Recent Bio-inspired Algorithms for Solving Flexible Job Shop Scheduling Problem: A Comparative Study

Dongsheng Yang[1], Xianyu Zhou[1], Zhile Yang[2](✉), and Yanhui Zhang[2](✉)

[1] Intelligent Electrical Science and Technology Research Institute, Northeastern University, Shenyang 110819, China

[2] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518000, Guangdong, China

{zl.yang,yh.zhang}@siat.ac.cn

Abstract. Flexible job shop scheduling problem (FJSP) is an extended formulation of the classical job shop scheduling problem, endowing great significance in the modern manufacturing system. The FJSP defines an operation that can be processed by any machine from a given set, which is a strong constrained NP-hard problem and intractable to be solved. In this paper, three recent proposed meta-heuristic optimization algorithms have been employed in solving the FJSP aiming to minimize the makespan, including moth-flame optimization (MFO), teaching-learning-based optimization (TLBO) and Rao-2 algorithm. Two featured FJSP cases are carried out and compared to evaluate the effectiveness and efficiency of the three algorithms, also associated with other classical algorithm counterparts. Numerical studies results demonstrate that the three algorithms can achieve significant improvement for solving FJSP, and MFO method appears to be the most competitive solver for the given cases.

Keywords: Flexible job shop scheduling · Makespan · MFO · TLBO · Rao-2 algorithm

1 Introduction

The key component of production management for modern enterprises is effective production planning and scheduling. It is of great significance for reducing production costs, shortening production cycle and improving production efficiency. The job shop scheduling problem has the major characteristics including modeling complexity, computer complexity, dynamic randomness, multiple constraints, and multi-objectiveness.

The production planning and scheduling problem is to arrange the production tasks delivered on the equipment according to the sequence. It discusses how to arrange the processing resources and sequence of the operations under the premise of satisfying the processing constraints, aiming to minimize the product manufacturing time and the consumption cost. Due to the complexity of the production operation management system and various real-world constraints, the production scheduling problem becomes a NP-hard problem [6]. The classical job shop scheduling problem (JSP) assumes that there is no flexibility of the resources (including machines and tools) for each operation of the corresponding job. In another word, the problem requires that one machine only processes one type of the operation. However, in the real world application, many flexible manufacturing systems are used to improve the production efficiency in modern manufacturing enterprises [14].

In the light of this, the FJSP attracts increasing attentions from both research and industrial areas [2]. The FJSP can be divided into two sub-problems: the machine selection (MS) and the operations sequencing (OS), adding a more complicated scenario MS to the conventional JSP problem. Many methods have been proposed to solve this problem so far, including exact algorithm [3], dispatching rules [1], evolutionary algorithm (EA) [18], local search algorithms [7] and so on. For exact algorithm, Torabi et al. proposed a mixed integer nonlinear program for deterministic FJSP [16]. Roshanaei et al. presented two MILP models [13]. However, exact algorithm cannot obtain good results for large scale FJSP. So Tay and Ho used dispatching rules for multi-objective FJSP [15]. Ziaee proposed a construction procedure based heuristic for FJSP [19]. As for EA, the most used method is genetic algorithm, Pezzella et al. integrated GA with different strategies to solve FJSP [10]. Driss et al. proposed a new chromosome representation method and some novel crossover and mutation strategies for FJSP [4]. However, EA is lack of local search ability. So local search method is used in FJSP and tabu search (TS) is the most effective method for FJSP. Vilcot and Billaut used TS for the objective of minimum makespan and maximum lateness [17]. Jia and Hu proposed a TS based pathrelinking algorithm for multi-objective FJSP [7].

