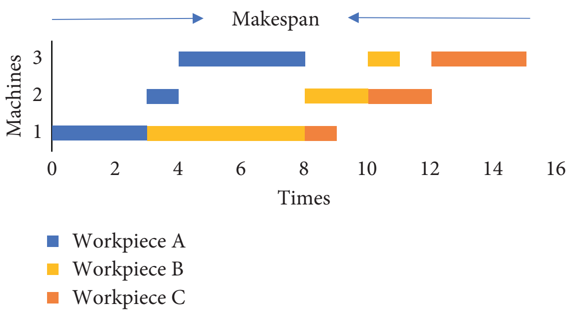

Figure 25: Gantt chart of permutation flow shop scheduling with makespan.

Table 9: The factors of level and range.

Factors	Level	Number of level
Number of jobs	10, 20, and 50	3
Number of machines	2, 4, and 8	3
The processing time of each workpiece on each machine	Discrete uniform [1, 5]	1

Table 10: Permutation flow shop scheduling test results.

Problem		MASO			ASO			FPA			BAT	
	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
10-2	**28**	33.7	39	**26**	32.9	41	**28**	33.5	38	**29**	34.7	38
10-4	**36**	39.1	44	**36**	40.7	46	**38**	41.5	46	**39**	41.2	43
10-8	**50**	55.2	63	**51**	55.8	60	**52**	56.1	61	**50**	56.8	60
20-2	**56**	63.7	71	**58**	65.3	75	**59**	64.2	71	**57**	64.2	75
20-4	**64**	72.6	76	**66**	74.7	83	**62**	70	78	**68**	73.9	80
20-8	**81**	87.6	95	**85**	89.3	97	**84**	88.4	93	**80**	88.4	92
50-2	**147**	158.3	180	**150**	159.1	169	**145**	157	167	**150**	157.6	172
50-4	**153**	164.2	180	**155**	164.4	186	**154**	167.2	184	**157**	166.6	176
50-8	**171**	180.4	191	**173**	178.6	185	**180**	186.6	194	**181**	185.2	190
		PSO			CS			ACOR			CSA	
	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
10-2	**30**	35.3	40	**30**	35.4	39	**29**	34.6	44	**28**	33.5	40
10-4	**35**	40.6	46	**35**	43.5	50	**35**	39.1	42	**34**	40.2	44
10-8	**51**	57.2	64	**51**	57.1	62	**50**	54.2	57	**50**	55.5	61
20-2	**55**	66.4	74	**57**	61.3	67	**62**	65.5	69	**56**	62	70
20-4	**69**	74.8	83	**68**	71.1	80	**64**	69	77	**64**	71.9	83
20-8	**85**	89.7	97	**88**	92.8	98	**82**	87.9	92	**85**	91.5	100
50-2	**143**	158.3	166	**153**	160.6	171	**157**	159.9	167	**147**	157.1	173
50-4	**162**	167.5	178	**160**	167.4	191	**149**	161.2	172	**161**	168.3	173
50-8	**183**	189.8	200	**180**	194.9	201	**177**	182.9	189	**184**	188.7	199

In the "20-2" problem, MASO has the better performance. The minimum makespan of MASO is equal to CSA and is slightly inferior to PSO. In the "20-4" problem, the minimum makespan of MASO is equal to ACOR and CSA and is slightly inferior to FPA. The minimum makespan of MASO ranks second among all the algorithms. In the "20-8" problem, the optimization capability of MASO is only inferior to BAT, ranking second among all algorithms, but its average is the best. In summary, the overall performance of MASO is better than all other algorithms under the scale of 20 workpieces because MASO has strong optimization accuracy and can coordinate the balance between global exploration and local development. In addition to MASO, ACOR performs well on the scale of the problem, so ACOR is also suitable for optimizing this complex permutation flow shop scheduling problem.

In the problem of "50-2," the search effect of MASO is the best, but the average of MASO is worse. In the "50-4" problem, the search effect of MASO is only slightly worse than ACOR. In the "50-8" problem, the search ability of MASO is the best and far better than other algorithms.

Therefore, MASO works better on the problem of higher complexity, especially on the problem of 20 and 50 workpieces. The reason is that the immune operator and reinforcement learning operator strengthen the utilization of the dominant position in the atom population, so the search speed and accuracy of MASO are promoted. Experiments confirm that MASO has advantages to optimize engineering problem.

