Int j simul model 19 (2020) 2, 323-333
ISSN 1726-4529 Original scientific paper

CONSTRUCTION AND SIMULATION OF MULTI-OBJECTIVE RESCHEDULING MODEL BASED ON PSO

Li, J. X.[#] & Wen, X. N.
School of Economics and Management, Xidian University, Xi'an 710126, China
E-Mail: lijunxia@stu.xidian.edu.cn ([#] Corresponding author)

Abstract
Job-shop scheduling is critical to the normal operation of the production process. However, there is not yet a robust rescheduling strategy for dynamic job-shop scheduling problems (DJSPs), which is disturbed by multiple random dynamic events. To make up for the gap, this paper classifies dynamic events by scheduling strategies, and details the hypotheses and constraints of dynamic job-shop scheduling. Then, a multi-objective rescheduling model was established to minimize the maximum completion time and maximum machine load of DJSPs. The model was solved by the particle swarm optimization (PSO). Finally, our model was proved effective and robust through MATLAB simulations. The research results provide a reference for the application of swarm intelligence in the field of the JSP.
(Received in October 2019, accepted in February 2020. This paper was with the authors 3 months for 1 revision.)

Key Words: Job-Shop Scheduling Problem (JSP), Particle Swarm Optimization (PSO), Dynamic Events, Multi-Objective Rescheduling

1. INTRODUCTION

In the manufacturing industry, job-shop scheduling, an important aspect of the production plan, guarantees the normal operation of the production process. However, not all scheduling plans can satisfy the production demand. Therefore, the optimization of the scheduling plan becomes the key issue of the job-shop scheduling problem (JSP).

To make matters worse, the production environment is disturbed by various factors, such as emergency insertion of orders, early delivery of products, and machine failure. The original scheduling plan cannot automatically adapt to the disturbed environment. This calls for dynamic scheduling that handles dynamic events in time and reduces their impacts on the production process.

The JSP is widely regarded as non-deterministic polynomial-time (NP) hard. Currently, NP-hard problems are mainly resolved through exact solution and approximate solution [1]. Exact solution methods, namely, branch and bound (BB) method [2] and mathematical programming [3], only apply to small-scale scheduling problems, because of their high time consumption. Approximate solution methods, namely, artificial intelligence (AI) and neighbourhood search [4], can effectively solve large-scale JSPs in a fast and convenient manner.

The JSPs usually deal with the scheduling of multiple jobs on the same machine. Considering machine assignment and operation sequence, the JSP can be extended into the flexible JSP (FJSP), in which the same process consumes different time on different machines, and the same machine takes different time to process different operations.

Taking uncertain factors into account, the dynamic JSP (DJSP), a special form of the FJSP, is more in line with the actual production environment. The uncertain factors may occur at any time during the production process, exerting a huge impact on the processing of jobs.

This paper aims to develop a robust and feasible rescheduling model for DJSPs with multiple objectives. For this purpose, the dynamic events that may occur in job-shops were classified into three categories. Based on each type of events, the hypotheses, constraints, and

objectives of dynamic job-shop scheduling were discussed in details. Next, our rescheduling model was established with the goal to minimize the maximum completion time and maximum machine load. The established model was solved by the particle swarm optimization (PSO) and verified through MATLAB simulation.

2. LITERATURE REVIEW

2.1 The PSO

The common approaches to solve the JSP include swarm intelligence algorithms, neutral networks (NNs), genetic algorithm (GA), simulated annealing (SA) algorithm, and tabu search (TS). Among them, the PSO [5], a typical swarm intelligence algorithm, has been widely applied in the JSPs, for its fast convergence, good robustness, and ease of implementation. But the traditional PSO is not suitable for all kinds of JSPs. Thus, many attempts have been made to optimize this algorithm.

Yeh et al. [6] encoded particles in the PSO with binary coding, creating a continuous PSO for discrete problems. Li et al. [7] obtained an integer by iteratively adjusting the position vector of particles, which facilitates the solving of the FJSP. Bu [8] added chaotic disturbance to individual optimal particle to prevent its position from converging, so that the particle can search around the neighbourhood of the global optimal solution. Wong and Ngan [9] effectively solved the FJSP through combining the PSO and the GA.

