Int j simul model 17 (2018) 3, 534-542
ISSN 1726-4529 Original scientific paper

MODIFIED BINARY PARTICLE SWARM OPTIMIZATION ALGORITHM IN LOT-SPLITTING SCHEDULING INVOLVING MULTIPLE TECHNIQUES

Zhang, H. P.[*,#]; Ye, J. H.[**]; Yang, X. P.[***]; Muruve, N. W.[*] & Wang, J. T.[****]
[*] North China University of Water Resources and Electric Power, Zhengzhou 450046, China
[**] Henan University of Economics and Law, Zhengzhou 450046, China
[***] Nanchang Institute of Technology, Nanchang 330099, China
[****] Henan Juzhixiang Construction Co., Ltd, Zhengzhou 450016, China
E-Mail: zhanghuaping@ncwu.edu.cn, yejianhua99@sina.com, yxp1189@sina.com, nmuruve@gmail.com, 497025740@qq.com ([#] Corresponding author)

Abstract
This paper aims to strike a balance between cost, time and quality of multi-technique, multi-process flexible job-shop scheduling problem (FJSP) and thus improve the overall performance of the FJSP model. For this purpose, a bi-objective planning model was established for multi-technique, multi-process FJSP according to the multi-objective planning method in operational theory. Then, the structure of solution was designed for the established model, and the binary particle swarm optimization (BPSO) algorithm was modified to simulate the proposed method. Through the simulation, the following conclusions were put forward: (1) The established bi-objective planning model for multi-technique, multi-process FJSP meet the general requirements of manufacturing enterprises, laying the basis for further modification and derivation; (2) The modified BPSO (MBPSO) algorithm can solve the said model in a stable, fast and accurate manner. The research findings shed theoretical new light on the solution and application of real-world multi-technique, multi-process FJSP and provide practical guidance for the flow shop enterprises to improve their scheduling plans.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Multi-Technique, Multi-Process Flexible Job-Shop Scheduling Problem, Modified Binary Particle Swarm Optimization Algorithm, Largescale Batch Production

1. INTRODUCTION

Nowadays, manufacturing has developed into an industry for largescale, efficient batch processing involving multiple techniques and processes. The traditional multi-variety, single-piece and small-batch production methods could no longer adapt to the fast development of the manufacturing industry. Against this backdrop, the intelligent manufacturing must keep pace with the constant updates of manufacturing hardware. Actually, the intelligent manufacturing has long been applied in traditional manufacturing. Whether it is the traditional job-shop scheduling problem (JSP) or the flexible shop-shop scheduling problem (FJSP), many planning theories and solving algorithms have been integrated into these classic problems of operational research. Based on the traditional multi-technique, multi-process FJSP, this paper probes into the optimization of the equal-size lot-splitting JSP involving numerous jobs. The research findings have great theoretical and practical significance for largescale multi-technique and multi-process manufacturing enterprises [1-4].

Scheduling refers to the optimization of the overall performance under multiple constraints through the overall planning of performance indices and rational arrangement of job sequence. Being a small branch under the scheduling research, the JSP means the realization or approximation of the pre-set production goals under the constraints like time, resource quantity, cost control and quality standard through rational utilization of resources according to relevant operational theories. By the production mode, the JSPs fall into the

following categories: (1) Single-machine scheduling: every job in the job-shop must be processed by a single machine for a fixed processing time; (2) Parallel-machine scheduling: every job in the job-shop is processed by a series of machines with the same functions, which is the only sequence of the entire processing process; (3) Flow shop scheduling: every job in the job-shop is processed by a series of machines with different functions following a fixed sequence; (4) JSP: every job in the job-shop is processed by a series of machines with different functions; each job requires unique techniques and sequence; the makespan differs with the job sequences on each machine; (5) FJSP: the machines in the sequence of each job are not uniquely specified in this extension of the JSP [5-11].

