Application Research of Improved Particle Swarm Optimization Algorithm Used on Job Shop Scheduling Problem

Guangzhi Xu[1](✉) and Tong Wu[2]

[1] China Academy of Electronics and Information Technology, National Engineering Laboratory for Risk Perception and Prevention (NEL-RPP), Beijing 100041, China

[2] Department of Intelligent Manufacturing, Beijing Institute of Computer Technology and Application, Beijing 100854, China

Abstract. Aiming at the needs of the job shop to arrange production scheduling reasonably in the actual industry, an improved particle swarm optimization method is proposed to solve the job shop scheduling problem (JSSP). By analyzing the scheduling characteristics of the job shop and according to various resource constraints, a process-based coding and activity scheduling decoding mechanism suitable for particle swarm optimization is designed. Use the proportional mutation strategy and the ring topology structure to improve the traditional particle swarm optimization algorithm, and combine the elite selection learning strategy to retain excellent individual information which accelerate the convergence speed. The improved particle swarm optimization algorithm is used to solve standard instance problems of different scales, and the effectiveness of the algorithm is verified by comparing with other methods.

Keywords: Particle swarm optimization · Large-scale optimization · Job shop scheduling problem

1 Introduction

The job shop scheduling problem is one of the most representative and core scheduling problems, and has important practical significance for the advanced manufacturing industry. In recent years, it has attracted a large number of researchers to conduct in-depth research on this problem. Many effective optimization algorithms are proposed for the NP-hard problem of job shop scheduling. A distributed particle swarm optimization algorithm is proposed in Literature [1], using coding and decoding technology method to solve job shop scheduling problem, through simulation examples to verify the effectiveness of this algorithm, and then extended this method to other actual assembly systems. An ant colony algorithm with a local search mechanism is proposed to solve the dual-objective job shop scheduling problem, the purpose of seeking a balance between the two objectives [2]. An improved artificial bee colony algorithm is proposed to solve the more complex job shop scheduling problem [3]. Literature [4] mixed genetic algorithm and particle swarm optimization algorithm with Cauchy distribution to obtain an improved algorithm with better performance to solve flexible job shop scheduling problem. Literature [5] applied tabu search as a local search method, and combined with particle swarm optimization algorithm to establish an algorithm framework for solving job shop scheduling problems. A particle swarm optimization algorithm based on neighborhood structure is proposed which performs a meticulous search through critical path analysis technique. This algorithm is more effective for complex job-shop problems [6].

In this paper, an improved particle swarm optimization algorithm is proposed which is based on the ring topology, combined with the variable ratio mutation strategy and the elite learning strategy, according to a coding and decoding technology to solve job shop scheduling problem. So as to provide a new way for the research of job shop scheduling problem.

2 Mathematical Model of Job Shop Scheduling Problem

A typical job shop scheduling problem is described as follows: a scheduling system has n sets of workpieces to be processed J = {1, 2, …, n}, which can be set on m machines M = {1, 2, …, m} to complete the processing task. Each workpiece has m processing steps, named separately $O_{1m}, \ldots, O_{2m}, \ldots, O_{nm}$, where O_{11} represents the first step of workpiece 1, O_{1m} represents the mth step of workpiece 1, and so on to the mth step of workpiece n. Each workpiece has a predetermined processing route, and the processing operation time of the process of each workpiece on the corresponding machine is defined. The purpose of scheduling is how to reasonably arrange the processing order of all workpieces on each machine tool to satisfy the optimization constraints [7]. Workshop scheduling problems generally have two general constraints, which named order constraints and resource constraints. In the process of processing, in addition to satisfying the constraints, the following ideal conditions are usually assumed:

(1) The processing of the next process can be started only after the processing of the previous process of each workpiece is completed, and no process has the priority of preemptive processing;
(2) At the same time, a certain process of a workpiece can only be processed on one machine, and a lathe cannot process two processes at the same time;
(3) The processing time of each workpiece includes its preparation time and processing setting time, and the processing time remains unchanged;
(4) During the entire processing process, once the process is performed, it cannot be interrupted, and each lathe is fault-free and effective;
(5) Each workpiece cannot be processed multiple times on the same lathe, and all lathes handle different types of processes;
(6) Without special regulations, the processing time of the workpiece remains unchanged.

