

RESEARCH

Open Access



A chaotic simulated annealing and particle swarm improved artificial immune algorithm for flexible job shop scheduling problem

Rui Zeng^{1,2*} and Yingyan Wang¹

Abstract

Reasonable scheduling of flexible job shop is key to improve production efficiency and economic benefits; in order to solve the problem in flexible job shop scheduling problem, a novel flexible job shop scheduling method based on improved artificial immune algorithm is proposed. Firstly, a mathematical model of the flexible job shop scheduling is established, and the total shortest processing time is taken as the objective function. Secondly, artificial immune algorithm is used to solve the problem, and particle swarm optimization algorithm is taken as the operator to embed into manual immune algorithm for maintaining the diversity of population and prevent obtaining local optimal solution. Finally, the performance of the algorithm is tested by simulation experiments on standard set. The results show that the proposed algorithm can obtain better flexible job shop scheduling scheme and especially has more significant advantages in solving large-scale problems in comparison with other algorithms.

Keywords: Flexible job shop scheduling, Artificial immune algorithm, Mutation operator, Particle swarm optimization algorithm

1 Introduction

Job shop scheduling problem (JSP) refers to the ordering of available shared resource allocation and production tasks within a certain period of time, so that certain performance indicators can be optimized. The typical JSP model is more ideal. It is difficult to fully reflect the actual application. With the intensification of the marketization of manufacturing, the production resources are not infinitely usable, and the processing time of each process alone is not always the same. This results in the flexibility of the production process. The flexible job shop scheduling problem (FJSP) came into being; it is more in line with the actual production situation, but it also increases the complexity of the problem [1]. FJSP is a typical combinatorial optimization

problem, which belongs to the NP problem and has always been a key and difficult point in the research of the manufacturing industry [2].

For FJSP problems, a large number of scholars and researchers have invested a lot of time and energy in in-depth research, made some research progress, and proposed many FJSP problem solving algorithms [3]. The current FJSP problem solving algorithms can be roughly divided into three categories: exact solution algorithms, heuristic algorithms, and artificial intelligence algorithms. The branch-and-bound method is the most classic and accurate solution algorithm. It is simple and easy to implement. For small-scale FJSP problems, the solution efficiency is high; however, as the scale of the manufacturing industry continues to grow, the scale of processing products is getting larger and larger. The boundary method is difficult to meet the FJSP problem solving requirements of modern production and manufacturing [4]. The most representative heuristic algorithm is the Lagrangian relaxation method. Others

* Correspondence: jamse007@126.com

¹School of Electro-Mechanical and Information Technology, Yiwu Industrial and Commercial College, Yiwu, China

²College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China

have high efficiency in solving small-scale FJSP problems, but they have the same flaws as the exact solution algorithms, and their application scope is limited [5]. Artificial intelligence algorithm is mainly used to simulate the behavior of biological groups in nature. It has the advantages of parallelism, fast search speed, etc. It is the most widely used in FJSP problem solving and has become the main research direction [6]. Some scholars proposed the use of genetic algorithm, simulated annealing, particle swarm optimization algorithm, ant colony algorithm, and firefly algorithm to solve the FJSP problem and obtained better results [7–9]. However, in practical applications, these artificial intelligence algorithms all have their own deficiencies, such as the late convergence speed is slow, easy to obtain local optimal solution [10]. Artificial immune algorithm (AIA) is a kind of evolutionary algorithm that simulates the biological immune system. It is a multi-point random search algorithm. It has self-learning, self-organization, and memory under the premise of maintaining the excellent characteristics of the genetic algorithm. Such characteristics, so compared with other evolutionary algorithms, have better global search capabilities; have been successfully applied in cloud computing resource scheduling, robot path planning, and other fields; and provide a new research tool for solving FJSP problems [11].

In order to obtain a more optimal FJSP problem optimization program and reduce the production cost of the manufacturing industry, an improved FJSP problem solving method based on the modified artificial immune algorithm (MAIA) was proposed, and some standard examples were used to test the performance of the algorithm.

2 The mobile agent-based data fusion algorithm

2.1 Mathematical model of FJSP

The FJSP can be expressed as follows: n workpieces are processed on m machines, the processing time and production cost of each operation are known, and the processing order of each workpiece on each machine is constrained. The requirements are determined and the process constraints are required. The processing start time, completion time, and processing sequence of all the parts on the compatible machines make the production cycle, production cost, and equipment utilization process performance index optimal or suboptimal. The objective function of FJSP is

$$\min f_1 = \min(\max C_i) \quad (1)$$

$$\begin{aligned} \min f_2 &= \min F_p \\ &= \sum_{k=1}^m \sum_{j=1}^{n_j} \sum_{i=1}^{n_i} \left(F_{ijk} X_{ijk} + (T_{S_{ijk}} - T_{E_{i(j-1)k}}) F_{S_{ij}} X_{ijk} \right) \end{aligned} \quad (2)$$

$$\min f_3 = \min \sum_{k=1}^m \sum_{j=1}^{n_j} \sum_{i=1}^{n_i} p_{ijk} X_{ijk} \quad (3)$$

$$\min f_4 = \min \left(\max \sum_{j=1}^{n_j} \sum_{i=1}^{n_i} p_{ijk} X_{ijk} \right) \quad (4)$$