Considering the above, this paper establishes a model for multi-technique, multi-process FJSP, and simulates the model with the improved binary particle swarm optimization (PSO) algorithm, aiming to strike a balance between cost, time and quality and optimize the overall performance of the model.

The remainder of this paper is organized as follows: Section 2 creates a mathematical model for largescale lot-splitting scheduling involving multiple techniques and processes; Section 3 improves the traditional PSO algorithm based on the discrete search space and designs the structure of the solution to the multi-technique, multi-process FJSP; Section 4 simulates the established model with the improved binary PSO (BPSO) algorithm; Section 5 wraps up this paper with some valuable conclusions.

2. PROBLEM DESCRIPTION

Fig. 1 compares the process flow of three traditional scheduling strategies. The process flow of the traditional JSP is shown in Fig. 1 a. In this production mode, it is assumed that each job can only be processed by the same technique following a fixed sequence. Thus, the traditional JSP is also known as single-technique, single-process flow line problem. As shown in Fig. 1 b, the FJSP assumes that each job, provided with only one sequence, can be processed on multiple machines. Therefore, the FJSP is sometimes called single-technique, multi-process JSP. Figs. 1 c and 1 d describe the FJSP with multiple techniques and processes (MPF-JSP). In the MPF-JSP, it is assumed that each job can be processed by more than one technique and process. Here, the equal-size lot-splitting FJSP involving multiple jobs, objectives and techniques is modelled and simulated.

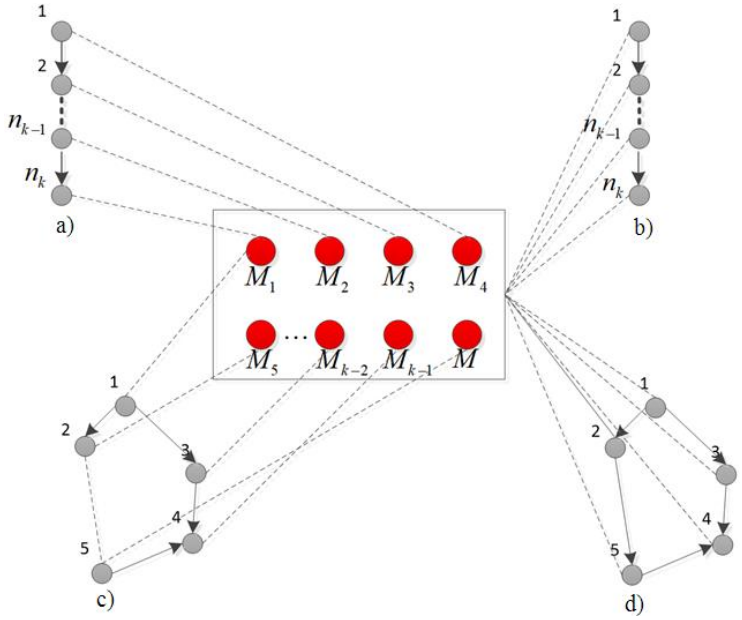

Figure 1: Comparison between JSP, FJSP and MPF-JSP.

2.1 Nomenclature

N – the total number of items i;
M – the total number of machines to process items i;
P_i – the products i, $i \in (1, N)$;
Q_i – the total number of products i, $i \in (1, N)$;
A_i – the number of processing batches of item i, $i \in (1, N)$;
L_i – the number of processing batches in J_i, $i \in (1, H)$;
C_i – the number of batches of moving units in item i, $i \in (1, N)$;
D_i – the number of batches of moving units in J_i, $i \in (1, H)$;
B_i – the number of processing batches of item i, $i \in (1, N)$;
O_i – the number of moving units in item i, $i \in (1, N)$;
H – the total number of processing batches;
J_i – the i^{th} processing, $i \in (1, H)$;
T_i – the time to complete the last process in the i^{th} processing, $i \in (1, H)$;
G_i – the number of moving units in the i^{th} processing, $i \in (1, H)$;
St_{irjk}, Ct_{irjk} – the time to adjust the k^{th} machine and the time to process a single item for the j^{th} process in the i^{th} processing, $i \in (1, H)$, $j \in (1, n_i)$, $k \in (1, M)$;
Ps_{irjk}, Pc_{irjk} – the cost to adjust the k^{th} machine and the cost to process a single item for the j^{th} process in the i^{th} processing, $i \in (1, H)$, $j \in (1, n_i)$, $k \in (1, M)$;
S_{irhjk}, E_{irhjk} – the time to start and the time to complete the j^{th} process in the i^{th} processing of the h^{th} moving unit, $i \in (1, H)$, $j \in (1, n_i)$, $h \in (1, G_i)$, $k \in (1, M)$;

