Hindawi
Journal of Advanced Transportation
Volume 2017, Article ID 1527858, 12 pages
https://doi.org/10.1155/2017/1527858

Research Article

A Variable Interval Rescheduling Strategy for Dynamic Flexible Job Shop Scheduling Problem by Improved Genetic Algorithm

Lei Wang,[1] Chaomin Luo,[2] and Jingcao Cai[1]

[1] *School of Mechanical and Automotive Engineering, Anhui Polytechnic University, Wuhu 241000, China*
[2] *Department of Electrical and Computer Engineering, University of Detroit Mercy, Detroit, MI 48221, USA*

Correspondence should be addressed to Lei Wang; wangdalei2000@126.com

Received 27 April 2017; Accepted 14 June 2017; Published 19 July 2017

Academic Editor: Rongxin Cui

In real-world manufacturing systems, production scheduling systems are often implemented under random or dynamic events like machine failure, unexpected processing times, stochastic arrival of the urgent orders, cancellation of the orders, and so on. These dynamic events will lead the initial scheduling scheme to be nonoptimal and/or infeasible. Hence, appropriate dynamic rescheduling approaches are needed to overcome the dynamic events. In this paper, we propose a dynamic rescheduling method based on variable interval rescheduling strategy (VIRS) to deal with the dynamic flexible job shop scheduling problem considering machine failure, urgent job arrival, and job damage as disruptions. On the other hand, an improved genetic algorithm (GA) is proposed for minimizing makespan. In our improved GA, a mix of random initialization population by combining initialization machine and initialization operation with random initialization is designed for generating high-quality initial population. In addition, the elitist strategy (ES) and improved population diversity strategy (IPDS) are used to avoid falling into the local optimal solution. Experimental results for static and several dynamic events in the FJSP show that our method is feasible and effective.

1. Introduction

The flexible job shop scheduling problem (FJSP) is an extension of the classical job shop scheduling problem (JSP) [1]. FJSP can be divided into two subproblems [2]: One is to determine operation sequences, and the other one is to select corresponding machine for these operations. The feasible solution is generated by using these two subproblems in order to make one or some objective functions optimal or near-optimal. FJSP is more complex than JSP because not only the sequencing problem for all operations but also an available machine for each operation needs to be done. Therefore, FJSP is considered as an extremely NP-hard problem [3, 4]. Hence, it cannot be solved by some exact approaches [5] because of the increased computational time in an exponential manner. In recent years, numerous heuristics approaches are proposed to solve FJSP, and these methods include simulated annealing (SA) [6], tabu search (TS) [7], ant colony optimization (ACO) [8], particle swarm optimization (PSO) [9], artificial bee colony (ABC) [10], GA [11], evolutionary algorithm (EA) [12], and improved or hybrid algorithms [13, 14].

As we know, in these heuristics approaches, GA is one of the most popular methods for solving the FJSP because of its potential parallelism and robustness. However, GA has a slow convergence rate and tends to fall into the local optimum; therefore many improved or hybrid genetic algorithms are proposed to deal with the above-mentioned disadvantages. Reference [1] proposed a modified genetic algorithm based on a fuzzy roulette wheel selection, new crossover operator, and new mutation operator for flexible job shop scheduling problems. Zhang et al. [2] designed global and local selection methods in order to generate high-quality initial population for minimizing the makespan. A new chromosome representation and different crossover and mutation strategies are used to minimize the makespan in [11]. A hybrid genetic algorithm with tabu search is proposed to solve FJSP with transportation and bounded processing times constraints in [15]. An improved genetic algorithm for solving the distributed and flexible job shop scheduling problem was introduced in [14]. A novel variable neighborhood genetic algorithm for multiobjective flexible job shop scheduling problems was given in [16].