To prevent the PSO from the local optimum trap, Ohmi and Panday [10] updated particles by the crossover and mutation of the GA, and searched for the optimal solution by the SA algorithm. Mekhmoukh and Mokrani [11] introduced spatial neighbourhood to the PSO, converted particles in the same spatial neighbourhood into a sub-swarm, and changed the threshold dynamically as per the evolution results, thereby ensuring the diversity of the swarm. To increase population diversity, Marinakis and Marinaki [12] adjusted the dynamic selection of neighbourhood based on neighbourhood topology, and bolstered the information exchange between neighbourhoods through social belief.

2.2 Multi-objective DJSP

For a multi-objective DJSP, it is necessary to optimize every object at the same time. Cruz-Chávez et al. [13] improved the SA algorithm, and coupled the improved algorithm with the idea of the TS, aiming to minimize the delivery time of the FJSP. Zhang et al. [14] combined global search, local search, and random generation into an initialization method to generate a high-quality initial population for the FJSP with three objectives, namely, minimum maximum machine load, machine load balance, and minimum completion time. Qi et al. [15] improved the PSO based on the SA algorithm and Baldwinian learning strategy, and applied the improved algorithm to solve the FJSP with the aims to minimize the completion time, maximum machine load, and maximum load of a single machine.

To realize sustainable, intelligent, and green manufacturing, Zhang et al. [16] proposed a new rescheduling method based on the PSO, which minimizes the completion time and global consumption for machine failure. Agrawal et al. [17] evaluated the risk of rescheduling through analytic approach, and minimized the rescheduling risk by the GA, ensuring the stability of the maximum completion time. Wu et al. [18] developed a detailed mathematical model and processing steps for new added jobs, and solved the multi-objective DJSP with an improved GA. Targeting the JSP constrained by delivery bottleneck, Zuo et al. [19] constructed a scheduling model to deliver products satisfactorily under the constraint, and solved the model with an improved PSO.

2.3 Dynamic events

Some scholars have tried to optimize the scheduling plan faced with several dynamic events. For example, Wen et al. [20] improved the variable neighbourhood search algorithm to optimize the scheduling plan under the insertion of random jobs and machine failure. Yolmeh and Kianfar [21] designed an efficient hybrid GA to solve three dynamic events, including insertion of random jobs, machine failure, and the change of processing time, and proved the advantages of the algorithm in solution quality and runtime.

3. MULTI-OBJECTIVE DYNAMIC JOB-SHOP SCHEDULING MODEL

3.1 Classification of dynamic events

Dynamic events can be classified according to the key information in the JSP, such as jobs, machines, and operations. Here, dynamic events are categorized by scheduling strategies, because this research aims to find the suitable strategies to schedule various dynamic events.

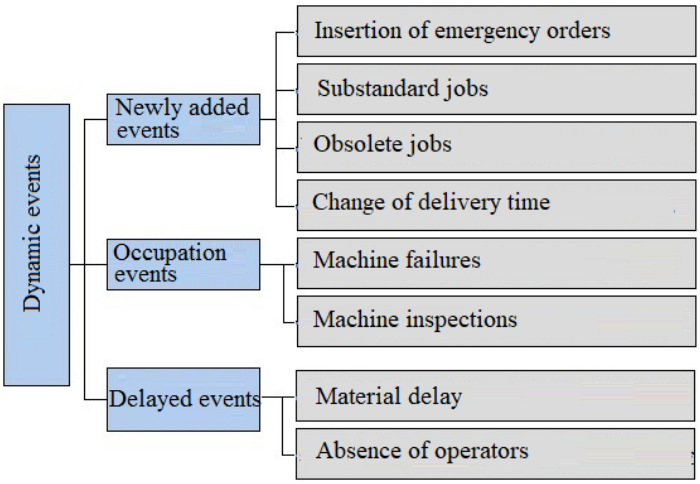

Figure 1: Classification of dynamic events.

As shown in Fig. 1, new added events add jobs to the processing sequence, including emergency orders, substandard jobs, and obsolete jobs. Each emergency order may include one or more jobs. In occupation events, the machine is occupied for some reason and cannot process jobs. A failed machine, whether repaired or not, is regarded as occupied. Delayed events mainly refer to material delays. A delayed event can be understood as a "time" occupation event, i.e. the delayed time is occupied and cannot be used to process jobs.