The corresponding constraint condition is

$$\begin{cases} \sum_{k=1}^m X_{ijk} = 1, X_{ijk} = 1 \text{ or } 0 \\ T_{S_{ijk}} = \max(T_{E_{i(j-1)k}}, t_{1k}) \\ \max(t_{1k}) + p_{ijk} \leq t_{2k}; \\ p_{ijk} \geq 0 \text{ and } p_{i0k} = 0 \\ i = 1, 2, \dots, n_i, j = 1, 2, \dots, n_j \\ k = 1, 2, \dots, m \end{cases} \quad (5)$$

In the function, C_i is the completion time of the work piece J_i ; F_p is the total processing cost; F_{ijk} is the processing cost of the j th process of workpiece i on the k machine tool; $T_{S_{ijk}}$ is the first processing time on the k machine tool for the j th process of workpiece i ; $T_{E_{i(j-1)k}}$ is the workpiece i in $(j-1)$ th process on the k machine end time; t_{1k} and t_{2k} are the start and end points of the idle time period of machine k , respectively; $F_{S_{ij}}$ is the unit time storage cost between the $(j-1)$ th and j th processes of the workpiece i ; p_{ijk} is the time required for the j th machining operation of the workpiece i on the k th machine; and X_{ijk} indicates whether the j th procedure of the workpiece i is selected to be machined on the k machine [12].

2.2 Improved artificial immune algorithm

Artificial immune algorithm (AIA) is an intelligent algorithm for simulating an artificial immune system, with good robustness, parallel search ability, and so on; its working principle is as follows: first, through the combination of antibodies and antigens and then through the antibody cloning, mutation, selection, and other operations, to achieve the optimal purpose of the antibody and antigen expression problem objective function. It is very suitable for solving a multi-objective optimization problem; the FJSP problem is a multi-objective optimization problem in essence, and the work steps of artificial immune algorithm are as follows:

1. Set the objective function and corresponding constraints of the multi-objective optimization problem.
2. Generate the initial antibody group of the artificial immune algorithm, that is, the candidate solution or feasible solution of the multi-objective optimization problem.
3. Calculate the affinity of the antibody and the antigen; the affinity can describe the degree of matching between the solution and the objective function.
4. Save the antibody with higher affinity and enter the next-generation antibody group to generate immune memory cells.
5. Selecting, promoting, and inhibiting antibodies to ensure the diversity of individuals in the antibody population.
6. Crossing and mutating individuals to create new antibody populations.
7. For individuals with low antibody-adaptive fitness values, individuals with high fitness values in memory cells are replaced to produce next-generation antibody populations.
8. If the algorithm satisfies the end condition, the algorithm is terminated; otherwise, the process jumps to step 3.

From the above, we can see that the workflow of the standard artificial immune algorithm is shown in Fig. 1.

2.3 Correlation operator of artificial immune algorithm

Artificial immune algorithm requires some operator during the work process, as follows:

1. Affinity evaluation operator between antibodies. Set the antibody as a viable solution to optimization

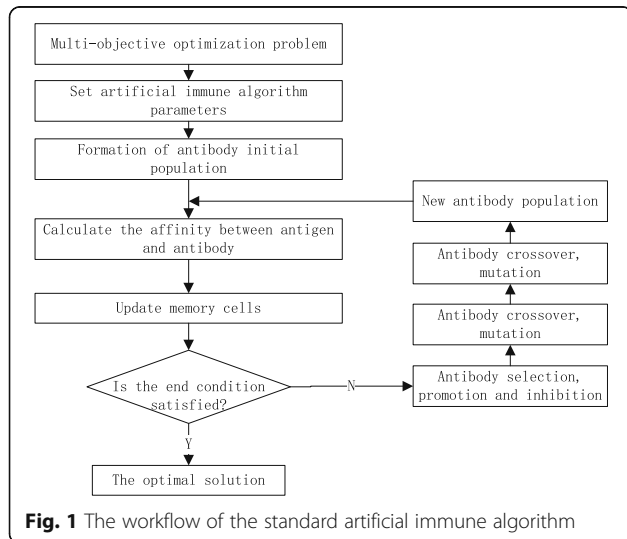


Fig. 1 The workflow of the standard artificial immune algorithm

problems. The affinity of the i th and j th antibodies is $aff(ab_i)$ and $aff(ab_j)$, respectively. The affinity evaluation operator is mainly used to describe the similarity of i and j ; the affinity evaluation formula is

$$aff(ab_i, ab_j) = \begin{cases} 1, & aff(ab_i) = aff(ab_j) \\ \frac{1}{1 + |aff(ab_i) - aff(ab_j)|}, & \text{else} \end{cases} \quad (6)$$

2. Antibody concentration evaluation operator. Set the individual of the antibody group as N . The calculation formula for the antibody concentration evaluation operator is

$$den(abi) = \frac{1}{N} \sum_{j=0}^{N-1} aff(ab_i, ab_j) \quad (7)$$

3. Mutation operator. The mutation operator is used for nascent antibodies to ensure multiple samples of individuals in the antibody group. The calculation formula is

$$\begin{cases} G'_N = G_M + \gamma \times \eta(0, 1) \\ \gamma = \frac{1}{\eta} \times e^{-f} \end{cases} \quad (8)$$

In the formula, G is genes of antibody; G' is genes of antigen; f is an affinity function; η is a control parameter; and $N(0, 1)$ is a Gaussian variable.