However, in fact, in real-world floor shop, the production environments are uncertain, and several dynamic or random events such as machine failure, unexpected processing times, random arrival of the urgent orders, and cancellation of the orders may often occur [17]. These dynamic events will lead the initial so-called static scheduling scheme to be infeasible; therefore many dynamic flexible job shop scheduling approaches are proposed to satisfy the new environments [18]. The robust scheduling for multiobjective flexible job shop problem with random machine breakdowns was investigated in [19]. A two-stage hybrid genetic algorithm is proposed to solve FJSP with random machine breakdowns by Al-Hinai and Elmekkawy [20]. A genetic algorithm was used to generate a robust and stable scheduling for one machine scheduling with random machine breakdowns in [21]. Zhao et al. proposed a dynamic rescheduling model by using multiagent system in [22]. A heuristic was used to perform the reactive scheduling in a dynamic job shop where jobs arrive over time in [23]. The random job arrivals and machine breakdowns are considered in [24], and a variable neighborhood search is used to solve these two kinds of dynamic events in the FJSP. Considering efficiency and stability, a metaheuristic approach based on GA is developed for dynamic flexible job shop scheduling in [25]. A hybrid genetic algorithm is presented to minimize makespan in dynamic job shop scheduling problem in [26]. Two evolutionary algorithms are used for multiobjective optimization in the FJSP under random machine breakdown in [27]. A hybrid method based on artificial immune systems and priority dispatching rules is proposed by Qiu and Lau [28] for dealing with dynamic online job shop scheduling problem. A hybrid tabu search and genetic algorithm integrated with a simulator to deal with the dynamic job shop scheduling problem with machine breakdowns and random job arrivals in [29]. The gravitational emulation local search algorithm is proposed to solve the multiobjective flexible dynamic job shop scheduling problem in [30]. A multicontextual dispatching rule is proposed by Lu and Romanowski to solve job shops with dynamic job arrival [31].

Although the dynamic or uncertain events in job shop problems or flexible job shop scheduling problems have attracted many researchers in recent years, most of the researches on dynamic scheduling problems focus on certain or primary events such as machine failure or random job's arrival, yet the research on uncertain job's damage is comparatively limited. Meanwhile, as mentioned above, GA has a slow convergence rate and tends to fall into the local optimum; the second obvious weakness of the conventional GA-based approaches is that their performance deteriorates rapidly for complex flexible job shop scheduling problems; the third problem is how to generate high-quality initial population.

Therefore, in this paper, the dynamic FJSP with several uncertain dynamic events is studied. An effective rescheduling strategy is proposed. An improved GA is proposed to minimize the makespan. Our first contribution is that a random initialization population by combining initialization machine and initialization operation with random initialization is designed for generating high-quality initial population. Our second contribution is that the elitist strategy (ES) and improved population diversity strategy (IPDS) are used to avoid falling into the local optimal solution. The third contribution is that the variable interval rescheduling strategy is proposed to reduce the impact of dynamic events on scheduling results. Simulation results demonstrate that the proposed method is effective in both scheduling stage and rescheduling stage.

Table 1: Processing information of FJSP with 2 jobs and 2 machines.

Job	Operation	Processing time		
		$M1$	$M2$	$M3$
$J1$	$O11$	2	3	4
	$O12$	3	4	5
	$O13$	3	—	—
$J2$	$O21$	7	2	3
	$O22$	4	4	—
	$O23$	2	—	5

2. Mathematical Formulation with Dynamic Events

In FJSP, there is a set of n jobs $J = \{1, 2, 3, \ldots, n\}$ to be processed on m machines indexed by $M = \{1, 2, 3, \ldots, m\}$. Each job has one or more operations denoted by o_{ij}, $j \in \{1, 2, \ldots, n_i\}$ (where n_i is the total operations of job J_i). Each operation in any job can be processed by one machine out of a predetermined set of available machines. The processing time of operation o_{ij} on machine k is denoted by t_{ijk}. FJSP is going to allocate each operation to a suitable machine and determine the sequence of allocated operations on each machine [32].

The assumptions on the FJSP are as follows:

(1) At time zero, each machine can be used.

(2) At time zero, each job can be started.

(3) One operation only can be processed on a machine at a time.

(4) Once an operation is processed on a machine, it cannot be interrupted.

(5) Both due dates and release times are not specified.

(6) The transportation and setup times of the jobs are ignored.

The objective is to minimize makespan denoted by C, and it can be calculated according to

$$\min \quad C = \max_{1 \le i \le n} \{C_i\}, \tag{1}$$

where C_i is the complete time of the ith job.

Table 1 shows an example of FJSP with 3 jobs and 2 machines. Each row and each column refer to an operation and one machine, respectively. In Table 1, the first row indicates that the first operation of $J1$ can be processed by

$M1$ (2 time unites) or $M2$ (3 time unites) or $M3$ (4 time unites). On the other hand, the sixth row shows that the third operation of $J2$ is allowed to be processed on $M1$ or $M3$. Symbol "—" means the machine cannot process corresponding operation.