3.2 Dynamic job-shop scheduling model

Once dynamic events occur, three scheduling strategies are available: event-driven rescheduling, periodic rescheduling, and hybrid rescheduling. In event-driven rescheduling, the system makes immediate response and rescheduling upon detecting dynamic events; in periodic rescheduling, the system performs rescheduling at regular intervals, whether dynamic events occur or not; in hybrid rescheduling, the system performs rescheduling at regular intervals, while continuously detecting dynamic events and solving the detected events right away. Therefore, hybrid rescheduling is a combination of the former two scheduling strategies. This paper adopts event-based rescheduling to solve the DJSP with dynamic events, and adjusts the rescheduling plan according to the specific event.

Rescheduling strategies can be divided into postponement rescheduling, insertion rescheduling, and complete rescheduling. The postponement rescheduling deals with special

problems like newly added orders. The operations in the new order are arranged directly after the current operation sequence [22], without affecting the sequence of other operations. However, insertion rescheduling and complete rescheduling both will affect the results of the original scheduling plan.

Insertion rescheduling is to insert unprocessed operations into the scheduled results without affecting other operations. The unprocessed operations must to meet the basic constraints of the JSP. In addition, the idle time of the selected machine must be longer than or equal to the processing time of the inserted operations. Thus, insertion rescheduling only applies to a few cases, although it does not affect the other operations. This strategy is more suitable for adding new simple operations that can be processed in a short time.

Complete rescheduling generates completely new results by rescheduling all unprocessed operations, including the operations of new tasks and unprocessed operations of the original tasks. The operation sequence and completion time of the original schedule will be completely changed. This strategy only needs to satisfy the basic constraints of the JSP.

Since the DJSP is a special case of FJSP, the first step of modelling dynamic job-shop scheduling is to develop a model of flexible scheduling, and then adjust some of the constraints. Before transforming the FJSP into a mathematical problem, the following hypotheses were presented to approximate the actual production environment, and simplify the modelling process:

H1: The operation sequence of any job must meet the operation constraints;

H2: The sequence of different jobs is not considered;

H3: All machines are available at the beginning;

H4: The processing time of each job is fixed on any machine;

H5: Once an operation is processed, the processing cannot be interrupted unless the machine becomes unavailable;

H6: The loading/unloading time and transport time of any job are neglected.

The DJSP in our research faces the following constraints:

(1) Machine constraints

The set of machines available for each operation is predetermined, and a subset of the total set of machines:

$$Y_{ij} \subseteq \{m_1, m_2, \dots, m_M\} \tag{1}$$

where, Y_{ij} is the set of machines available for operation O_{ij}.

The processing time of each operation on any machine should not be negative nor infinite:

$$0 < p_{ijm} < \infty, \qquad i = 1, 2, \dots, N, \qquad j = 1, 2, \dots, n, \qquad m = 1, 2, \dots, M \tag{2}$$

where, p_{ijm} is the processing time of the j^{th} operation of the i^{th} job on machine m.

The same operation can only be processed on one machine at a time. In other words, an operation is unprocessed, processed or being processed. The unprocessed state and being processed state can be respectively expressed as:

$$\sum_{m=1}^{M} X_{ijm} = 0, \qquad i = 1, 2, \dots, N, \qquad j = 1, 2, \dots, n \tag{3}$$

$$\sum_{m=1}^{M} X_{ijm} = 1, \qquad i = 1, 2, \dots, N, \qquad j = 1, 2, \dots, n \tag{4}$$

where, X_{ijm} is an indicator of whether operation O_{ij} is processed on machine m.

Each machine can only process one job at a time:

$$\sum_{i=1}^{N}\sum_{j=1}^{n} X_{ijm} = 1, \qquad m = 1, 2, \dots, N \tag{5}$$

(2) Time constraints

The first operation of any job should be started at or after time zero:

$$s_{i1} \geq 0, \qquad i = 1, 2, \dots, N \tag{6}$$

where, s_{i1} is the start time of the first operation of the i^{th} job.