$$x_{irjk} = \begin{cases} 1, & \text{if the } j^{\text{th}} \text{ process in the sequence } r \text{ of } J_i \text{ is executed on the } k^{\text{th}} \text{ machine} \\ 0, & \text{otherwise} \end{cases}$$

$i \in (1, H)$, $j \in (1, n_i)$, $k \in (1, M)$;

$$x_{irjk} = \begin{cases} 1, & \text{if both the } j^{\text{th}} \text{ process in the sequence } r \text{ of } J_i \text{ and the } g^{\text{th}} \text{ process} \\ & \text{in the sequence } r_2 \text{ of } J_f \text{ are executed on the } k^{\text{th}} \text{ machine, and} \\ & \text{the } j^{\text{th}} \text{ process precedes the } g^{\text{th}} \text{ process} \\ 0, & \text{otherwise} \end{cases}$$

$i, f \in (1, H)$, $j \in (1, n_i)$, $g \in (1, n_f)$, $k \in (1, M)$.

The known conditions are as follows:
$B_i = Q_i / A_i$, $i \in (1, N)$, A_i is a divisor of Q_i;
$O_i = A_i / C_i$, $i \in (1, N)$, C_i is a divisor of A_i;
$H = \sum_{i=1}^{N} B_i$;
$L_i = A_j$ for the processing batches i of item j;
$G_i = O_j$ and $D_i = C_j$ for the processing batches of item j.

2.2 Multi-objective optimization model

Let M and N be the number of machines and that of jobs in a job-shop, respectively. If there are more than one process available for job i, the following hypothesis can be put forward: (1) the numerous jobs are subjected to equal-size lot-splitting; (2) the processing time is fixed for each machine; (3) the adjustment time is known for each machine; (4) the entire processing is continuous with no breakpoints. Under the above hypotheses, the techniques and processes were rationally selected and arranged by an intelligent optimization algorithm, aiming to optimize several performance indices under multiple constraints.

Multi-objective optimization problems often involve such three indices as time, cost and benefit. For batch processing FJSP, the processing technique is optional, that is, the machine, adjustment time, processing cost and product quality vary with the processes. Meanwhile, the

time, cost and quality will also change with the sequence when the process flow is fixed. In actual production, product quality is the most important goal, followed by the minimization of time and cost. Therefore, the objective function of this paper must be established by the following principle: product quality should be the top priority in the selection of process and sequence, and time and cost should also be optimized. By this principle, the objective function can be established as:

$$\min Makespan = \min\left(\max_{i=1}^{H} T_i \right) \tag{1}$$

$$\min C_m = \min(C_s + C_c) = \min\left(\sum_{k=1}^{M}\sum_{i=1}^{H}\sum_{j=1}^{n_i} X_{irjk} \times Ps_{irjk} \times St_{irjk} + \sum_{k=1}^{M}\sum_{i=1}^{H}\sum_{j=1}^{n_i} X_{irjk} \times Pc_{irjk} \times Ct_{irjk} \times L_i \right) \tag{2}$$

S.T.