2.4 chaotic simulated annealing particle swarm parallel artificial immune optimization algorithm

Similar to other group intelligent algorithms, the standard artificial immune algorithm has the disadvantages of being easy to fall into local optimality and premature-ness. For this reason, the chaotic simulated annealing particle swarm optimization algorithm is introduced into the artificial immune algorithm. The chaos theory is used to dynamically adjust the parameters of the particle swarm optimization algorithm. At the same time, the dynamic inertia weight method is used to accelerate the convergence of the algorithm. The simulated annealing process based on the automatic attenuation coefficient is used to improve the probability and speed of searching for the optimal solution. The optimized artificial immune algorithm solves the problem of slow convergence in the late search period to ensure the diversity of the population and improves the search speed and efficiency of the algorithm, resulting in an improved artificial immune algorithm.

The execution process of chaotic simulated annealing particle swarm optimization algorithm consists of the following three parts.

1. Using the chaos theory to dynamically adjust parameters r_1 , r_2 of the particle swarm optimization algorithm, resulting in an excellent population;
2. Using the formulae (1) and (2) to search for the optimal solution in evolution;
3. Simulate the annealing algorithm to locally optimize the position of each particle in the particle swarm optimization algorithm and repeatedly run the iteration process until the termination condition is satisfied. The improved algorithm is shown in Fig. 2.

The steps of the chaotic simulated annealing particle swarm optimization algorithm are as follows:

1. Randomly generate initial populations of m particles, initialize each particle's velocity, and position, and give inertia weight ω , learning factors c_1 and c_2 , and initial acceptance probability P_r .
2. Calculate the fitness of each particle i and initialize the annealing temperature $T_0 = \frac{(f_{\min}^0 - f_{\max}^0)}{\ln P_r} = -\frac{|\Delta f|}{\ln P_r}$.
3. The fitness of each particle is taken as the particle extremum p_{best} , and the individual extremum is selected as the population extremum g_{best} .
4. If the algorithm reaches the termination condition, the result is output; otherwise, the cycle of k from 1 to M is performed, where M is the maximum number of iterations.

5. Calculate the fitness $f_i(k)$ and the average fitness $f_{\text{avg}}(k)$ for each particle.
6. If the particle's fitness is better than the original individual's extreme value p_{best} , then the current fitness is set to p_{best} and the optimal individual extreme value is chosen as the population extreme value g_{best} .
7. Update the flying position and velocity of each particle according to Eqs. (1) and (2).
8. Calculate the fitness of each new particle $f_i(k+1)$ and the average fitness $f_{\text{avg}}(k+1)$.
9. Calculate the amount of change in fitness caused by the two positions $\Delta f = f_i(k+1) - f_i(k)$. If $\Delta f < 0$ or $\exp(-\Delta f/T) > \text{rand}$, accept the new position; otherwise, keep the old position.
10. According to the individual fitness and average fitness, calculate the temperature automatic attenuation coefficient ζ .
11. $T_{k+1} = \zeta T_k$, $T_k + 1 = \zeta T_k$, $k = k + 1$, where $\zeta \in (0, 1)$, turn to (4).

2.4.1 Fitness parameter strategy

Using chaos to adjust the parameters related to particle velocity updates [13], when generating chaotic sequences, a logistic model is used:

$$\lambda_i^{t+1} = u\lambda_i^t(1-\lambda_i^t), i = 1, 2, \dots, n \quad (9)$$

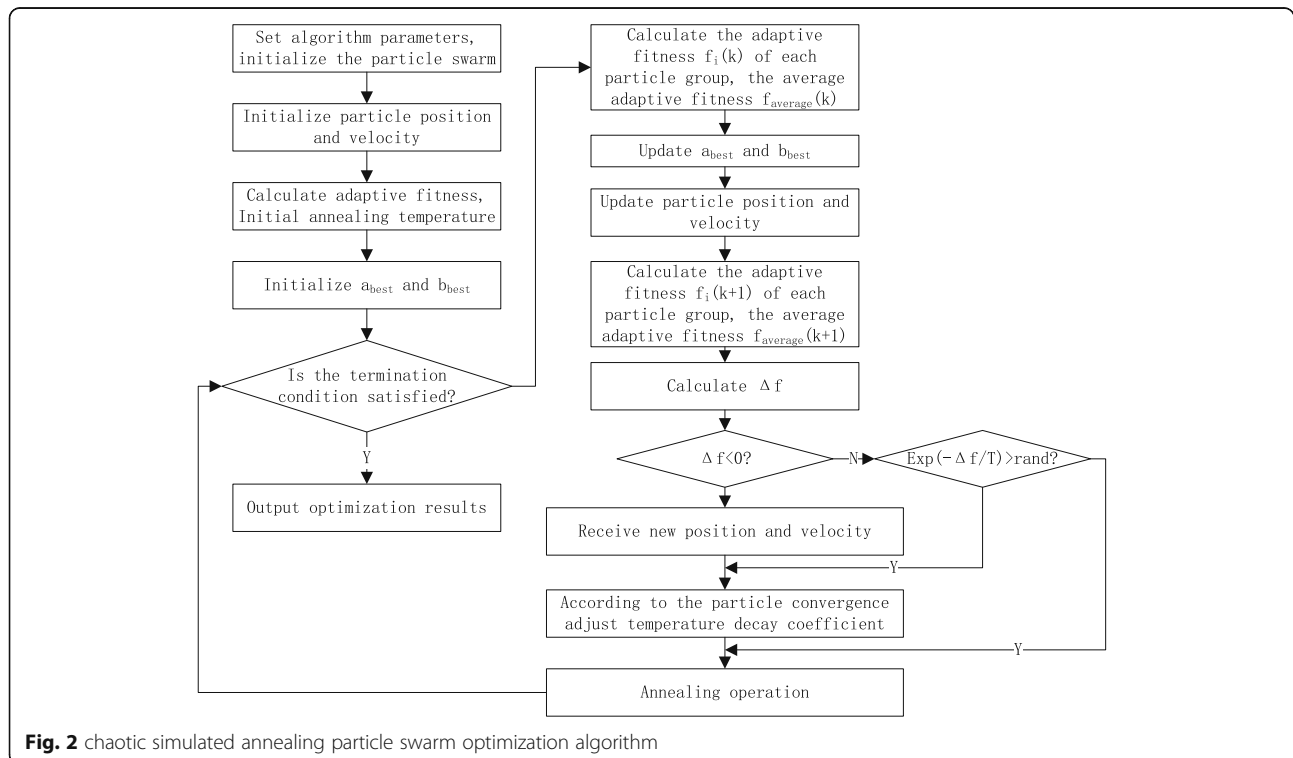


Fig. 2 chaotic simulated annealing particle swarm optimization algorithm

In the formula, λ_i^t is the value of the chaotic evolution of λ_i in step t , $\lambda_i \in [0, 1]$, $1 \leq u \leq 4$. When $u = 4$ and λ_i is not equal to 0.25, 0.5, or 0.75, the system exhibits full chaos. The chaotic sequence exhibits excellent randomness, and the trajectory of chaotic variable may not repeatedly traverse the entire search space. Perform chaos optimization on r_1 and r_2 as follows:

$$\begin{cases} r_i(t+1) = 4r_i(t)(1-r_i(t)) \\ r_i(t) \in (0, 1), i = 1, 2 \end{cases} \quad (10)$$

The inertia weight value ω can be used to balance the current velocity of the particle by controlling the historical velocity to balance the global and local search of the particle swarm optimization algorithm; at the same time, the appropriate ω can reduce the time to find the optimal solution. The dynamic adjustment mechanism of inertia weight is used to take a larger ω in the early stage of search. When the particle is searched in a large space, the algorithm has a good global search capability; as the number of iterations increases, ω gradually decreases, in the local area. The speed of particles gradually slows down to improve the search accuracy of the algorithm, so that the optimal solution can be found quickly and accurately. The inertia weight setting formula for each particle is as follows:

$$\omega(t) = \frac{\omega_{\max} - t(\omega_{\max} - \omega_{\min})}{t_{\max}} \quad (11)$$

In the formula, ω_{\max} and ω_{\min} are the maximum and minimum inertia weights, respectively; t is the current algebra; and t_{\max} is the maximum number of iterations.

2.4.2 Simulated annealing algorithm parameter control

2.4.2.1 Initial temperature setting The temperature initial value is an important influence parameter of the global search performance of the simulated annealing algorithm. The higher the initial temperature is, the stronger the global search ability is; and the greater the possibility of searching for the global optimal solution, the longer the search time. Conversely, although the search time is reduced, the global optimal solution may not be searched. Using a temperature initialization method based on fitness and acceptance probability, the initial temperature T_0 is determined by the following equation:

$$T_0 = \frac{(f_{\min}^0 - f_{\max}^0)}{\ln P_r} = -\frac{|\Delta f|}{\ln P_r} \quad (12)$$

In the formula, f_{\min}^0 , f_{\max}^0 , and Δf are the maximum and minimum target function fitness values and their differences calculated according to the initial particle

group, respectively, and p_r is the initial acceptance probability, and the general value is [0.7, 0.9].

2.4.2.2 Annealing velocity The global search ability of the simulated annealing algorithm depends on the annealing speed. In general, first set the initial temperature and then follow the temperature decay function to implement the cooling process. However, the fixed temperature decay function cannot perceive the current convergence condition and cannot dynamically adjust the local search depth according to the convergence condition. The dynamic temperature decay coefficient is introduced to enable the algorithm to perceive the local convergence according to the current fitness of the individual particles and the average fitness of the population. The temperature decay rate and the local search depth are dynamically adjusted according to the current conditions to ensure the diversity of the population in the search process. The formula for the adaptive temperature decay coefficient is:

$$\xi = \frac{\mu + N(0, 1) [1 - \exp(f_{pi} - f_{avg})]}{2T_k} \quad (13)$$

In the formula, f_{avg} is the average fitness value of the population; f_{pi} is the current particle's fitness; μ is the initial temperature attenuation coefficient; $N(0, 1)$ is a random number with a variance of 1 and a mean value of 0 Gaussian distribution; and T_k is the temperature of the particle in the previous iteration.