The start time of operation O_{ij} on machine m should be right after or later than its previous operation $O_{i(j-1)}$, and right after or later than the completion time of the previous operation O_{ab} of the machine m. Hence, the start time of the j^{th} operation of the i^{th} job on machine m must satisfy:

$$s_{ijm} = \max\{c_{i(j-1)k}, c_{abm}\} \tag{7}$$

where, $c_{i(j-1)k}$ is the completion time of the $(j-1)^{\text{th}}$ operation of the i^{th} job on machine k:

$$c_{i(j-1)k} = s_{i(j-1)k} + p_{i(j-1)k} \tag{8}$$

where, $p_{i(j-1)k}$ is the processing time of the $(j-1)^{\text{th}}$ operation of the i^{th} job on machine k. Then, the completion time of the previous operation O_{ab} of the machine m can be expressed as:

$$c_{abm} = s_{abm} + p_{abm} \tag{9}$$

Dynamic events occur randomly in time. At the time of occurrence, static scheduling has been completed and processing has begun:

$$0 < dyt < f_1 \tag{10}$$

where, f_1 is the performance index of maximum completion time.

In this paper, the minimization of the maximum completion time and the maximum machine load are selected as the optimization objectives. Then, the objective model of static scheduling was constructed. After that, the objectives of dynamic scheduling were adjusted according to the occurrence of each kind of dynamic events.

Completion time refers to the duration from time zero of processing to the completion of all operations in all jobs. Let p_{ijm} be the processing time of operation O_{ij} on machine m, and X_{ijm} be the indicator of whether operation O_{ij} is processed on machine m. For the i^{th} job, its completion time c_i can be obtained by traversing all machines and operations. Our scheduling strategy aims to minimize the maximum completion time of N jobs. This optimization objective and the completion time of each job can be respectively expressed as:

$$f_1 = min\{maxc_i\}, \qquad i = 1, 2, \dots, N \tag{11}$$

$$c_i = \sum_{m=1}^{M}\sum_{j=1}^{n} p_{ijm} \times X_{ijm} \tag{12}$$

The total processing time of a single machine after all operations in all jobs are completed is known as single machine load. For machine m, its machine load, i.e. the total processing time, can be obtained by traversing all operations in all jobs. Hence, the minimization of the maximum machine load can be described as:

$$f_2 = min\left\{max\sum_{m=1}^{M}\sum_{j=1}^{n} p_{ijm} \times X_{ijm}\right\}, \qquad m = 1, 2, \dots, M \tag{13}$$

4. SOLVING ALGORITHM

4.1 Overview of the PSO

The PSO is an important swarm intelligence algorithm inspired by the foraging behaviours of birds [23, 24]. Initially, the position and flight direction of each bird are random, but the birds will continuously search for the food source through information sharing.

When applied to an optimization problem, the PSO initializes a swarm of particles (solutions) within its feasible solution space, and evaluates the quality of each particle based on fitness. Each particle has two attributes: velocity and position. The former determines the direction and distance of particle movement.

During iterative searches, each particle updates its velocity and position based on the best-known individual solution $pbest$ and the best-known global solution $gbest$. In each iteration, the particle recalculates its fitness after updating its velocity and position. Then, the fitness values of the particle were compared to update the $pbest$, and the fitness values of all particles in the swarm were compared to update the $gbest$.

The velocity and position of each particle can be respectively updated by:

$$v_{i+1} = \omega v_t + C_1 r_1 (P_t - x_t) + C_2 r_2 (G_t - x_t) \tag{14}$$

$$x_{t+1} = x_t + v_{t+1} \tag{15}$$

where, ω is the inertia weight; C_1 and C_2 are learning factors; r_1 and r_2 are random numbers in the interval of [0, 1]. The specific meaning of each parameter is explained as follows:

Inertia weight ω controls the influence of the velocity at the previous moment on the velocity at the current moment, i.e. the ability of particle to expand the search space. The value of ω is positively correlated with the global search ability, and negatively with the local search ability.

Learning factors C_1 and C_2 adjust the weights of self-learning and group learning of particles. The expansion of search space is controlled by the two factors. The value of each learning factor is positively correlated with the difference between the velocity at the previous moment on the velocity at the current moment. If $C_1 = C_2 = 0$, then the particle will move at the initial velocity and direction until reaching the boundary of the feasible solution space, which reduces the probability of finding the optimal solution. The values of both learning factors are usually set to 2.