As mentioned above, the production environments are uncertain, and several dynamic or random events such as machine failure, unexpected processing times, arrival of an urgent order, and cancellation of orders may often occur. When a dynamic event occurs, rescheduling is needed in order to adapt to the dynamic environments.

At present, there are three types of dynamic rescheduling strategies and they are periodic rescheduling, event-driven rescheduling, and hybrid periodic with event-driven rescheduling.

Event-driven rescheduling means that when the scheduling system encounters unexpected events such as machine failure and urgent order insertion while the original scheduling scheme is being executed, the system starts the rescheduling immediately and reoptimizes the remaining unprocessed operations according to the actual production conditions. Event-based scheduling has a good emergent response to unexpected events, but it lacks the global analysis. Periodic rescheduling means that the entire process is divided into several time periods according to the actual processing conditions and machine load, and each period is a cycle. The system automatically performs another rescheduling for the remaining unprocessed operations after completing one period until the end of the entire process. Periodic scheduling can have a global consideration on the system based on the production situation of each processing period, but it cannot effectively deal with some emergencies. The hybrid periodic and event-driven rescheduling strategy can make full use of their advantages; however the results obtained by this hybrid rescheduling method are not necessarily optimal.

A variable interval rescheduling strategy (VIRS) is proposed to deal with the dynamic flexible job shop scheduling problem in this paper. VIRS means that the operations in a certain range are rescheduled after the dynamic event occurs. The range is determined according to operations affected directly by the dynamic event. The operations in other intervals which are not directly affected by the dynamic events keep the initial optimized processing sequence. Meanwhile, the corresponding machines are also unchanged.

The steps for determining the rescheduling interval are as follows.

Step 1. Determine the affected jobs by the dynamic events.

Step 2. Delete the affected jobs from the initial scheduling scheme.

Step 3. Merge the remaining unaffected jobs.

Step 4. Reschedule the affected and unaffected jobs by the dynamic events together.

Table 2: Processing information of FJSP with 10 jobs and 5 machines.

Job	Operation	Processing time				
		$M1$	$M2$	$M3$	$M4$	$M5$
$J1$	O_{11}	2	3	2	5	2
	O_{12}	5	3	2	4	3
	O_{13}	5	2	5	2	4
$J2$	O_{21}	2	2	2	2	4
	O_{22}	5	3	4	4	3
	O_{23}	3	3	3	3	5
$J3$	O_{31}	2	2	3	2	5
	O_{32}	3	5	5	3	4
	O_{33}	4	2	3	5	4
$J4$	O_{41}	4	4	4	4	2
	O_{42}	2	4	3	5	2
	O_{43}	2	4	2	3	3
$J5$	O_{51}	3	4	2	3	5
	O_{52}	5	3	2	2	4
	O_{53}	4	4	4	5	2
$J6$	O_{61}	3	3	2	3	5
	O_{62}	2	4	2	3	5
	O_{63}	4	2	2	2	4
$J7$	O_{71}	5	2	4	5	5
	O_{72}	3	5	4	5	4
	O_{73}	3	4	5	2	4
$J8$	O_{81}	2	2	4	5	3
	O_{82}	4	2	3	5	5
	O_{83}	2	2	4	3	4
$J9$	O_{91}	4	3	5	4	4
	O_{92}	2	3	2	4	5
	O_{93}	2	2	3	2	5
$J10$	O_{101}	4	2	3	4	3
	O_{102}	4	4	2	4	5
	O_{103}	2	5	2	3	3

Step 5. Determine the starting time t_1 and the finishing time t_2 of the job directly affected by dynamic events after rescheduling.

Step 6. The rescheduling interval is $[t_1, t_2]$.

Taking the processing information of FJSP in Table 2 as an example, firstly, a static scheduling scheme is obtained and the scheduling Gantt chart is shown in Figure 1. Assume that $M1$ is failure and cannot be repaired at time 5. Therefore jobs 2, 7, 8, and 9 are affected directly by the dynamic event, as shown in Figure 2. The rescheduling Gantt chart obtained by using the dynamic scheduling strategy based on variable rescheduling interval is presented in Figure 3. It can be seen from Figure 3 that the second operation of job 5 is shifted backward, and the third operation of job 3 and the first operation of job 1 are shifted backward so that the processing order of the job which is not directly affected by the dynamic