$$E_{irhjk} - E_{irh(j-1)m} \geq Ct_{irjk} \times D_i \quad \text{when} \quad X_{irjk} = X_{ir(j-1)m} = 1, k \neq m \tag{3}$$

$$E_{ir1jk} - E_{irG_i(j-1)m} \geq St_{irjk} + Ct_{irjk} \times L_i \quad \text{when} \quad X_{irjk} = X_{ir(j-1)m} = 1, k = m \tag{4}$$

$$E_{fr_2G_fgk} - E_{ir_1G_ijk} \geq St_{fr_2gk} + Ct_{fr_2gk} \times L_f \quad \text{when} \quad X_{fr_2gk} = X_{ir_1jk} = 1, R_{ir_1jfr_2gk} = 1 \tag{5}$$

$$E_{irG_ijk} - S_{ir1jk} \geq St_{irjk} + Ct_{irjk} \times L_i \quad \text{when} \quad X_{ijk} = 1 \tag{6}$$

3. IMPROVED PSO (MPSO)

The PSO is originally attributed to Kennedy and Eberhart and was first intended for simulating social behaviour, as a stylized representation of the movement of organisms in a bird flock. For a specific problem, a potential solution is viewed as a bird or a particle. Each particle searches for its position in a *D*-dimensional search space based on its own experience and the experience of other particles. In the standard PSO algorithm, a particle is described by a vector *p* and a velocity *v*. At time *t*, particle *i* first calculates the new velocity by Eq. (7) and then updates its position byEq. (8).

$$v_i^{'d} = wv_i^d + c_1 r_1 \left(p_i^d - x_i^d \right) + c_2 r_2 \left(p_g^d - x_i^d \right) \tag{7}$$

$$x_i^{'d} = x_i^d + v_i^{'d} \tag{8}$$

where w is the inertia weight of velocity; c_1 and c_2 are the self-cognition coefficient and the social learning coefficient, respectively; $v_i{}^{'d}$ and $v_i{}^d$ are the current velocity and the previous update on velocity, respectively; $p_i{}^d$ and $p_g{}^d$ are the best-known position of particle i and that of all particles in the d-dimensional search space, respectively; r_1 and r_2 are random variables in [0, 1].

The standard PSO algorithm is suitable for searching continuous space. However, most combinatorial optimization problems face a discrete-continuous search space. This calls for improvement of the standard PSO. In 1997, Kennedy and Eberhart proposed the BPSO based on the discrete search space, which works well in solving discrete combinatory optimization problems [12-14].

3.1 Basic BPSO

The basic BPSO first updates the velocity according to Eq. (9) using the binary variables $v_i{}^d$, $p_i{}^d$, $p_g{}^d$ and $x_i{}^d$, then converts the velocity into a binary probability of 1 by the sigmoid function and maps it into the interval of (0, 1), and changes the current binary bit state by determining the random values:

$$s\left(v_i^d\right)=\frac{1}{1+\exp\left(-v_i^d\right)} \tag{9}$$

$$x_i^d=\begin{cases}1 & \text{if rand}(\ )\le s\left(v_i^d\right)\\ 0 & \text{otherwise}\end{cases} \tag{10}$$

where *rand*() is a random number uniformly distributed in (0, 1). To prevent $s(v_i^d)$ from being too close to 0 or 1, a maximum velocity $v_{\max}$ was set to limit the range of v_i^d, $v_i^d \in [-v_{\max}, v_{\max}]$. Despite this fast convergence, the standard PSO often falls into the local optimum trap and stops from looking for the global optimal solution. To overcome this defect, the standard BPSO algorithm was improved to ensure the efficient, accurate search for the global optimal solution [15-17].

3.2 Structure and steps of MBPSO algorithm

(1) Structure of the solution

The structure of the solution directly bears on the efficiency and quality of the algorithm. It must be designed specifically for each problem model. Besides, the coding format should be determined according to the pre-set goal for each model. Here, the binary code is adopted because our model is solved by the BPSO algorithm with multiple Boolean variables. In Fig. 2, the matrix containing *D* submatrices is one of the solutions for all populations in the PSO. In other words, the number of matrices in Fig. 2 equals the number of populations in the PSO.