Random numbers r_1 and r_2 add random disturbances to particles. The randomized particle movements help to expand the search space.

The standard PSO is implemented in the following steps:

Step 1: Input the relevant parameters;

Step 2: Randomly initialize the swarm;

Step 3: Calculate the fitness of each particle;

Step 4: Compare the fitness values of all particles to obtain the best-known global solution;

Step 5: Judge if the maximum number of iterations is reached; if yes, terminate the optimization and output the result; otherwise, go to Step 6;

Step 6: Update the velocity and position of each particle;

Step 7: Calculate the fitness and update the best-known solution of each particle;

Step 8: Update the best-known global solution and go to Step 5.

4.2 Solving multi-objective dynamic job-shop scheduling with PSO

To solve our model with the PSO, the first step is to encode the feasible solutions. Here, every feasible solution is encoded by a common two-layer coding strategy based on both operation and machine. The coding of a feasible solution is illustrated in Fig. 2 below.

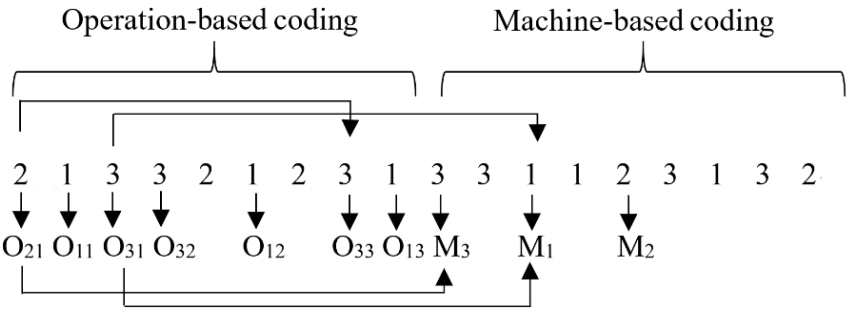

Figure 2: The structure of two-layer coding.

The operation-based coding encodes each operation of a job with the serial number of the job, and the serial number of the operation in the operation sequence of the job. In Fig. 2, the first digit 2 indicates that operation O_{21} is the first operation of the 2nd job; the second, sixth and ninth digits indicate that the corresponding operations belong to the 1st job.

The machine-based coding adds the serial number of each machine to the operation-based code of the corresponding operation. The length of machine-based code is the same as the operation-based code. In Fig. 2, the first bit 3 indicates that operation O_{21} is processed on the machine 3; the third bit 1 indicates that operation O_{31} is processed on machine 1.

During decoding, the encoding procedure was reversed. When the machine-based code was decoded, the processing would be infeasible if the processing time of the current operation on the machine was negative, zero or greater than 1,000. In this case, the code was modified by randomly selecting another machine from the set of available machines.

The PSO parameters were defined and configured as follows:

Swarm size $Size$:

This parameter directly bears on the search ability and computing load. For general problems, the swarm size should be set within [20, 40] to obtain a good solution. For difficult problems, the swarm size should be properly expanded.

Maximum number of iterations T_{max}:

This parameter should be adjusted in the light of the problem scale. If T_{max} is too small, the PSO will terminate without finding the optimal solution; if T_{max} is too large, the extremums will remain unchanged after the algorithm converges to the optimal solution, causing a great waste of resources.

Maximum velocity V_{max}:

This parameter represents the maximum distance that each particle moves in the iterative process, and controls the search and expansion abilities of the algorithm. If V_{max} is too large, particles may fly over the optimal solution or even beyond the feasible search space; if V_{max} is too small, the particles will stuck in the local area, so that the algorithm falls into the local optimum trap. In general, the maximum velocity is controlled between -2 and 2.

Inertia weight ω:

This parameter reflects how much the current velocity of each particle is affected by the velocity at the previous moment. The inertia weight is often set to decrease linearly from 0.8 to 0.4.

Learning factors C_1 and C_2:

As mention before, the two parameters are generally set to 2.