6. Conclusion

The MASO is proposed to improve ASO in this paper. The chaotic operator, immune operator, and reinforcement learning operator are introduced to enhance ASO. The chaotic operator overcomes the disadvantages of nonuniform initialization in ASO. The immune operator and reinforcement learning operator can make better use of the dominant position in the atom population and employs adaptive parameters to avoid human interference factors. The experimental results in 21 benchmark functions indicate that MASO is better than the seven comparison algorithms in global search ability and convergence speed. T test proves that the performance of MASO is significantly better than the seven comparison algorithms on most benchmark functions. The Wilcoxon rank sum test illustrates that the excellent performance of MASO is obtained by the interaction of three operators. When MASO is applied to optimize the permutation flow shop scheduling problem, the search accuracy of MASO is better than the seven contrast algorithms. MASO does not require the objective function to be convex, continuous, or derivable, so it has strong advantages in the field of numerical optimization. Therefore, MASO can be applied in machine learning, engineering optimization, municipal traffic management, and other optimization fields.

Data Availability

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

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Guangxi Key Laboratory of Multimedia Communications and Network Technology. This work was financially supported by the Natural Science Foundation of China (Grant no. 61761006) and the Natural Science Foundation of Guangxi Province (Grant no. 2017GXNSFAA198263).

References

[1] Z. Y. Wu and A. M. Rubinov, “Global optimality conditions for some classes of optimization problems,” *Journal of Optimization Theory and Applications*, vol. 145, no. 1, pp. 164–185, 2010.

[2] H. Duan and P. Qiao, “Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning,” *International Journal of Intelligent Computing and Cybernetics*, vol. 7, no. 1, pp. 24–37, 2014.

[3] M. S. Nagano, H. H. Miyata, and D. C. Araújo, “A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times,” *Journal of Manufacturing Systems*, vol. 36, pp. 224–230, 2015.

[4] K. James and E. Russell, “Particle swarm optimization,” in *Proceedings of 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Western Australia, 1995.

[5] S. Biswas, S. Kundu, and S. Das, “Inducing niching behavior in differential evolution through local information sharing,” *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 246–263, 2015.

[6] R. Rajabioun, “Cuckoo optimization algorithm,” *Applied Soft Computing*, vol. 11, no. 8, pp. 5508–5518, 2011.

[7] X.-S. Yang, M. Karamanoglu, and X. He, “Flower pollination algorithm: a novel approach for multiobjective optimization,” *Engineering Optimization*, vol. 46, no. 9, pp. 1222–1237, 2014.

[8] S. Mirjalili, “SCA: a Sine Cosine Algorithm for solving optimization problems,” *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[9] C. Qu, Z. Zeng, J. Dai, Z. Yi, and W. He, “A modified sine-cosine algorithm based on neighborhood search and greedy levy mutation,” *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 4231647, 19 pages, 2018.

[10] S. Li, H. Fang, and X. Liu, “Parameter optimization of support vector regression based on sine cosine algorithm,” *Expert Systems with Applications*, vol. 91, pp. 63–77, 2018.

[11] H. Gao, Y. Shi, C. M. Pun et al., “An improved artificial bee colony algorithm with its application,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 1853–1865, 2018.

[12] Q.-K. Pan, H.-Y. Sang, J.-H. Duan, and L. Gao, “An improved fruit fly optimization algorithm for continuous function optimization problems,” *Knowledge-Based Systems*, vol. 62, pp. 69–83, 2014.

[13] M. A. Wei and Z. X. Sun, “A global cuckoo optimization algorithm using coarse-to-fine search,” *Acta Electronica Sinica*, vol. 43, no. 12, pp. 2429–2439, 2015.

[14] H. C. Liu and Z.J. Wu, “Differential evolution algorithm using rotation-based learning,” *Acta Electronica Sinica*, vol. 43, no. 10, pp. 2040–2046, 2015.

[15] A. Baykasoğlu, A. Hamzadayi, and S. Y. Köse, “Testing the performance of teaching–learning based optimization (TLBO) algorithm on combinatorial problems: flow shop and job shop scheduling cases,” *Information Sciences*, vol. 276, pp. 204–218, 2014.

[16] H. Gao, Z. Fu, C.-M. Pun, H. Hu, and R. Lan, “A multi-level thresholding image segmentation based on an improved artificial bee colony algorithm,” *Computers & Electrical Engineering*, vol. 70, pp. 931–938, 2018.

[17] S. Abdollahpour and J. Rezaeian, “Minimizing makespan for flow shop scheduling problem with intermediate buffers by using hybrid approach of artificial immune system,” *Applied Soft Computing*, vol. 28, pp. 44–56, 2015.

[18] C. Zhang, Z. Shi, Z. Huang, Y. Wu, and L. Shi, “Flow shop scheduling with a batch processor and limited buffer,” *International Journal of Production Research*, vol. 55, no. 11, pp. 3217–3233, 2017.

[19] G. Moslehi and D. Khorasanian, “A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion,” *Computers & Operations Research*, vol. 52, pp. 260–268, 2014.

[20] W. Zhao, L. Wang, and Z. Zhang, “A novel atom search optimization for dispersion coefficient estimation in groundwater,” *Future Generation Computer Systems*, vol. 91, pp. 601–610, 2018.