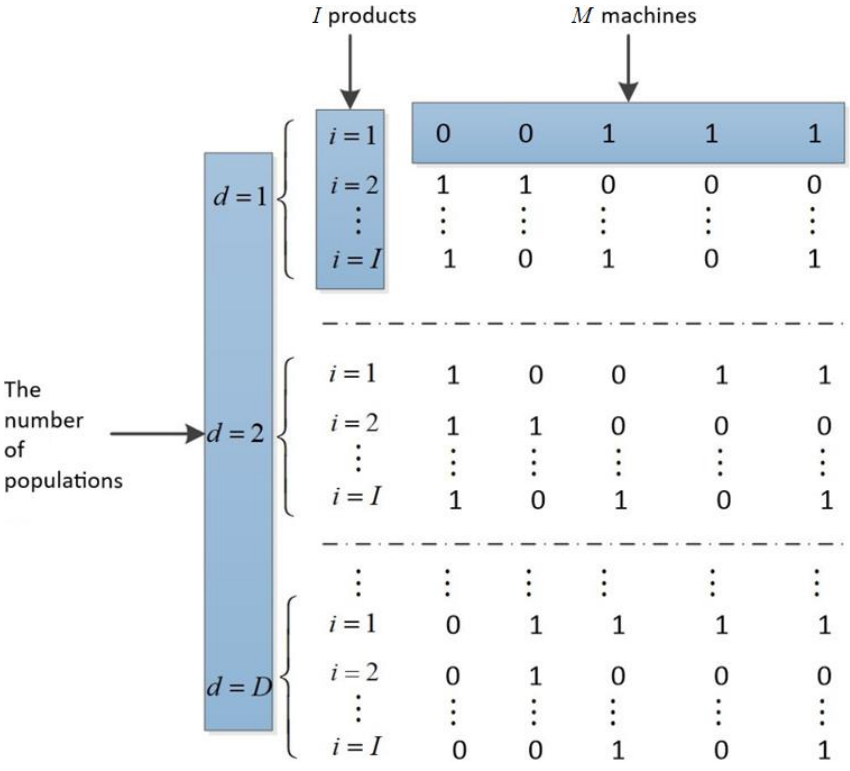

Figure 2: Structure of solution for the MBPSO.

As stated above, each submatrix of the matrix in Fig. 2 represents one of the solutions of the PSO algorithm, denoted as d ($d=1,2,\ldots,D$), with D being the total number of populations. In each submatrix, there are I rows and M columns. Each row stands for a type of products while each column means a machine. Each element in the matrix is valued 1 or 0. If the value is 1, the product can be processed by the machine in its column; otherwise, the value is 0.

The structure of the solution reveals that the designed algorithm is a pure heuristic one. A pure heuristic algorithm can approximate the global optimal solution much faster and more accurate than the traditional planning algorithm. It is suitable for the large-scale complex model established in our research.

(2) Neighbourhood structure of the population

The Gbest model is used in the standard PSO algorithm. By this model, each iterative solution of the population converges to the current optimal solution Gbest. This strategy is a mixed blessing. On the one hand, the algorithm can converge very quickly; on the other hand, the algorithm is prone to the local optimum trap and may converge prematurely. With the same velocity update formula, both the PSO and the BPSO face the said problems. In this paper, the neighbourhood structure of the standard BPSO population is improved, such that the current optimal solution of BPSO is determined according to not only Gbest but also the best-known solution Pbest of some individuals selected at a certain probability from the population. This is to properly degrade the algorithm and increase the diversity of the population. To prevent the local optimum trap, all populations were randomly initialized regularly every fixed number of sampling periods.

(3) Steps of the MBPSO algorithm

Step 1: Population initialization: Randomly generate the initial position and initial velocity of the population and calculate its fitness.