2.4.3 Design of artificial immune optimization algorithm based on chaotic simulated annealing particle swarm parallel algorithm

2.4.3.1 Cross-antibody acceptability of the algorithm

Set the antibodies $a_i = \{\phi_0, \dots, \phi_{n-1}\}$ and $a_j = \{r_0, \dots, r_{n-1}\}$; n represents the number of genes in an antibody. The cross-knot between them is k , and the resulting new antibodies are:

$$T_c(a_i, a_j) = \begin{cases} a'_i = \{\phi_0, \dots, \phi_k \in a_i, r_{k+1}, \dots, r_{n-1} \in a_j\} \\ a'_j = \{r_0, \dots, r_k \in a_j, \phi_{k+1}, \dots, \phi_{n-1} \in a_i\} \end{cases} \quad (14)$$

The sub-antibody $S = \{a'_i, a'_j\}$ is accepted according to the Metropolis guideline of the simulated annealing algorithm. $\min\{1, \exp(1 - \text{aff}(S) - \text{aff}(F)/T_k)\} > r_c$, where r_c is a random number, $F = \{a_i, a_j\}$ is the parent antibody, and T_k is the k th annealing temperature.

2.4.3.2 Variant antibody acceptability of the algorithm

Set the variant junction of antibody $a_i = \{\phi_0, \dots, \phi_{n-1}\}$ to k and the new antibody generated is:

$$T_m(a_i) = a'_i = \{\phi_0, \dots, \phi_k, \phi_{k+1}, \dots, \phi_{n-1}\}, \phi'_k \oplus \phi_{k-1} \quad (15)$$

The sub-antibody a'_i is accepted according to the Metropolis guideline of the simulated annealing algorithm. $\min\{1, \exp(1 - \text{aff}(a_i) - \text{aff}(a'_i)/T_k)\} > r_m$, where r_m is a random number and T_k is the k th annealing temperature.

2.4.3.3 Improvement in selecting factor calculation method Set the annealing selection probability of antibody $a_i = \{\phi_0, \dots, \phi_{n-1}\}$ to $P(\alpha_i)$, and its selection factor $\delta(\alpha_i)$ is calculated as:

$$\delta(a_i) = p(a_i) \times |A(t)| \quad (16)$$

In the formula, $|A(t)|$ is the scale of the population $A(t)$; $Str(\alpha_i)$ represents the enhancement degree. Its calculation formula is

$$Str(a_i) = \sum_{j=1}^n d_{ij}, d_{ij} = \begin{cases} 1 & \text{aff}(a_i) \geq \text{aff}(b_i) \\ 0 & \text{aff}(a_i) < \text{aff}(b_i) \end{cases} \quad (17)$$

The sub-antibody a'_i is accepted according to the Metropolis guideline of the simulated annealing algorithm, $\min\{1, \exp(1 - \text{aff}(a_i) - \text{aff}(a'_i)/T_k)\} > r_m$ where r_m is a random number and T_k is the k th annealing temperature.

In order to test the performance of the improved artificial immune algorithm (MAIA) and the standard artificial immune algorithm (AIA), a typical nonlinear function is selected to analyze its performance. These function tests are defined as follows:

$$f_1(x) = \exp(-(x_1-3)^2 - (x_2-5)^2) + \exp(-x_1^2 - x_2^2), |x| \leq 10 \quad (18)$$

$$f_2(x) = \exp(-(x_1-4)^2 - (x_2-4)^2) + \exp(-(x_1+4)^2 - (x_2-4)^2) + 2^* \exp(-x_1^2 - (x_2+4)^2) + 2^* \exp(-x_1^2 - x_2^2), |x| \leq 5 \quad (19)$$

$$f_3(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2 \times \pi \times x_i) + 10), |x_i| \leq 5.12 \quad (20)$$

$$f_4(x) = -e^{-(x-4)^2 - (y-4)^2} - e^{-(x+4)^2 - (y-4)^2} - 2e^{-x^2 - (y+4)^2} - 2e^{-x^2 - y^2} \quad (21)$$

$$f_5(x) = -0.5 + \frac{\sin \sqrt{x^2 + y^2} - 0.5}{(1.0 + 0.001(x^2 + y^2))^2} \quad (22)$$

$$f_6(x) = \sum_{i=1}^N [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (23)$$

$$f_7(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (24)$$

The above seven standard test functions have different shapes and can test the performance of the algorithm in an all-round way. $f_1 \sim f_3$ are continuous unimodal functions and are usually used to measure the convergence speed of the algorithm. From the convergence performance of the $f_1 \sim f_3$ functions, it can be concluded that compared to the AIA, the MAIA avoids the difficulty of the AIA being trapped in the local optimum, and the global search capability is stronger, which proves the feasibility of improving the AIA.

The functions $f_4 \sim f_7$ are complex nonlinear multi-peak functions, and there are a large number of local extrema, which are usually used to measure the population diversity and global search performance of the algorithm. From the performance tests of $f_4 \sim f_7$ functions, it can be concluded that compared with AIA, MAIA has fewer evolution iterations, better overcomes the disadvantages of AIA (such as easy to fall into local optimum and premature), and has higher convergence accuracy and speeds up convergence (Fig. 3).