In a typical job shop scheduling problem, conditions 1 and 2 are the sequence constraints and resource constraints of the processing process, and the other conditions are the ideal processing conditions designed to construct a fault-free processing process [8].

In order to establish an ideal mathematical model, J = {1, 2, …, n} represents the workpiece set, M = {1, 2, …, m} represents the machine set, each workpiece has the same process, The process set is O = {1, 2, …, m}, and the workpiece Ji has O_{ij} processes, then the set of all processes is $I = \sum_{i=1}^{n}\sum_{j=1}^{m} O_{ij}$, where each process meets the sequence constraints. Set T_{ij} represent the processing time of step O_{ij}, F_{ij} is the completion time of step O_{ij}, and A(t) represents the set of steps being processed at time t. If the operation O_{ij} is to be processed on the machine M_k, it is expressed as $E_{ijk} = 1$, otherwise it is $E_{ijk} = 0$. Using minimized maximum completion time as the optimization goal, the mathematical model can be described as follows,

$$S = \min\left(\max_{i \in n}(\max_{k \in m} X_{ik})\right) \tag{1}$$

The constraints are

$$F_{ik} \le F_{ij} - T_{ij},\ \mathrm{k} \in P_{ij};\ \ i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, m \tag{2}$$

$$\sum\nolimits_{j \in A(t)} E_{ijk} \le 1,\ \ k \in m;\ \ t \ge 0 \tag{3}$$

$$F_{ij} \ge 0,\ i = 1, 2, \ldots,\ n;\ \ j = 1, 2, \ldots, m \tag{4}$$

where X_{ik} is the completion time of the workpiece i on the machine k. Formula (1) is the minimized the maximum workpiece completion time, which is makespan performance index. Equation (2) is used to guarantee sequence constraints between processes, and Eq. (3) indicates that each lathe can only process one process at the same time. Equation (4) ensures that all processes are completed.

A typical example is shown in Table 1, including 3 workpieces, 2 processes, and 2 machines in job shop scheduling problem.

Table 1. Processing timetable for job shop scheduling problem

Workpieces	Process	Machine and processing time	
		M1	M2
J1	O_{11}		2
	O_{12}	2	
J2	O_{21}	3	
	O_{22}		2
J3	O_{31}	1	
	O_{32}		1

Disjunctive graph models are generally used to describe job shop scheduling problems. For the job shop scheduling problem of n workpieces and m machines, the corresponding disjunctive graph model is represented by $G = (N, A, E)$, where N is the set of points composed of all the processing steps, including virtual operations 0 and n * m + 1 (respectively representing start processing and end processing); *A* represents a set of arcs connected by a sequence of prescribed constraints in the same workpiece. The connected arc is also called a directed arc and is represented by a solid line. The dotted line connects the extraction arcs between all processes on the same machine, and the extraction arcs of all machines form the set *E*.