The remainder of this paper is structured as follows: the problem formulation of FJSP is discussed in Sect. 2, followed by the three compared algorithms that are briefly introduced in Sect. 3, where the encoding and decoding method are also given. Experimental studies and the corresponding discussion are reported in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Problem Formulation

The formulation of FJSP is described as follows: Assuming that there are M machines in the workshop, which need to process N jobs within the overall time period. Each job consists of a series of operations that allows them to be processed in a set of available machines. In this paper, the objective function is to minimize the maximal completion time, i.e. makespan (C_{max}), which is denoted as follows:
Notations for the formulation:

i number of jobs
j operations of the jobs
k number of machines

B_{ijk} starting time of operation j of job i on machine k
P_{ijk} processing time of operation j of job i on machine k
F_{ijk} completion time of operation j of job i on machine k
F_i completion time of job i

Objective function: Minimize C_{max}
Constraints:

- Jobs are independent and preemption or cancellation of jobs is not permitted.

$$B_{ijk} + P_{ijk} = F_{ijk} \tag{1}$$

$$\sum_k B_{ijk} \geq \sum_k F_{i(j-1)k} \tag{2}$$

- Every machine can only process one job at a time.

$$B_{ik} + P_{ik} \leq B_{(i+1)k} \quad \forall j \tag{3}$$

- One operation of each job can be processed by only one machine at a time.

$$B_{ij} + P_{ij} \leq B_{i(j+1)} \quad \forall k \tag{4}$$

- All jobs and machines are available at time zero and the transmission time between machines is ignored.

$$B_{ijk} \geq 0 \tag{5}$$

$$F_{ijk} \geq 0 \tag{6}$$

- Processing time is deterministic and includes other elements of set-up, transportation and inspection. the makespan is the maximal completion time of all jobs.

$$F_i \geq F_{ij} \quad \forall j \tag{7}$$

$$C_{max} \geq F_i \quad \forall i \tag{8}$$

3 Three Algorithms for FJSP

3.1 Algorithm Introduction

Moth-Flame Optimization Algorithm. Moth-flame optimization algorithm is a modern intelligent bio-inspired optimization algorithm proposed in 2015 [8]. It mimics the navigation mechanism of the moth in the space during the flight. Several advantages could be found in the algorithm design. First of all, due to that several flames is formulated and moths are considered flying around the individual flame and do not interfere with each other, the parallel optimization ability of the algorithm is strong. In addition, due to the spiral wrap path of the moth is assumed in the algorithm, as the number of iterations increases, it gradually approaches the contemporary flame center with a certain random amount. Such scheme avoids the whole population to be easily falling into the local optimal solution, and therefore guarantees the global optimal solution of the algorithm with excellent search performance and robustness.

The position update mechanism of each moth relative to the flame can be expressed by an equation:

$$M_i = S(M_i, F_j) \tag{9}$$

where M_i represents the ith moth, F_j represents the jth flame, and S represents the spiral function. The spiral function of the moth flight path is defined as follows:

$$S(M_i, F_j) = D_i e^{bt} \cos(2\pi t) + F_j \tag{10}$$

where D_i represents the linear distance between the ith moth and the jth flame, b is the defined logarithmic spiral shape constant, and the path coefficient t is a random number in $[-1, 1]$. The expression of D_i is as follows:

$$D_i = |F_j - M_i| \tag{11}$$

In order to reduce the number of flames in the iterative process and balance the global search ability and local development ability of the algorithm in the search space, an adaptive mechanism for the number of flames is proposed. The formula is as follows:

$$flame.number = round(N - G * \frac{N-1}{Gm}) \tag{12}$$

where G is the current number of iterations, N is the initial maximum number of flames, and Gm is the maximum number of iterations. Due to the reduction in flames, the moths corresponding to the reduced flames in each generation update their position based on the flame with the worst fitness value.