3 Improved application of artificial immune algorithm in FJSP

3.1 Simulation parameters

In order to test and improve the performance of the artificial immune algorithm in the FJSP application, select four instances of FJSP:

- (1) 4 workpieces are processed on 6 machines.
- (2) 8 workpieces are processed on 10 machines.
- (3) 12 workpieces are processed on 9 machines.
- (4) 12 workpieces are processed on 10 machines

The application example adopts C++ and MATLAB mixed programming. The author uses Matlab to call the MEX program files written in the C programming language and complete the simulation experiment.

3.2 Results and analysis

3.2.1 Improve the effectiveness of artificial immune algorithm analysis

The experimental results of an application example based on an improved artificial immune algorithm are shown in Fig. 4. From Fig. 4, we can see that the improved artificial immune algorithm can get a better flexible job shop scheduling scheme. The simulation experiment results prove the effectiveness of the improved artificial immune algorithm in flexible job shop scheduling problems.

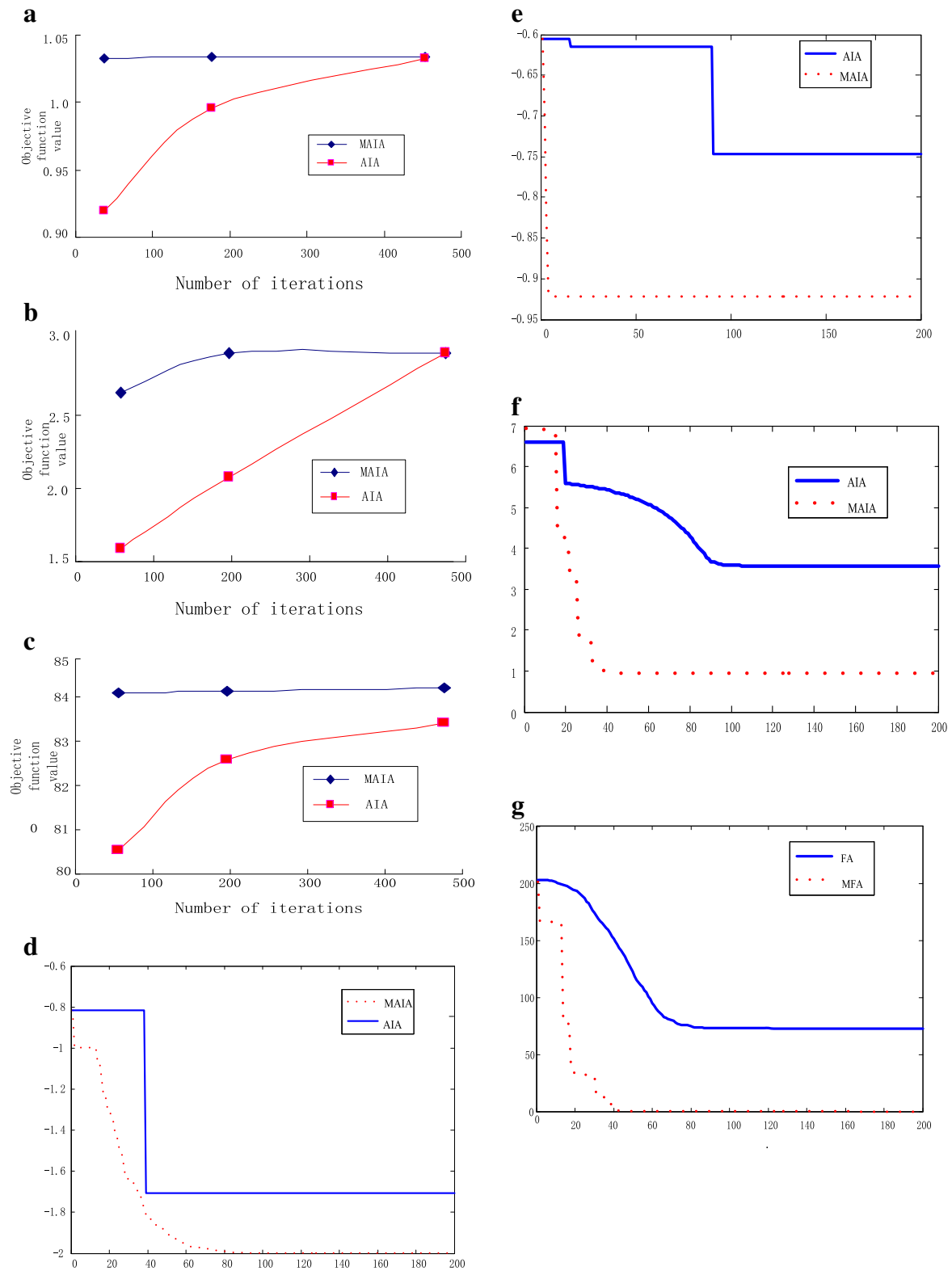


Fig. 3 Comparison of performance before and after improvement of artificial immune algorithm. **a** Comparison of the convergence curves of f_1 . **b** Comparison of the convergence curves of f_2 . **c** Comparison of the convergence curves of f_3 . **d** Performance comparison of f_4 . **e** Performance comparison of f_5 . **f** Performance comparison of f_6 . **g** Performance comparison of f_7