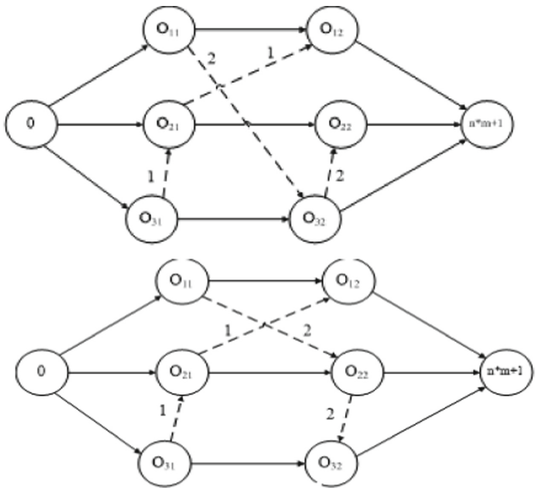

Fig. 1. The feasible disjunction diagram in Table 1

Figure 1 is an example of the JSSP problem extraction graph shown in Table 1, which represents two different feasible solutions. In the Fig. 1, the arc set 1 constitutes E1 which represents the sequence of processes on the machine 1, and the arc set 2 constitutes E2 which corresponds to the sequence of workpieces processed on the machine 2. It can be analyzed that the ultimate goal of scheduling is to find a solution that traverses all processes and satisfies the shortest path in set E. Optimizing the maximum completion time in the scheduling scheme is to make the longest path shorter in set E. As shown in the example, the longest path of the extraction graph b is 7 unit times in set E2 corresponding to machine 2, which is more unit time than E1 corresponding to machine 1. In order to reduce the spent time in set E2, find a new solution through the optimization scheme, such as inserting the original process O_{32} into the idle machine 2, which change the processing sequence on the machine 2. Shortening the longest path makes it possible to find an optimized solution in a shorter time.

Compared with the traveling salesman problem, the job shop scheduling problem has a more complicated situation, which is caused by the following reasons [9]:

1) The scope of the solution space is obviously increased. Assuming that there are n work pieces and m machines, which can combine $(n!)^m$ solutions. For example, a scheduling problem with a scale of 10 * 10, the solution space is expanded to 3.96 * 1065, if it traverses all the values in the solution space, it need to use

a computer that operates 100 million times per second to continuously work for 1.26 * 1050 years.

2) Due to the various constraints of the problem, the difficulty of encoding and decoding is increased, especially for the optimization algorithm for solving continuity problems, appropriate processing is required.
3) When solving hypersurface optimization problems, there are often more irregularly distributed suboptimal solutions. Due to the lack of prior information of the problem, it is more difficult to find the optimal solution to the problem.

3 Job Shop Scheduling Algorithm Based on Improved Particle Swarm Optimization

This paper proposes an improved particle swarm optimization algorithm for the job shop scheduling problem. First step, the process of the workpiece is coded. Next step, the discrete scheduling problem is converted into a continuous problem that can be solved by the particle swarm optimization algorithm. Then, the target area is searched according to an improved particle swarm optimization algorithm. Finally, the optimal solution obtained by the algorithm is activated decoding. Determine the processing sequence of the workpiece, and select the optimal processing procedure of the job shop scheduling problem according to the minimum completion time.

3.1 Encoding Operation

Encoding operation is an important process link of particle swarm optimization algorithm applied to workshop production scheduling problems. Without encoding operation, particle swarm optimization algorithm cannot be used to solve discrete scheduling problems. Therefore, this chapter uses a process-based coding technique to make continuous optimization algorithms for solving discrete problems. Through coding, each particle contains all the workpiece processes, which represents a feasible solution to the corresponding scheduling problem. Different discrete values are used to represent different workpieces, and the number of occurrences of the values indicates different processing procedures for each workpiece. For example, if a processing workshop needs m machines to process n workpieces, each particle is composed of n $\times$ m variable dimensions, and the variable dimensions in the particles represent the sequence-constrained process. After decoding operations, the machine pair can be obtained. The processing sequence of the workpiece process forming a feasible solution to the job shop scheduling problem.

For example, a 3 $\times$ 3 workshop scheduling problem consisting of 3 machines and 3 workpieces. After the encoding operation, one of the particles is [1 2 2 3 3 2 1 1 3], and the values 1, 2, and 3 represent different workpieces. One of the workpieces has 3 different processing steps, so each number is appearing 3 times. Use O_{ij} to represent the j-th processing step of the i-th workpiece, then the operation step indicated by the corresponding particle is [O_{11} O_{21} O_{22} O_{31} O_{32} O_{23} O_{12} O_{13} O_{33}].