Teaching-Learning-Based Optimization Algorithm. The optimization algorithm based on 'teaching and learning' simulates the interaction of student-teacher in a class, which is a group intelligent optimization algorithm proposed in 2011 [12]. The improvement of the grades of students in the class requires the teacher's 'teaching'. In addition, the students need to 'learn' to promote the absorption of knowledge. Among them, teachers and students are both individuals in the evolutionary algorithms, and the teacher is the best individual in each iteration. The following formula shows the process of 'teaching':

$$X_i^{new} = X_i^{old} + difference \tag{13}$$

$$difference = r_i * (X_{best} - F_i * Popmean) \tag{14}$$

where X_i^{old} and X_i^{new} represent the values of the ith student before and after learning; X_{best} is the student who gets the best grades, e.g. the teacher; $Popmean = \frac{1}{N}\sum_{i=1}^{N}(X_i)$ is the average value of all students; N is the number of students; the teaching factor $F_i = round[1 + rand(0, 1)]$ and the learning step

$r_i = rand(0,1)$. After the 'teaching' phase is completed, the students are updated according to their grades e.g. fitness value. In the process of 'learning', for each student X_i, a learning object $X_j (j \neq i)$ in the class is randomly selected, and X_i adjusts himself by analyzing the difference between himself and the student X_j, with the formula is as follows:

$$X_i^{new} = \begin{cases} X_i^{old} + r_i * (X_i - X_j) & f(X_j) < f(X_i) \\ X_i^{old} + r_i * (X_j - X_i) & f(X_i) < f(X_j) \end{cases} \tag{15}$$

Also the students are updated according to their fitness values.

Rao-2 Algorithm. Rao-2 algorithm is a simple metaphor-less optimization algorithm proposed in 2019 for solving the unconstrained and constrained optimization problems [11]. The algorithm is based on the best and worst solutions obtained during the optimization process and the random interactions between the candidate solutions. The algorithm requires only common control parameters like population size and number of iterations and does not require any algorithm-specific control parameters. The individuals are updated according to the following formula:

$$\begin{aligned} X_{j,k,i}^{new} &= X_{j,k,i} + r_{1,j,i} * (X_{j,best,i} - X_{j,worst,i}) \\ &+ r_{2,j,i} * (|X_{j,k,i} or X_{j,h,i}| - |X_{j,h,i} or X_{j,k,i}|) \end{aligned} \tag{16}$$

where $X_{j,best,i}$ and $X_{j,worst,i}$ are the value of the variable j for the best and worst candidate during the ith iteration. $X_{j,k,i}^{new}$ is the updated value of $X_{j,k,i}$. $r_{1,j,i}$ and $r_{2,j,i}$ are the two random numbers for the jth variable in the range [0, 1]. $X_{j,k,i}$ and $X_{j,h,i}$ are the candidate solution k and any randomly picked candidate solution h. If the fitness value of the kth solution is better than that of the hth solution, the term '$X_{j,k,i} or X_{j,h,i}$' becomes $X_{j,k,i}$. And if the fitness value of the hth solution is better than that of the kth solution, the term '$X_{j,h,i} or X_{j,k,i}$' becomes $X_{j,k,i}$.

Then update the individuals according to their fitness values.

3.2 Encoding and Decoding

The method of encoding is shown as follows:

- The individuals are corresponding to the solutions of the FJSP, where each individual is a matrix of m rows and n columns, m is the number of the jobs, n is the number of operations for each job.
- Each element in the individual represents the machine used in the corresponding process.
- Initialize the individuals by selecting from the alternative machines randomly.
- During the iterations of the algorithm, each row of the individual is treated as a variable.

Table 1. The parameters settings of optimization algorithms and benchmark

Parameters	Value
Size of the population	400
Total number of generations	200
b of MFO	1
t of MFO	$(-1 - \frac{G}{Gm} - 1) * rand + 1$
Scale factor of DE	0.7
Crossover constant of DE	0.9
Inerita weight of PSO	$0.9 - 0.4 * \frac{G}{Gm}$

The procedure of decoding is as follows:

Step 1: Obtain the best individual which has the information on which machine to use for each operation of each job.
Step 2: Determine the allowable starting time for each operation which satisfies all the constrains mentioned above. Specifically, the starting time is the completion time of previous job processed on the same machine, but if the completion time of the last operation for the same job is longer than the completion time of previous job processed on the same machine, the starting time should be the completion time of the last operation.
Step 3: Determine the completion time for each operation, which should be the sum of starting time and processing time.
Step 4: Use the set of starting and completion time to paint the Gantt chart. The ith occurrences of the job number in the square represents the ith operation of the job.