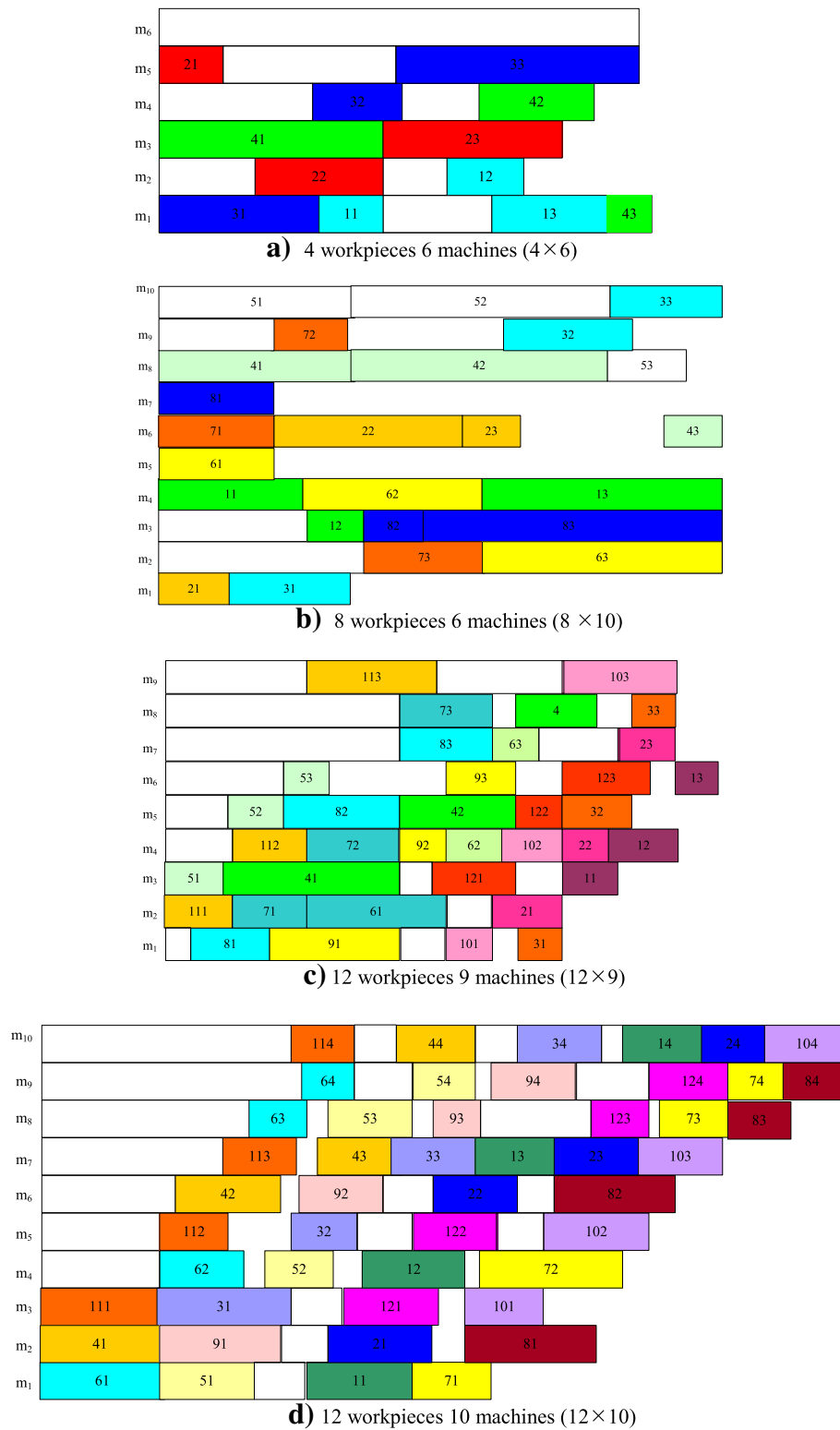


Fig. 4 Improved FJSP solution result of artificial immune algorithm. **a** 4 workpieces 6 machines (4×6). **b** 8 workpieces 6 machines (8×10). **c** 12 workpieces 9 machines (12×9). **d** 12 workpieces 10 machines (12×10)

Table 1 Performance comparison with current classical algorithms

The scale of FJSP problem	Number of iterations				Number of successes			
	Improved artificial immune algorithm	Literature [14]	Literature [15]	Literature [16]	Improved artificial immune algorithm	Literature [14]	Literature [15]	Literature [16]
4 × 6	100	160	163	164	100	98	95	94
8 × 10	250	376	365	393	100	91	90	91
12 × 9	300	539	560	534	90	73	76	75
12 × 10	380	536	696	542	88	53	60	63

3.2.2 Analysis of advantages of improved artificial immune algorithm

In order to test the superiority of the improved artificial immune algorithm, the FJSP classical algorithm of the literature [14–16] was selected for comparison experiments. All the experiments were performed 100 times. The improved artificial immune algorithm and the classical algorithm searched for flexibility. The number of optimal solutions for the job shop scheduling problem and the number of iterations are shown in Table 1. Analysis of the Table 1 shows that for a smaller flexible job shop scheduling problem. The success rate of all algorithms is relatively high and the number of iterations is also relatively small. For the smaller flexible job shop scheduling problems, such as 12 workpieces 10 machines, the success rate of the improved artificial immune algorithm is much higher than that of the current classical solution algorithm, and the number of iterations is relatively small. This algorithm improves the efficiency of solving the flexible job shop scheduling problem and proves that is superior of solving in FJSP.