The velocity and position of each particle was updated iteratively by Eqs. (14) and (15), respectively. In the velocity update formula, the first part ωv_t is the self-cognition part, meaning that the particle is affected by the current velocity; the second part $C_1 r_1 (P_t - x_t)$ is the self-learning part, meaning that the particle will be affected by its optimal position; the third part $C_2 r_2 (G_t - x_t)$ is the group learning part, meaning that the particles will be affected by the swarm. In each part, the parameters reflect the degree of influence by that part. For

example, if the self-learning factor $C_1 = 0$, the particles all fly to the global optimal position, making it easy to fall into local optimum. If the group-learning factor $C_2 = 0$, the lack of information sharing will slow down the convergence.

The PSO can be terminated under two conditions: the pre-set fitness is realized, or the maximum number of iterations is reached. In JSPs, the position and value of the optimal solution are unknown. Therefore, the termination condition of maximum number of iterations was adopted.

5. SIMULATION VERIFICATION

To verify its feasibility, our rescheduling strategy was tested through MATLAB simulation of JSPs in different scales, using benchmark examples of a 3×3 complete FJSP and a 5×6 partial FJSP.

The selected datasets were verified separately for each kind of dynamic events. The FJSPs with at most two machines available for each operation were found not suitable for machine failure simulation.

Concerning period of dynamic events, the time close to the initial time of processing is almost similar to the occurrence of the dynamic event, and the time close to the end of processing is rescheduled. Therefore, there is no room for rescheduling after the occurrence of dynamic events. As a result, 25 %-75 % of the total processing time of each FJSP was selected as the period for dynamic event occurrence.

In addition, the occurrence time of dynamic events was divided into two periods for simulation. The impacts of rescheduling results on the original scheduling were compared to reveal the robustness of the rescheduling strategy when dynamic events occur in different periods.

The swarm size and the maximum number of iterations were set according to the problem scale (Table I).

Table I: Parameter settings.

Problem	$\boldsymbol{Size}$	$\boldsymbol{T_{max}}$
Complete FJSP	4	10
Partial FJSP	20	20

The 3×3 complete FJSP was simulated first, and the results are shown in Table II.

Table II: Simulation results of the complete FJSP.

Jobs	**Operations**	**Optional machines and processing time**		
		$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$
$\boldsymbol{J_1}$	O_{11}	4	2	3
	O_{12}	3	2	5
	O_{13}	2	4	6
$\boldsymbol{J_2}$	O_{21}	3	5	2
	O_{22}	5	3	4
	O_{23}	2	3	5
$\boldsymbol{J_3}$	O_{31}	6	4	5
	O_{32}	3	4	5
	O_{33}	4	3	5

Table III: Processing schedule of the 4th job on each machine.

Job	Operation	Optional machine and processing time		
		M_1	M_2	M_3
J_4	O_{41}	5	4	5
	O_{42}	3	4	6
	O_{43}	4	4	6

Table IV: Simulation results of the partial FJSP.

Jobs	Operations	Optional machines and processing time					
		M_1	M_2	M_3	M_4	M_5	M_6
J_1	O_{11}	3	2	5	—	3	—
	O_{12}	—	2	3	2	—	—
	O_{13}	2	3	—	—	5	—
	O_{14}	3	—	—	4	6	—
	O_{15}	—	5	—	7	5	8
J_2	O_{21}	3	—	4	—	3	—
	O_{22}	5	3	—	5	4	—
	O_{23}	—	2	4	—	—	11
	O_{24}	—	6	—	8	—	4
	O_{25}	4	6	—	5	4	—
J_3	O_{31}	5	7	—	5	—	—
	O_{32}	—	3	—	4	6	—
	O_{33}	—	—	12	—	8	11
	O_{34}	7	5	—	6	8	—
	O_{35}	7	9	—	6	7	6
J_4	O_{41}	9	—	—	8	—	8
	O_{42}	—	5	—	7	—	6
	O_{43}	3	—	5	7	—	6
	O_{44}	6	—	9	7	5	4
	O_{45}	—	9	7	—	7	—
J_5	O_{51}	4	5	8	—	6	8
	O_{52}	5	4	—	6	—	8
	O_{53}	5	—	7	—	—	6
	O_{54}	—	6	8	—	7	—
	O_{55}	7	—	9	—	8	6

For the 3×3 complete FJSP, the swarm size and the maximum number of iterations is 4 and 10, respectively. In the original scheduling plan, the maximum completion time and the maximum machine load are both 13. When an emergency order was received at time 5, the 4th job should be added to the processing sequence (Table III). If the new job was directly scheduled, then the maximum completion time and maximum machine load would be 27 and 21, respectively; under the same conditions, the unprocessed operations and the new job were rescheduled, reducing the maximum completion time and maximum machine load to 18 and 17, respectively. Hence, the rescheduling strategy shortens the maximum completion time, and balances the machine load.