4 Experimental Results and Discussions

The objective of this paper is to minimize the maximal completion time. The comparisons among the three algorithms and other algorithms are provided to compare the optimization performance. These algorithms are compared on one medium and one large size FJSP (MFJS01 and MFJS10). MFSJ01 represents that this problem has 5 jobs with 3 operations and 6 machines, MFSJ10 represents that this problem has 12 jobs with 4 operations and 8 machines. The algorithm terminates when the number of iteration reaches to the maximum generation value. The parameters for the two experiments are shown in Table 1.

The data of the experiment is adopted from literature [5]. Table 2 shows the experimental results and comparisons of these algorithms. To eliminate the randomness, 10 independent run is implemented for each problem. ‘Best’ represents the minimum value of makespan, and ‘Mean’ represents the average value of makespan. The results of HSA/SA, HSA/TS, HTS/TS, HTS/SA, ISA and ITS are adopted from literature [5] and [9]. The results with * are the best result for the given problem among these algorithms.

Table 2. The statistical results obtained by algorithms

Algorithm	MFSJ01 best	MFSJ01 mean	MFSJ10 best	MFSJ10 mean
MFO	469*	469*	1294*	1340.8*
TLBO	469*	469*	1368	1384.9
Rao-2	469*	469*	1445	1491.1
GWO	469*	469.8	1507	1553.6
DE	469*	469*	1430	1466.8
PSO	469*	469.8	1500	1549.7
HSA/SA [5]	479	503.2	1538	1621.8
HSA/TS [5]	491	504	1615	1693.4
HTS/TS [5]	469*	502.8	1404	1511.8
HTS/SA [5]	469*	499	1384	1428.2
ISA [9]	488	517.8	1546	1733.3
ITS [9]	548	584.2	1737	1737

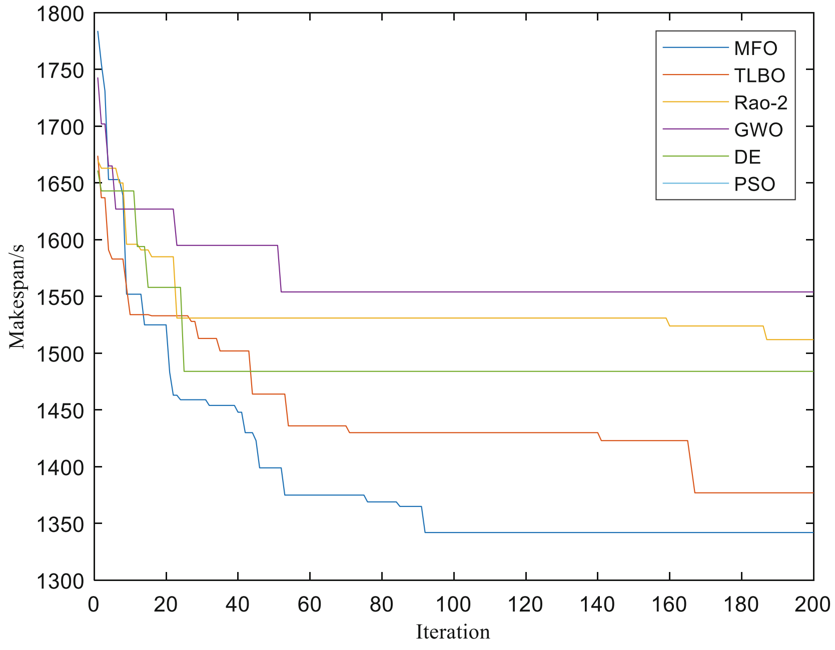

Fig. 1. Convergence results of the makespan for all the compared algorithms

It could be found in the Table 2 that MFO obtains all the best results for the two problems. Although other 7 algorithms can obtain the same results for problem MFSJ01 by 469, the average value may not as good as MFO. MFO also