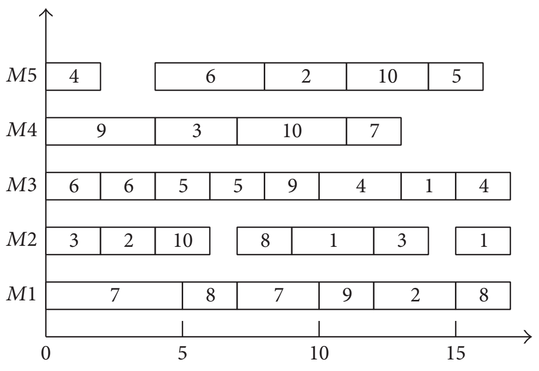

Figure 1: Static scheduling Gantt chart.

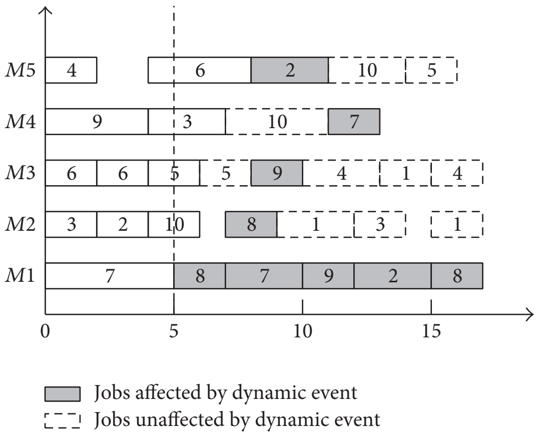

Figure 2: $M1$ failure at time 5.

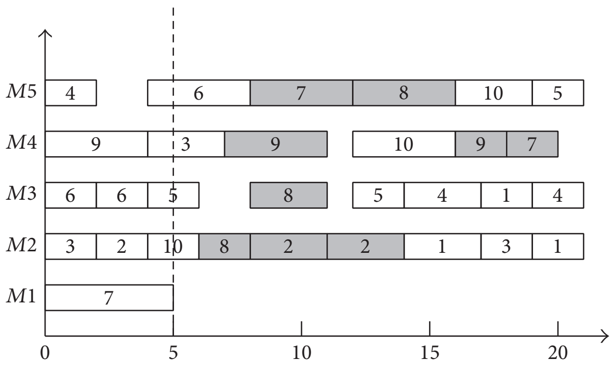

Figure 3: Dynamic rescheduling Gantt chart.

event keeps unchangeable. Meanwhile, the corresponding machines are also unchanged.

3. Proposed Genetic Algorithm

3.1. Encoding. As mentioned above, there are two subproblems in the FJSP; therefore the feasible encoding of FJSP has two parts. Figure 4 shows a feasible solution for the instance shown in Table 1. The first part of the chromosome is the sequence of operations while the second part is the machine allocation for executing the operations in the first part.

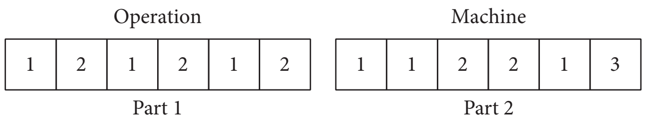

Figure 4: Chromosome structure.

The chromosome consists of 12 genes. Numbers 1 and 2 which belong to part 1 represent jobs $J1$ and $J2$, respectively. As $J1$ and $J2$ both have three operations, therefore numbers 1 and 2 appear three times in part 1. On the other hand, numbers 1, 2, and 3 which occur in part 2 represent machines $M1$, $M2$, and $M3$, respectively. For example, the first gene, 1, in part 2 means that the first operation of $J1$ is to be processed by machine $M1$, the last gene, 3, in part 2 means that the third operation of $J2$ is to be processed by machine $M3$.

3.2. Initialization. The quality of the initialization population plays an important role for GA's performance. The random initialization population is used by most researches, while this method probably generates the inferior population. As a result, the population number or evolutionary number needs to be enlarged to obtain the near-optimal solution. Kacem et al. proposed a localization method for generating initial population [34]; however, this method is dependent strongly on the order in which machines and operations are given in the table [35]. A mix of random initialization population by combining initialization machine and initialization operation with random initialization is proposed in this paper.

3.2.1. Initialization Machine. Since the objective is the makespan, therefore the machine with less processing time has a priority to process the corresponding operation. The processing order of the jobs is not considered in this stage, and the specific steps are as follows.