In the event of machine failure, the partial FJSP of 5×6 was simulated. The simulation results are recorded in Table IV above.

In the original scheduling plan, the maximum completion time and the maximum machine load are 43 and 39, respectively. When machine 2 failed at time 22, the operation being processed and its subsequent operations on that machine could not be processed any more, and the current operation was scrapped, calling for reprocessing. The current operation on machine was the 4^{th} operation of the 2^{nd} job. After rescheduling, the maximum completion time was 44 and the maximum machine load was 36. The rescheduling results were not much different from the original results.

6. CONCLUSIONS

Based on scheduling strategies, this paper divides dynamic events into three classes (i.e. newly added events, occupation events and delayed events). On this basis, the hypotheses, constraints, and objective functions of dynamic job-shop scheduling were introduced in details. Then, the authors established a dynamic job-shop scheduling model, according to the impacts of each type of dynamic events. After that, the PSO was adopted to solve the multi-objective DJSP, and the coding and decoding processes were explained clearly. Finally, the feasibility and robustness of our rescheduling strategy were verified through MATLAB simulation. The research results shed new light on the application of swarm intelligence in the field of the JSP.

ACKNOWLEDGEMENT

Shaanxi province soft science research program key project **(**2016KRZ010**)**; sponsored by the Seed Foundation of Innovation Practice for Graduate Students in Xi'dian University.

REFERENCES

[1] Deb, S.; Fong, S.; Tian, Z.; Wong, R. K.; Mohammed, S.; Fiaidhi, J. (2016). Finding approximate solutions of NP-hard optimization and TSP problems using elephant search algorithm, *Journal of Supercomputing*, Vol. 72, No. 10, 3960-3992, doi:10.1007/s11227-016-1739-2

[2] Lawler, E. L.; Wood, D. E. (1966). Branch-and-bound methods: a survey, *Operations Research*, Vol. 14, No. 4, 699-719

[3] Mavrotas, G. (2009). Effective implementation of the ε-constraint method in Multi-Objective Mathematical Programming problems, *Applied Mathematics and Computation*, Vol. 213, No. 2, 455-465, doi:10.1016/j.amc.2009.03.037

[4] Wu, D. (2019). Multi-objective decision-making of new retailing terminals based on particle swarm optimization and genetic algorithm, *Journal Européen des Systèmes Automatisés*, Vol. 52, No. 6, 607-615, doi:10.18280/jesa.520608

[5] Liao, J.; Lin, C. (2019). Optimization and simulation of job-shop supply chain scheduling in manufacturing enterprises based on particle swarm optimization, *International Journal of Simulation Modelling*, Vol. 18, No. 1, 187-196, doi:10.2507/IJSIMM18(1)CO5

[6] Yeh, W.-C.; Lin, Y.-C.; Chung, Y. Y.; Chih, M.-C. (2010). A particle swarm optimization approach based on Monte Carlo simulation for solving the complex network reliability problem, *IEEE Transactions on Reliability*, Vol. 59, No. 1, 212-221, doi:10.1109/TR.2009.2035796

[7] Li, X.; McLerran, L.; Voloshin, M.; Wang, R.-T. (1991). Corrections to high-energy particles interacting through an instanton as quantum fluctuations in the position of the instanton, *Physical Review D*, Vol. 44, No. 9, 2899-2915, doi:10.1103/PhysRevD.44.2899

[8] Bu, S. (2006). Quantum trajectory of an individual particle in the presence of chaos, *Physica D: Nonlinear Phenomena*, Vol. 217, No. 2, 103-106, doi:10.1016/j.physd.2006.03.016