Owing to the particle swarm optimization algorithm is an optimization algorithm for solving continuity problems, it is necessary to convert continuous real numbers into discrete codes to use the particle swarm optimization algorithm. First, sort the particles in ascending order, take the corresponding position index to form a matrix of the same size, divide all the elements in the matrix by the number of operations and take an integer, and then determine the dimensional order of the process of each workpiece in the particle, use a continuous series of ascending numbers indicate the process of the workpiece. The starting data of different workpieces is associated with the corresponding workpiece number, and a continuous code is formed according to the sequence of the processes. For example, a particle's process-based code is [1 2 2 3 3 2 1 1 3], and it becomes [1 4 5 7 8 6 2 3 9] by the above conversion method, where the value 1 indicates the first of the 1 workpiece. In the process, the value 2 represents the second process of 1 workpiece, the value 4 represents the first process of 2 workpiece, and so on, and the value 9 represents the third process of 3 workpiece. The values 1, 2, and 3 are used to represent the sequence constraints of 1 workpiece process, respectively, to avoid repeated occurrence of the values. The above method provides a way to solve discrete optimization problems using continuous intelligent optimization methods.

3.2 Particle Swarm Initialized

The initialized particle swarm should be widely distributed and cover most of the search space. Therefore, in this chapter, the initial population is generated by random distribution, then the initial population is normalized by coding technology, so that each initial particle meets the constraints of the actual problem, ensuring the feasibility and rationality of each particle. Randomly perturb of 20% particles and select the same number of particles to form the initial population. Expand the distribution of the initial population through the above methods [10].

3.3 Improved Particle Swarm Algorithm

In this part, the improved particle swarm algorithm based on the ring topology and proportional variation of the elite learning strategy is used to solve the scheduling optimization problem of the job shop.

Niching Methods with Ring Topology. Most existing niching methods usually suffer from the niching parameters adjustment, such as the species distance in species conserving genetic algorithm and the species radius in the speciation-based PSO. The niching radius should be set to neither too large nor too small. If it set too large, which would be very hard to capture the global optima exactly. On the other hand, if the niche radius set too small, these niches tend to prematurely converge. Relying on the niching radius is a main disadvantage for niching methods.

In this paper, it will prove that a PSO with ring topology can urge stable niching behavior and maintain diverse population without any niching parameters. In particular, one key advantage of this type of PSO algorithms is that there is no need to specify any niching parameters, which are usually required in existing traditional niching methods. In reducing the premature convergence of PSO, more ring communication topologies (two neighborhood members and one interacting member) have been shown to be very effective.

The appropriate PSO algorithm with a ring topology uses the local memory of particles to form a stable network. It reserves the best positions it has found so far and maintains solutions diverse through elitist learning strategy instead of one particle *p*best only. A ring topology is described as follows. One particle interacts with two neighboring particles to form an ecological niche. The first particle is the neighbor of the last one and vice versa. The neighborhood of the *i*-th particle returns the best-fit personal best solution $nbest_i$ in its neighborhood, which represents the neighborhood best for the *i*-th particle.

Different particles residing on the ring are possible to have different *nbests*. So it is possible to converge to different optima for different particles over time.

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 (pbest_{ij}^{k} - x_{ij}^{k}) + c_2 r_2 (nbest_{ij}^{k} - x_{ij}^{k}) \tag{5}$$

The ring topology neighborhood structure not only provides an opportunity to decelerate the rapid information propagation from the super individuals, but also makes different neighborhood bests to exist together (rather than becoming homogeneous) over time. The reason is that a particle's *nbest* will be updated only when there exists a better personal best around [11].