Step 1. Read time table T for all processing operations.

Step 2. Find the minimal value $T\_\min_i$ for each operation.

Step 3. Determine the probability (see (2)) for selecting machine to process corresponding operation according to $T\_\min_i$. In (2), apq stands for the total operations of the ith job.

Step 4. Select randomly machine according to the selection probability calculated in Step 3.

Step 5. Repeat Step 4 until the initialization machine is finished.

$$p_i = \frac{1/T\_\min_i}{\sum_i^{apq}\left(1/T\_\min_i\right)}. \qquad (2)$$

3.2.2. Initialization Operation

Step 1. Read the current processing information table.

Step 2. Calculate the completion time mak_cur_i after selecting an optional job.

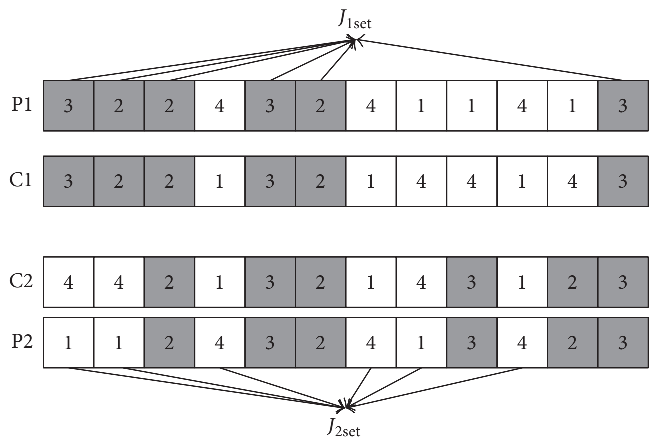

FIGURE 5: Crossover operation for operation part.

Step 3. Determine the probability for selecting each job according to (3).

Step 4. Determine the selected job according to the selection probability calculated in Step 3.

Step 5. Repeat Step 4 until the initialization operation is finished.

$$p_i = \frac{1/mak\_cur_i}{\sum_i^n (1/mak\_cur_i)}. \quad (3)$$

3.2.3. Random Initialization

Step 1. Select randomly an optional machine.

Step 2. Select randomly the job from an optional set of the jobs.

Step 3. Repeat Steps 1 and 2 until the initialization population is finished.

3.3. Selection. The selection based on elitist strategy (ES) and improved population diversity strategy (IPDS) is used in this paper. The elitist strategy can copy some of the best individuals from one parent to the offspring. Compared with the traditional probabilistic selection, the pure elitist strategy can keep the best solution monotonically improving from one generation to the next. However this may lead to a local minimum. Therefore, the IPDS is used to avoid falling into the local optimal solution. The specific steps of IPDS are as follows.

Step 1. Read the population information *pop*, and set i and j to 1 ($1 \leq i \leq pops$), respectively.

Step 2. Get the ith individual of the *pop*.

Step 3. Let $j = i + 1$ and then compare the ith individual and the jth individual. If individuals i and j have high similarity, then delete the ith individual.

Step 4. Repeat Step 3 and $j = j + 1$.

Step 5. Repeat Steps 2, 3, and 4 until $i = pops - 1$.

3.4. Crossover Operation. For operation part, the precedence operation crossover (POX) in [36] is used. Assume that chromosomes P1 and P2 are two parents and chromosomes C1 and C2 are two offspring, the detailed crossover steps of the POX are as follows.

Step 1. Randomly divide the job set $\{1, 2, 3, \dots, n\}$ into two nonempty subsets J_{1set} and J_{2set}.

Step 2. Copy the genes belonging to J_{1set} in chromosome P1 to C1. Copy the genes belonging to J_{1set} in chromosomes P2 to C2 and keep these genes in the same position.

Step 3. Copy the genes belonging to J_{2set} in chromosomes P2 to C1. Copy the genes belonging to J_{1set} in chromosomes P1 to C2 and keep these genes in the same order.

An example is shown in Figure 5.

For machine part, a multipoint crossover operation is adopted. The procedure of crossover operation is as follows: A set rand(0, 1), which is composed of 0 and 1, is randomly generated. The length of the chromosome is equal to the machine encoding. Exchange the genes, which are the same as the genes in position 1 in set rand(0, 1), in parents P1 and P2, and then create two offspring C1 and C2.

In addition, for partial flexible job shop scheduling problem (P-FJSP), when the machine number is greater than the total number of the available machines by this crossover operation, then one machine is randomly selected for processing the corresponding operation.