Step 2: Shutdown judgment: Whenever the program loops to shutdown judgment, judge if the termination condition is satisfied; terminate the program and obtain the result if it is satisfied, or continue with the loop.

Step 3: Velocity and position update: Update the velocity and position of the population by Eqs. (7), (9) and (10) and calculate its fitness.

Step 4: Random initialization: Randomly initialize the population every fixed number of sampling periods, and calculate the fitness and the current optimal solution Gbest1; replace the previously stored optimal solution Gbest with Gbest1 if the latter is better.

Step 5: Population degradation: Degrade the population very fixed number of sampling periods, and randomly replace the global optimal solution Gbest with the local optimal solution Pbest of some individuals by roulette method; meanwhile, save the current global optimal solution Gbest; restore the current optimal solution as Gbest if no better solution is available at the end of the sampling period.

Step 6: Fitness evaluation: Evaluate the fitness value and go to Step 2.

4. PERFORMANCE ANALYSIS

The MBPSO algorithm has three adjustable parameters, namely, the inertia weight of velocity w, the self-cognition coefficient c_1 and the social learning coefficient c_2. For our model, the efficiency and accuracy of the algorithm is affected by the parameter settings. Therefore, the sensitivity of the parameters should be analysed, so that reasonable model parameters can be set to improve the performance of the algorithm. Considering the three parameters c_1, c_2 and w, three sets of tests were designed, each of which ran 100 times with 20 iterations per time. The purpose is to compare the mean variances under different parameter settings. During the test, the three parameters c_1, c_2 and w all ranged within [0, 2]. During the sensitivity analysis on one parameter, the other two parameters remain the same. The original data of each set of tests were subjected to a third-order polynomial fitting, aiming to reveal the trend of the model's overall fitness with the changes of parameter values. The results are illustrated in Figs. 3 to 5.

Fig. 3 displays the result of sensitivity analysis on self-cognition coefficient c_1 ($c_1 \in [0, 2]$; $c_2 = 2$; $w = 0.9$). It can be seen that the overall fitness was relatively low when c_1 was around

1.5. Fig. 4 shows the result of sensitivity analysis on social learning coefficient c_2 ($c_2 \in [0, 2]$; $c_1 = 1.5$; $w = 0.9$). As shown in the figure, the overall fitness was relatively low when c_2 was about 2. Fig. 5 presents the result of sensitivity analysis on the inertia weight of velocity w ($w \in [0, 2]$; $c_1 = 1.5$; $c_2 = 2$). It is clear that the overall fitness was relatively low when w was around 1.3. As a random algorithm, the MBPSO has different results in each run. Hence, the "relatively low" refers to the low probability reflected by the fitted curve. It is still possible to achieve better result under other parameter values in a single run.

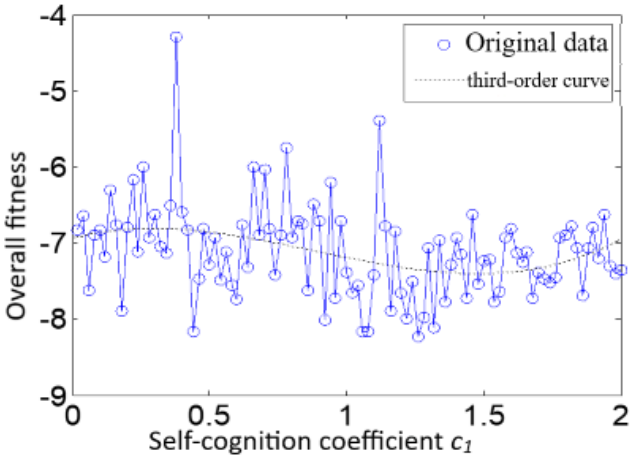

Figure 3: Sensitivity analysis of self-cognition coefficient c_1.