Elitist Select and Learning Strategy. It is obvious that PSO with ring topological structure supplies the opportunity for each particle to learn from its local niche best and the *p*best solutions. Correspondingly, the possibility of being trapped by local optimum will be decrease. However, owing to the natural difficulties of multimodal functions it is also liable to be trapped by local optima. Furthermore, particle's *p*best is used as the learning source of other PSOs. The promising particles are adopted used as the potentially exemplars to guide the particle flying. In traditional PSO algorithm, each particle will abandon its current *p*best solution if even better solution is existing. However, the abandoned solutions may also have better character and include the hopeful information about the global optimal solution. So as to make use of the useful historical information, one elitist set is constructed as an exemplar guidance pool.

The elitist set is composed by a history *p*best and other satisfactory suboptimal individuals with 10 individuals. The sub-optimal solutions record some abandoned individuals; however, the location is opposite to the *p*best solutions. After one iteration, the worst solution in elitist set is updated by the current particle or the *p*best with better function value. Each particle learns information from the elitist set (*eset*) randomly, which is described as (6).

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 (eset_{ij}^{k} - x_{ij}^{k}) + c_2 r_2 (nbest_{ij}^{k} - x_{ij}^{k}) \tag{6}$$

Mutation Strategy. A new hybrid mutation strategy is proposed which intends to maintaining the diversity and avoiding premature convergence. The proposed mutation strategy balances between exploration and exploitation by means of combining the mutation operation of differential evolution (DE) with the global search ability of PSO.

In traditional algorithms, large search space usually means higher risk for divergence, so it is an adventure to make particles have larger search space. Therefore, how to enlarge the exploitation areas of particles is an important improving direction for PSO. However, too large exploitation areas will mismatch the inherent need of algorithm to decrease the quality of particles. Mutation operation can preserve information through crossover operation in differential evolution. If a random number is larger than the crossover probability the component of the original position of the particle will be copied to the new individual. So, it is natural to combine both sides to utilize their advantages simultaneously. Conservatism and adventurism principle mutation, i.e., scaling mutation strategy, is defined as follows

$$l_j = \begin{cases} 0 \; if \quad s_j < -1 \\ 1 \; else \end{cases} \tag{7}$$

$$v_{ij}^{k+1} = wv_{ij}^k + c_1 r_1 (\text{eset}_{ij}^k - x_{ij}^k) + c_2 r_2 (lbest_{ij}^k - x_{ij}^k) \tag{8}$$

$$x_{ij}^{k+1} = x_{ij}^k + |s_j| \cdot l_j v_{ij}^{k+1} \tag{9}$$

where Gaussian random number s˜ N (0, 1), lj ∈ {0, 1}. If sj < −1, lj = 0, xk + 1ij = xkij. At this current conservatism principle procedure, the corresponding j-th component of the i-th particle will preserve the initial information from the previous position.

If sj ≥ −1, lj = 1. Currently, it becomes the adventurism principle, in which particles are possible to move to new positions with different coefficients. So, the search space is possible to be scaled by this mutation strategy according to the above analysis.

$$v_{ij}^{k+1} = wv_{ij}^k + c_1 r_1 (eset_{ij}^k - x_{ij}^k) + c_2 r_2 (nbest_{ij}^k - x_{ij}^k) \tag{10}$$

$$x_{ij}^{k+1} = x_{ij}^k + |s_j| \cdot l_j v_{ij}^{k+1} \tag{11}$$

3.4 Decoding Operations

The optimal solution for the job shop scheduling problem must be active scheduling, so this chapter uses active decoding operations to calculate the minimum maximum completion time by calculating the processing time of the process on each machine. The activity decoding operation flow is: first, determine the machine number of the workpiece process represented by each particle; then, calculate the start processing time and completion processing time of the workpiece process; then, according to the sequence constraints and resource constraints, the workpiece process is the corresponding machine starts processing at the earliest allowable processing time. After finishing all the workpiece processes, calculate the total processing time of each machine one by one, and select the longest time as the adaptation value.