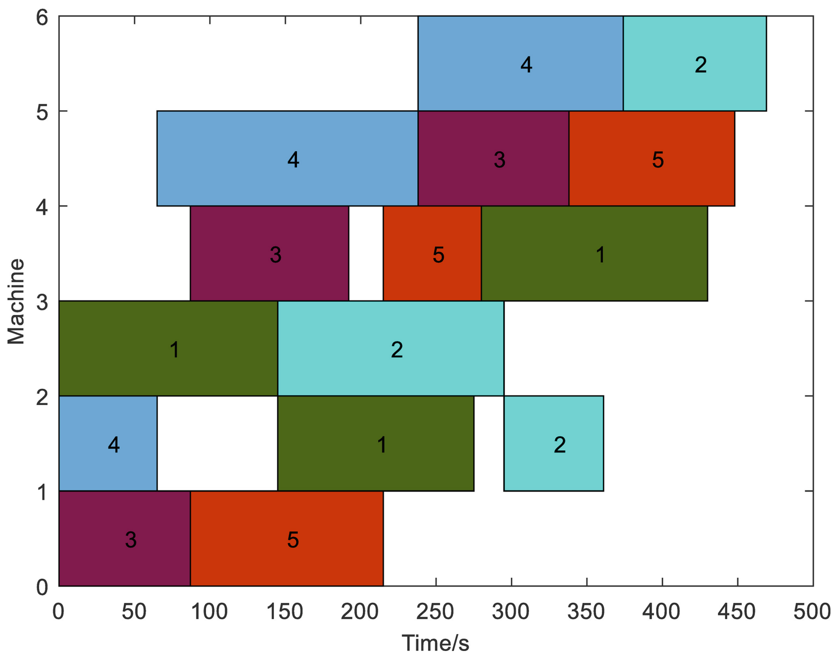

Fig. 2. Gantt chart of problem MFSJ01 of MFO

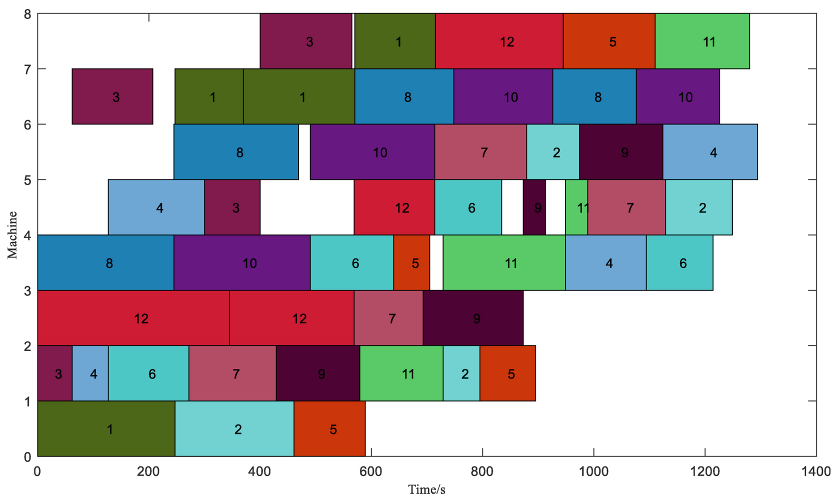

Fig. 3. Gantt chart of problem MFSJ10 of MFO

obtains the best result for problem MFSJ10. The results of TLBO for MFSJ10 is outperformed by MFO, but its result is better than other algorithms. In terms of the performance of Rao-2, it obtains the same best result for MFSJ01 as several other algorithms, it does not perform well in solving MFSJ10. This means that MFO has both good effectiveness and high efficiency for solving FJSP. Figure 1 shows the convergence results of the makespan of each algorithm in a featured run. It again shows that MFO converges fast and obtains the best result. Figures 2 and 3 illustrate the Gantt charts of the optimal solution obtained by MFO for problem MFSJ01 and MFSJ10 respectively.