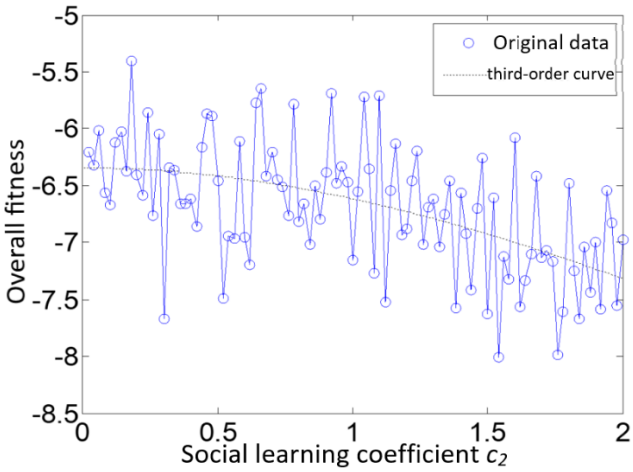

Figure 4: Sensitivity analysis of social learning coefficient c_2.

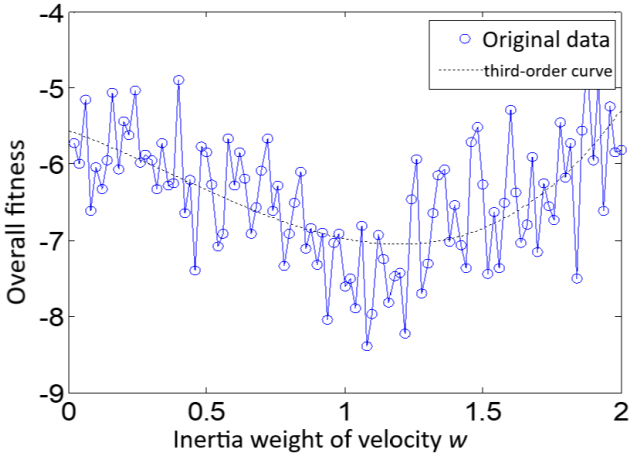

Figure 5: Sensitivity analysis of inertia weight of velocity w.

5. CONCLUSIONS

This paper first introduces the background, purpose, significance, methodology and structure of our research, and then proposes the MBPSO and proper parameter settings through modelling, algorithm design, and algorithm test. In terms of modelling, the multi-objective planning method of operational research was employed to establish a bi-objective programming model for multi-technique, multi-process FJSP. In terms of algorithm solution, the structure of solution was designed for the established model, and the model was subjected to numerical simulation by the MBPSO. In terms of algorithm test, the author conducted parameter sensitivity analysis of the model using the MBPSO and a classic example. Through the above procedure, the parameter settings suitable for our model were obtained as follows: the self-cognition coefficient $c_1 = 1.5$, the social learning coefficient $c_2 = 2$, and the inertia weight of velocity $w = 1.3$. The research findings lay a theoretical basis for the solution and application of real-world multi-technique, multi-process FJSP and provide practical guidance for the flow shop enterprises to improve their scheduling plans.

ACKNOWLEDGEMENT

This paper is supported by National Natural Science Foundation of China (71602049, 51567018), Colleges and Universities' Key Scientific Research Projects of Henan Province (16A630022) and Humanities-Society Scientific Research Projects of Henan Education Department (2016-gh-262).

REFERENCES

[1] Nagata, Y. (2006). Niching method for combinatorial optimization problems and application to JSP, *IEEE Congress on Evolutionary Computation*, 2822-2829, doi:10.1109/CEC.2006.1688663

[2] Zhang, X.; Deng, Y.; Chan, F. T. S.; Xu, P.; Mahadevan, S.; Hu, Y. (2013). IFSJSP: A novel methodology for the Job-Shop Scheduling Problem based on intuitionistic fuzzy sets, *International Journal of Production Research*, Vol. 51, No. 17, 5100-5119, doi:10.1080/00207543.2013.793425