This chapter uses the most commonly minimum maximum completion time as the optimization goal of the job shop scheduling problem, that is makespan. Through the decoding operation, the corresponding makespan value can be obtained.

4 Process Steps of JSSP-SERPSO Algorithm

Steps of the algorithm are as follows:

Step 1: Initialize the population and define control parameters of the elite library.
Step 2: Encode the generated particles, calculate the fitness value of each particle, and record the optimal value. Establish an elite library using the best 10 particles in the population.
Step 3: Construct a ring topology structure, use the left (right) particles of the current particle to form a niche neighborhood, and record the optimal particles in the neighborhood.
Step 4: Update each particle in the population according to formula (10) and (11), replace the current individual optimal particle with a random particle in the elite library, replace the global optimal particle with the optimal particle in the neighborhood, and mutate the dimension of the current particle using normal distribution.
Step 5: Calculate the optimal solution through decoding. If the iteration termination criterion is met, stop the iteration and output the optimal solution. If not, return to Step 2.

In Step 4, the replacement operation represents that the particles obtain update information from better elite particles and neighborhood particles in the population, while the mutation operation improves the diversity of the population and provide more effective and feasible solutions.

5 Experiments and Results Analysis

In order to verify the performance of the proposed JSSP-SERPSO algorithm, several experiments are developed using the standard JSSP test, including a total of 10 scheduling problems in LA (LA01, LA02, LA06, LA07, LA11, LA12, LA18, LA23, LA26, LA31). The goal is to minimize the maximum completion time of workpieces. The designed experiments compare the results of the proposed JSSP-SERPSO algorithm with other algorithms that are GA, LSGA, GRASP, PGA. Parameters of the JSSP-SERPSO algorithm are set as follows: the population size is set to n = 100; the size of elite library is 10; the maximum number of iterations is MaxIter = 10000. For the large-scale problems after LA18, the maximum number of iterations MaxIter = 30000. For each test problem, the experiment runs 20 times continuously. The parameter settings of the comparison algorithms are designed according to the original pattern.

Table 2 gives the optimization results of JSSP-SERPSO and some other algorithms in the literature. Best is for JSSP - SERPSO algorithm to get the optimal solution, "–" indicates no corresponding data. In order to try to eliminate the random factor, algorithms run independently 20 times, selecting average value as an indicator of algorithm, which is to make the algorithm more stable and effective. As Table 2 shows, JSSP-SERPSO can basically obtain the optimal solution for small-scale simples about LA01, LA02, LA06, LA07, LA11 and LA12 problems. Its average value can also reach the optimal solution, which has good stability. For the larger and more difficult problem of LA18, JSSP-SERPSO also achieves excellent results and achieves an optimal value. In the LA23 with a size of 15 $\times$ 10, JSSP-SERPSO may obtain the optimal solution, but its mean value cannot obtain the value of the optimal solution. JSSP-SERPSO in scale for LA26 problem, not to find the optimal value to solve the problem, according to the average index, the lack of certain algorithm local search ability which need to design a kind of auxiliary local search algorithm to improve the local search ability of the algorithm. In LA31 problem, JSSP-SERPSO can obtain the optimal value of the problem every time, which shows the effectiveness of the algorithm in dealing with larger scale scheduling problems.

Table 2. Comparison results between JSSP-SERPSO and other algorithms

Instance	Size	BKS	SERPSO		GA	LSGA	GRASP	PGA
			Best	Mean				
LA01	10 × 5	666	666	666	666	—	666	666
LA02	10 × 5	655	655	655	666	—	655	681
LA06	15 × 5	926	926	926	926	—	926	926
LA07	15 × 5	890	890	890	890	—	890	890
LA11	20 × 5	1222	1222	1222	1222	—	1222	1222
LA12	20 × 5	1039	1039	1039	1039	—	1039	1039
LA18	10 × 10	848	848	851	848	857	848	916
LA23	15 × 10	1032	1032	1039	1032	1047	1032	1072
LA26	20 × 10	1218	1246	1251	1231	1307	1271	1278
LA31	30 × 10	1784	1784	1784	1784	1784	1784	—