5 Conclusion

In this paper, three algorithms including MFO, TLBO and Rao-2 are used for solving flexible job shop scheduling problem in comparing the optimization performance. FJSP is more complicated than the classical JSP with more constraints considered whereas more flexibility is endowed. The corresponding encoding and decoding methods is illustrated and two featured scales FJSP is introduced as the benchmarks to make the comparison. Through comprehensive results comparison of the three selected algorithms and some other popular method, MFO gets the best results for both problem MFSJ01 and MFSJ10, followed by TLBO. Rao-2 get the best result for problem MFSJ01 while performs pool in the larger scale problem MFSJ10. Future research will be addressing the further improvement for the algorithms variants and constraint handling methods, and more realistic objectives would be comprehensively considered.

Acknowledgment. This research work is supported by the National Key Research and Development Project under Grant 2018YFB1700500, and Science and Technology Project of Shenzhen (JSGG20170823140127645).

References

1. Baykasoğlu, A., Özbakır, L.: Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system. Int. J. Prod. Econ **124**(2), 369–381 (2010)
2. Chaudhry, I.A., Khan, A.A.: A research survey: review of flexible job shop scheduling techniques. Int. Trans. Oper. Res. **23**(3), 551–591 (2015)
3. Demir, Y., İşleyen, S.K.: Evaluation of mathematical models for flexible job-shop scheduling problems. Appl. Math. Model. **37**(3), 977–988 (2013)
4. Driss, I., Mouss, K.N., Laggoun, A.: A new genetic algorithm for flexible job-shop scheduling problems. J. Mech. Sci. Technol. **29**(3), 1273–1281 (2015). https://doi.org/10.1007/s12206-015-0242-7
5. Fattahi, P., Mehrabad, M.S., Jolai, F.: Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J. Intell. Manuf. **18**(3), 331–342 (2007). https://doi.org/10.1007/s10845-007-0026-8
6. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Math. Oper. Res. **1**(2), 117–129 (1976)

7. Jia, S., Hu, Z.H.: Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. Comput. Oper. Res. **47**, 11–26 (2014)
8. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl.-Based Syst. **89**, 228–249 (2015)
9. Özgüven, C., Özbakır, L., Yavuz, Y.: Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl. Math. Model. **34**(6), 1539–1548 (2010)
10. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible job-shop scheduling problem. Comput. Oper. Res. **35**(10), 3202–3212 (2008)
11. Rao, R.: Rao algorithms: three metaphor-less simple algorithms for solving optimization problems. Int. J. Ind. Eng. Comput. **11**(1), 107–130 (2020)
12. Rao, R.V., Savsani, V.J., Vakharia, D.P.: Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput.-Aided Des. **43**(3), 303–315 (2011)
13. Roshanaei, V., Azab, A., Elmaraghy, H.: Mathematical modelling and a meta-heuristic for flexible job shop scheduling. Int. J. Prod. Res. **51**(20), 6247–6274 (2013)
14. Seebacher, G., Winkler, H.: Evaluating flexibility in discrete manufacturing based on performance and efficiency. Int. J. Prod. Econ. **153**(4), 340–351 (2014)
15. Tay, J.C., Ho, N.B.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Ind. Eng. **54**(3), 453–473 (2008)
16. Torabi, S.A., Karimi, B., Ghomi, S.M.T.F.: The common cycle economic lot scheduling in flexible job shops: The finite horizon case. Int. J. Prod. Econ. **97**(1), 52–65 (2005)
17. Vilcot, G., Billaut, J.C.: A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem. Int. J. Prod. Res. **49**(23), 6963–6980 (2011)
18. Yuan, Y., Xu, H.: Multiobjective flexible job shop scheduling using memetic algorithms. IEEE Trans. Autom. Sci. Eng. **12**(1), 336–353 (2013)
19. Ziaee, M.: A heuristic algorithm for solving flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology **71**(1–4), 519–528 (2013). https://doi.org/10.1007/s00170-013-5510-z