[3] Fan, K.; Zhang, R.-Q.; Xia, G. (2007). An improved Particle Swarm Optimization algorithm and its application to a class of JSP problem, *IEEE International Conference on Grey Systems and Intelligent Services*, 1628-1633, doi:10.1109/GSIS.2007.4443547

[4] Lan, M.; Xu, T.-R.; Huang, F. (2011). Solving flexible multi-objective JSP problem using mixed local search algorithm, *Computer Engineering and Design*, Vol. 32, No. 1, 293-296

[5] Zhao, L.-H.; Deng, F.-Q. (2010). New neighborhood searching methods for FJSP and corresponding algorithm, *Systems Engineering and Electronics*, Vol. 32, No. 8, 1662-1666, doi:10.3969/j.issn.1001-506X.2010.08.23

[6] Zhao, S.-K.; Fang, S.-L.; Gu, X.-J. (2014). Machine selection and FJSP solution based on limit scheduling completion time minimization, *Computer Integrated Manufacturing Systems*, Vol. 20, No. 4, 854-865, doi:10.13196/j.cims.2014.04.zhaoshikui.0854.12.20140416

[7] Tang, M.; Gong, D.; Liu, S.; Lu, X. (2017). Finding key factors for electric vehicle charging station location: a simulation and ANOVA approach, *International Journal of Simulation Modelling*, Vol. 16, No. 3, 541-554, doi:10.2507/IJSIMM16(3)CO15

[8] Zeng, Q.; Yang, Y.; Shen, L.; Zhang, J. (2011). Multiobjective optimization for batch production FJSP based on just in time delivery, *Computer Integrated Manufacturing Systems*, Vol. 17, No. 8, 1780-1789

[9] Zeng, Q.; Shen, L.; Pan, Q., Wu, L. (2014). Multi-objective elaborate scheduling method for batch production FJSP, *Computer Engineering and Applications*, Vol. 50, No. 2, 263-270

[10] Fu, W.-P.; Liu, D.-M.; Lai, C.-W.; Wang, W. (2011). Polychromatic-sets-based improved genetic algorithm for solving multi-species FJSP, *Computer Integrated Manufacturing Systems*, Vol. 17, No. 5, 1004-1010

[11] Grobler, J.; Engelbrecht, A. P.; Kok, S.; Yadavalli, S. (2010). Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time, *Annals of Operations Research*, Vol. 180, No. 1, 165-196, doi:10.1007/s10479-008-0501-4

[12] Liu, B.; Wang, L.; Jin, Y.-H. (2007). An effective PSO-based memetic algorithm for flow shop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 37, No. 1, 18-27, doi:10.1109/TSMCB.2006.883272

[13] Zupan, H.; Herakovic, N.; Zerovnik, J.; Berlec, T. (2017). Layout optimization of a production cell, *International Journal of Simulation Modelling*, Vol. 16, No. 4, 603-616, doi:10.2507/ijsimm16(4)4.396

[14] Halber, A.; Chakravarty, D. (2018). Wireless relay placement optimization in underground room and pillar mines, *Mathematical Modelling of Engineering Problems*, Vol. 5, No. 2, 67-75, doi:10.18280/mmep.050203

[15] Chen, X.; Qiu, J.; Liu, G. (2009). Optimal test selection based on hybrid BPSO and GA, *Chinese Journal of Scientific Instrument*, Vol. 30, No, 8, 1674-1680

[16] Chuang, L.-Y.; Yang, C.-H.; Li, J.-C.; Yang, C.-H. (2012). A hybrid BPSO-CGA approach for gene selection and classification of microarray data, *Journal of Computational Biology*, Vol. 19, No. 1, 68-82, doi:10.1089/cmb.2010.0064

[17] Dai, Y.; Wu, W.; Zhou, H. B.; Zhang, J; Ma, F. Y. (2018).Numerical simulation and optimization of oil jet lubrication for rotorcraft meshing gears, *International Journal of Simulation Modelling*, Vol. 17, No. 2, 318-326, doi:10.2507/IJSIMM17(2)CO6