It can also be seen from the table that JSSP-SERPSO compares with other algorithms. Because the basic structure and operational mode design of GA algorithm are more suitable for the optimization of discrete problems, JSSP-SERPSO usually shows certain advantages in solving discrete problems. PSO algorithm is an optimization method for solving continuous problems, so coding and decoding are needed to facilitate. Therefore, PSO algorithm has certain disadvantages compared with GA in algorithm construction. However, this does not hinder the diversity of problem solving, so PSO optimization algorithm can be explored and applied in a broader field. GA algorithm solves scheduling optimization problems, whether for small scale problems or large scale problems, shows better optimization performance and strong stability, with excellent overall performance. Compared with GA, JSSP-SERPSO is only slightly superior to GA algorithm in LA02.

Table 2 gives the comparative results of JSSP-SERPSO, LSGA, GRASP and PGA. According to the average value of JSSP-SERPSO algorithm, LA18, LA23, and LA26 problems, JSSP-SERPSO has a superior optimization effect and achieves a smaller maximum completion time, while LSGA only obtains the same optimization performance in

LA31 problems. For GRASP algorithm, the optimal value obtained on problem LA26 is slightly lower than the average value obtained by JSSP-SERPSO algorithm, which basically achieves the same performance on other problems. However, the optimal value obtained by PGA on the small scale LA02 problem and the large scale LA18, LA23 and LA26 problem is not as good as the average value obtained by JSSP-SERPSO algorithm.