4 Conclusions

FJSP is a major topic in the current production practice. There are many constraints, and the traditional intelligent algorithm has its own defects. This study proposes an chaotic simulated annealing and particle swarm improved artificial immune algorithm and applied to flexible job shop scheduling problem solving process. The experimental results show that the improved artificial immune algorithm's optimization ability and optimization efficiency are better than that of the standard artificial immune algorithm, and a better solution to the flexible job shop scheduling problem is obtained. Compared with the current typical flexible job shop scheduling problem, the algorithm is solved. Improving artificial immune algorithms has obvious advantages. How to introduce a better intelligent algorithm to obtain a better flexible job shop scheduling solution is the content that will be studied in the next step.

Abbreviations

AIA: Artificial immune algorithm; FJSP: Flexible job shop scheduling problem; JSP: Job shop scheduling problem; MAIA: Modified artificial immune algorithm

Acknowledgements

The research presented in this paper was supported by the Science Technology Department of Zhejiang Province, China.

Funding

The authors acknowledge the National Natural Science Foundation Project (Grant: 60972036), the National Education Information Technology Research "12th Five-Year" Program (Grant: 136241559), and the Science and Technology Program of Zhejiang Province (Grant: 2015C31171).

Authors' contributions

RZ is the main writer of this paper. He has proposed the main idea of the modified artificial immune algorithm for the flexible job shop scheduling problem algorithm and completed the simulation. YW had designed the fitness parameter stability and simulated annealing algorithm parameter. Both authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 13 March 2018 Accepted: 18 April 2018

Published online: 03 May 2018

References

1. D Pan, M Wang, Y Zhu, K Han, An optimization algorithm for locomotive secondary spring load adjustment based on artificial immune. *J. Cent. South Univ.* **20**(12), 3497–3503 (2013).
2. C Wang, X Lei, Protein-protein interaction network clustering based on artificial immune system. *J. Comp. App.* **33**(12), 3567–3570 (2013).
3. A Rezaee Jordehi, A chaotic artificial immune system optimisation algorithm for solving global continuous optimisation problems. *Neural Comput. & Applic.* **26**(4), 827–833 (2014).
4. A Karimi-Majid, M Fathian, B Amiri, A hybrid artificial immune network for detecting communities in complex networks. *Computing* **97**(5), 483–507 (2014).
5. B Naderi, M Mousakhani, M Khalili, Scheduling multi-objective open shop scheduling using a hybrid immune algorithm. *Int. J. Adv. Manuf. Technol.* **66**(5–8), 895–905 (2012).
6. R Liu, L Zhang, B Li, Y Ma, L Jiao, Synergy of two mutations based immune multi-objective automatic fuzzy clustering algorithm. *Knowl. Inf. Syst.* **45**(1), 133–157 (2014).
7. Y Xu, L Wang, S Wang, M Liu, An effective immune algorithm based on novel dispatching rules for the flexible flow-shop scheduling problem with multiprocessor tasks. *Int. J. Adv. Manuf. Technol.* **67**(1–4), 121–135 (2013).
8. G Qu, Z Lou, Application of particle swarm algorithm in the optimal allocation of regional water resources based on immune evolutionary algorithm. *Journal of Shanghai Jiaotong University (Science)*. **18**(5), 634–640 (2013).
9. C Kahraman, O Engin, I Kaya, RE Ozturk, Multiprocessor task scheduling in multistage hybrid flow-shops: a parallel greedy algorithm approach. *Appl. Soft Comput.* **10**(4), 1293–1130 (2010).

10. T Nichi, Y Hiranaka, M Inuiguchi, Lagrangian relaxation with cut generation for hybrid flow shop scheduling problems to minimize the total weighted tardiness. *Comput. Oper. Res.* **37**(1), 189–198 (2010).
11. CT Tseng, CJ Liao, A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Int. J. Prod. Res.* **46**(17), 4655–4670 (2008).
12. C Ziyu, Y Chunming, Y Mingshun, Application of new quantum particle swarm optimization algorithm to solve PFSP problem. *Technol. Innov. Manag.* **33**(2), 162–165 (2012).
13. J Zheng, P Yang, S Chen, G Shen, W Wang, Iterative re-constrained group sparse face recognition with adaptive weights learning. *IEEE Trans. Image Process.* **26**(5), 2408–2423 (2017).
14. SQ Liu, K Erhan, Scheduling trains with priorities: a no-wait blocking parallel-machine job-shop scheduling model. *Transp. Sci.* **45**(2), 175–198 (2011).
15. R Rubén, S Thomas, A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem. *Eur. J. Oper. Res.* **177**(3), 2033–2049 (2007).
16. TM Reza, S Nima, S Farrokh, A memetic algorithm for the flexible flow line scheduling problem with processor blocking. *Comput. Oper. Res.* **36**(2), 402–414 (2009).

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)