[9] Wong, T. C.; Ngan, S. C. (2013). A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop, *Applied Soft Computing*, Vol. 13, No. 3, 1391-1399, doi:10.1016/j.asoc.2012.04.007
[10] Ohmi, K.; Panday, S. P. (2009). Particle tracking velocimetry using the genetic algorithm, *Journal of Visualization*, Vol. 12, No. 3, 217-232, doi:10.1007/BF03181860
[11] Mekhmoukh, A.; Mokrani, K. (2015). Improved fuzzy C-means based particle swarm optimization (PSO) initialization and outlier rejection with level set methods for MR brain image segmentation, *Computer Methods and Programs in Biomedicine*, Vol. 122, No. 2, 266-281, doi:10.1016/j.cmpb.2015.08.001
[12] Marinakis, Y.; Marinaki, M. (2013). Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem, *Soft Computing*, Vol. 17, No. 7, 1159-1173, doi:10.1007/s00500-013-0992-z
[13] Cruz-Chávez, M. A.; Martínez-Rangel, M. G.; Cruz-Rosales, M. H. (2015). Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem, *International Transactions in Operational Research*, Vol. 24, No. 5, 1119-1137, doi:10.1111/itor.12195
[14] Zhang, G.-H.; Shao, X.; Li, P.-G.; Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering*, Vol. 56, No. 4, 1309-1318, doi:10.1016/j.cie.2008.07.021
[15] Qi, Y.; Liu, F.; Liu, M.; Gong, M.-G.; Jiao, L.-C. (2012). Multi-objective immune algorithm with Baldwinian learning, *Applied Soft Computing*, Vol. 12, No. 8, 2654-2674, doi:10.1016/j.asoc.2012.04.005
[16] Zhang, L.-P.; Li, X.; Gao, L.; Zhang, G.-H. (2016). Dynamic rescheduling in FMS that is simultaneously considering energy consumption and schedule efficiency, *International Journal of Advanced Manufacturing Technology*, Vol. 87, No. 5-8, 1387-1399, doi:10.1007/s00170-013-4867-3
[17] Agrawal, R.; Pattanaik, L. N.; Kumar, S. (2012). Scheduling of a flexible job-shop using a multi-objective genetic algorithm, *Journal of Advances in Management Research*, Vol. 9, No. 2, 178-188, doi:10.1108/09727981211271922
[18] Wu, C.-C.; Liu, S.-C.; Zhao, C.; Wang, S.-Z.; Lin, W.-C. (2018). A multi-machine order scheduling with learning using the genetic algorithm and particle swarm optimization, *The Computer Journal*, Vol. 61, No. 1, 14-31, doi:10.1093/comjnl/bxx021
[19] Zuo, Y.; Gu, H.-Y.; Xi, Y.-G. (2008). Study on constraint scheduling algorithm for job shop problems with multiple constraint machines, *International Journal of Production Research*, Vol. 46, No. 17, 4785-4801, doi:10.1080/00207540701324143
[20] Wen, Y.; Xu, H.; Yang, J. (2011). A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system, *Information Sciences*, Vol. 181, No. 3, 567-581, doi:10.1016/j.ins.2010.10.001
[21] Yolmeh, A.; Kianfar, F. (2012). An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times, *Computers & Industrial Engineering*, Vol. 62, No. 4, 936-945, doi:10.1016/j.cie.2011.12.017
[22] Song, L. J.; Gu, H. P.; Jin, S. Y.; Zhao, H. (2015). Rescheduling methods for manufacturing firms applying make-to-order strategy, *International Journal of Simulation Modelling*, Vol. 14, No. 4, 719-731, doi:10.2507/IJSIMM14(4)CO18
[23] Huang, C.-J.; Zhou, X.-H.; Hou, D.-S. (2018). Online no-wait scheduling of leather workshop supply chain based on particle swarm optimization, *Journal Européen des Systèmes Automatisés*, Vol. 51, No. 1-3, 153-167, doi:10.3166/JESA.51.153-167
[24] Wang, Y.; Yang, O.; Wang, S. N. (2019). A solution to single-machine inverse job-shop scheduling problem, *International Journal of Simulation Modelling*, Vol. 18, No. 2, 335-343, doi:10.2507/IJSIMM18(2)CO7