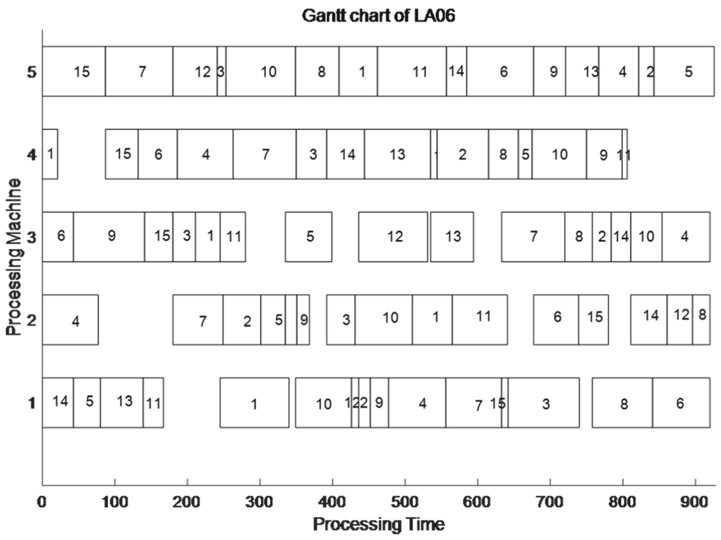

Fig. 2. Gantt chart of problem LA06

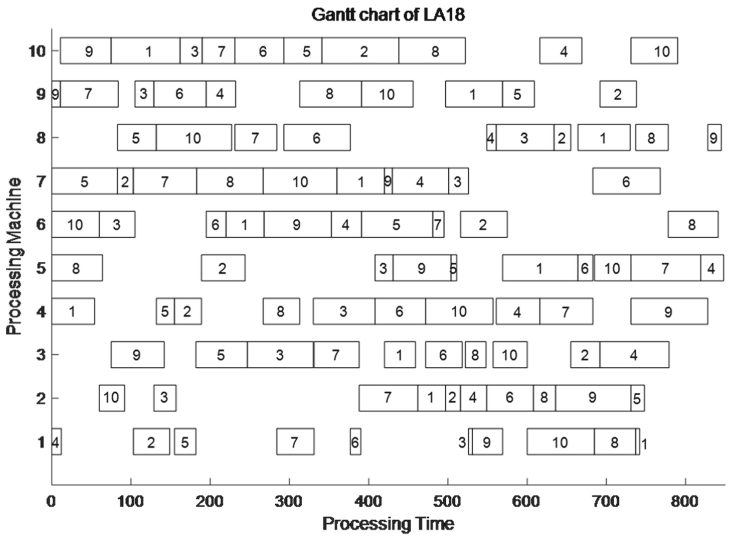

Fig. 3. Gantt chart of problem LA18

Figures 2, 3 give the JSSP-SERPSO Gantt chart for problems of LA06, LA18. As can be seen from the figure, the minimum maximum completion time was obtained in the whole processing process due to the adoption of the active decoding strategy.

The experimental results show that the JSSP-SERPSO algorithm proposed in this chapter achieves good results in solving job shop scheduling problem and provides an effective way for solving JSSP problem.

6 Conclusion

This paper proposes a new SERPSO algorithm, which combines ring topology and scale mutation operation to solve JSSP problem. Ring topology with niche structure increase the population diversity and avoid particles trapped in local optimum, which enhance the ability to jump out of local optimal trap particles, increase the effective solution of the search range. Combined with the elite library, particles with potential ability are used as learning objects for the next generation of particles, so that the better particles with good information can participate in the evolutionary process of the group. Through coding and decoding techniques, SERPSO algorithm can solve discrete scheduling problem. Ten standard JSSP test problems with small and large scale are adopted to evaluate the proposed algorithm and compare with other algorithms. The experimental results verify the optimization ability and effectiveness of the proposed algorithm for solving JSSP problems.

References

1. Nouiri, M., Bekrar, A., Jemai, A., Niar, S., Ammari, A.C.: An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. J. Intell. Manuf. **29**(3), 603–615 (2015). https://doi.org/10.1007/s10845-015-1039-3
2. Huang, R.H., Yu, T.H.: An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting. Appl. Soft Comput. **57**, 642–656 (2017)
3. Sharma, N., Sharma, H., Sharma, A.: Beer froth artificial bee colony algorithm for job-shop scheduling problem. Appl. Soft Comput. **68**, 507–524 (2018)
4. Jamrus, T., Chien, C.F., Gen, M., et al.: Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing. IEEE Trans. Semicond. Manuf. **99**, 1036–1052 (2017)
5. Hao, G., Kwong, S., Fan, B., et al.: A hybrid particle-swarm tabu search algorithm for solving job shop scheduling problems. IEEE Trans. Ind. Inf. **10**, 2044–2054 (2017)
6. Abdel-Kader, R.F.: An improved PSO algorithm with genetic and neighborhood-based diversity operators for the job shop scheduling problem. Appl. Artif. Intell. **32**(5), 433–462 (2018). https://doi.org/10.1080/08839514.2018.1481903
7. Zheng, X.L., Wang, L.: A collaborative multi-objective fruit fly optimization algorithm for the resource constrained unrelated parallel machine green scheduling problem. IEEE Trans. Syst. Man Cybern.: Syst. **48**(5), 790–800 (2018)
8. Cruz-Chávez, M.A.: Neighbourhood generation mechanism applied in simulated annealing to job shop scheduling problems. Int. J. Syst. Sci. **46**(15), 2673–2685 (2015). https://doi.org/10.1080/00207721.2013.876679
9. Spanos, A.C., Ponis, S.T., Tatsiopoulos, I.P., et al.: A new hybrid parallel genetic algorithm for the job-shop scheduling problem. Int. Trans. Oper. Res. **21**(3), 479–499 (2014)

10. Li, Y., et al.: Intelligent prediction of private information diffusion in social networks. Electronics **9**(5), 719 (2020)
11. Zhang, L., Li, Y., Liao, Y., Kong, R., Wu, W.: An empirical exploration of intelligence system based big data. J. CAEIT **11**(6), 608–613 (2016)