[21] I. Sandholt, K. Rasmussen, and J. Andersen, “A simple interpretation of the surface temperature/vegetation index space for assessment of surface moisture status,” *Remote Sensing of Environment*, vol. 79, no. 2-3, pp. 213–224, 2001.

[22] W. Zhao, L. Wang, and Z. Zhang, “Atom search and its application to solve a hydrogeologic parameter estimation problem,” *Knowledge-Based Systems*, vol. 163, pp. 283–304, 2018.

[23] U. Yüzgeç and M. Eser, “Chaotic based differential evolution algorithm for optimization of baker’s yeast drying process,” *Egyptian Informatics Journal*, vol. 19, no. 3, pp. 151–163, 2018.

[24] C. Choi and J.-J. Lee, “Chaotic local search algorithm,” *Artificial Life and Robotics*, vol. 2, no. 1, pp. 41–47, 1998.

[25] C.-s. Zhou and T.-l. Chen, “Chaotic annealing for optimization,” *Physical Review E*, vol. 55, no. 3, pp. 2580–2587, 1997.

[26] L. N. d. Castro and J. I. Timmis, “Artificial immune systems as a novel soft computing paradigm,” *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 7, no. 8, pp. 526–544, 2003.

[27] A. Charles, Janeway, and R. Medzhitov, “Innate immune recognition,” *Annual Review of Immunology*, vol. 20, pp. 197–216, 2002.

[28] S. A. Mohamed Elsayed, R. A. Ammar, and S. Rajasekaran, “Artificial Immune systems: models, applications, and challenges,” in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 256–258, New york, NY, USA, 2012.

[29] Y. Jiang, J. Zhou, Y. Gan, and Z. Cai, “A method of in-depth-defense for network security based on immunity principles,” in *Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 10–12, Chengdu, China, August 2009.

[30] G. Shi and Y. Jing, “Research of improved immune clonal algorithms and its applications,” in *Proceedings of IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pp. 212–215, Hong Kong, China, May 2009.

[31] H. Yang, T. Li, X. Hu, F. Wang, and Y. Zou, “A survey of artificial immune system based intrusion detection,” *The Scientific World Journal*, vol. 2014, Article ID 156790, 11 pages, 2014.

[32] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[33] J. C. Caicedo and S. Lazebnik, “Active object localization with deep reinforcement learning,” in *Proceedings of the International Conference on Computer Vision*, pp. 2488–2496, Santiago, Chile, December 2015.

[34] H. Satija and J. Pineau, “Simultaneous machine translation using deep reinforcement learning,” in *Proceedings of the Workshops of International Conference on Machine Learning*, pp. 110–119, York City, NY, USA, June 2016.

[35] T. Mikolov and G. Zweig, “Context dependent recurrent neural network language model,” in *Proceedings of the IEEE Spoken Language Technology Workshop*, pp. 234–239, Miami, FL, USA, December 2012.

[36] D. Yu and L. Deng, “Deep learning and its applications to signal and information processing exploratory DSP,” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[37] E. Shelhamer, J. T. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[38] R. Hojjat and R. Amin, “Snap-drift cuckoo search: a novel cuckoo search optimization algorithm,” *Applied Soft Computing Journal*, vol. 52, pp. 771–794, 2017.

[39] B. Y. Qu, J. J. Liang, Z. Y. Wang, Q. Chen, and P. N. Suganthan, “Novel benchmark functions for continuous multimodal optimization with comparative results,” *Swarm and Evolutionary Computation*, vol. 26, pp. 23–34, 2016.

[40] B. Ye, C. Z. Zhu, C. X. Guo et al., “Generating extended fuzzy basisfunction networks using hybrid algorithm,” in *Proceedings of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 79–88, Changsha, China, August 2006.

[41] R. Storn, “On the usage of differential evolution for function optimization,” in *Proceedings of the Biennial Conference of the NorthAmerican Fuzzy Information Processing Society*, pp. 519–523, Berkeley, CA, USA, 1996.

[42] M. R. Garey, D. S. Johnson, and R. Sethi, “The complexity of flowshop and jobshop scheduling,” *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[43] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, “A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 301–305, 2013.

Advances in
Operations Research
Advances in
Decision Sciences
Journal of
Applied Mathematics
The Scientific
World Journal
Journal of
Probability and Statistics
International
Journal of
Mathematics and
Mathematical
Sciences
Hindawi
Submit your manuscripts at
www.hindawi.com
Journal of
Optimization
International Journal of
Engineering
Mathematics
International Journal of
Analysis
Journal of
Complex Analysis
Advances in
Numerical Analysis
Mathematical Problems
in Engineering
International Journal of
Differential Equations
Discrete Dynamics in
Nature and Society
International Journal of
Stochastic Analysis
Journal of
Mathematics
Journal of
Function Spaces
Abstract and
Applied Analysis
Advances in
Mathematical Physics