An example is shown in Figure 6.

3.5. Mutation Operation. For operation part, a mutation method by swapping genes is adopted. The detailed mutation procedure of this method is shown as follows.

Step 1. Select randomly two corresponding positions in parent P1.

Step 2. Swap the genes to generate an offspring C1.

An example is shown in Figure 7.

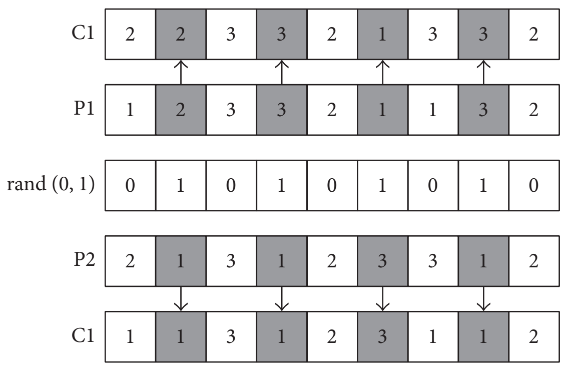

Figure 6: Crossover operation for machine part.

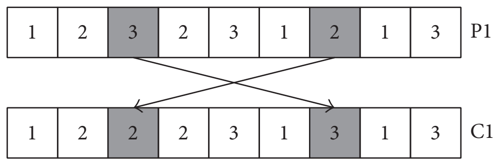

Figure 7: Mutation operation for operation part.

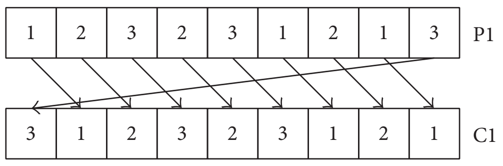

Figure 8: Mutation operation for machine part.

Table 3: Processing information for 5-job and 5-machine FJSP.

Job	Operation	Processing time				
		M1	M2	M3	M4	M5
$J1$	$O11$	5	4	7	4	10
	$O12$	4	4	10	3	9
	$O13$	5	8	3	6	10
$J2$	$O21$	8	9	9	9	4
	$O22$	3	2	2	1	7
	$O23$	8	2	4	7	8
$J3$	$O31$	7	2	1	9	9
	$O32$	8	7	7	8	2
	$O33$	4	6	1	7	1
$J4$	$O41$	1	8	7	9	8
	$O42$	7	8	3	4	10
	$O43$	8	3	1	6	6
$J5$	$O51$	3	7	7	10	6
	$O52$	2	10	9	5	10
	$O53$	2	2	5	9	5

Table 4: Comparison of the initializing population methods.

Method	The fitness of the best individual	The average fitness of the initial population
Improved initializing population	18.92	33.71
Random initializing population	26.02	42.15

For machine part, a mutation method by moving genes backward in the chromosome P1 is used to generate C1.

If the machine number obtained by this method is greater than the total number of the available machines by this crossover operation, then a machine is selected in the machine set to process the corresponding operation in a random manner.

An example is shown in Figure 8.

The flow chart of the proposed improved genetic algorithm and dynamic rescheduling strategy is shown in Figure 9.

4. Experiments and Results

4.1. Experiments of the Initialization Population Method. In order to verify the effectiveness of our proposed initialization population method, we compare our improved genetic algorithm with the basic genetic algorithm on the fitness of the best individual, the average fitness of the initial population, and the evolutional generation. Table 3 shows the processing information for 5-job and 5-machine FJSP.

Parameters are as follows: population size is 100, evolutionary generation is 100, the crossover probability is 0.7, and the mutation probability is 0.1.

Table 4 shows the comparison results by using our proposed improved initializing population method and random initializing population method by running randomly 100 times. It can be seen from Table 4 that our proposed initializing population method can obtain high-quality individuals.

Figure 10 shows the evolutionary curve obtained by the traditional random initializing population method. Figure 11 shows the evolutionary curve obtained by our proposed improved initializing population method. It can be seen that the random method takes more than 60 generations to obtain the optimal solution, while the proposed improved initializing population method only needs less than 30 generations to obtain the optimal solution. What is more, compared with the basic GA, the optimal solution obtained by using our improved GA is also better than that. Therefore, the evolutionary speed and solution quality can be improved obviously by using our proposed improved initializing population method. Figure 12 shows the Gantt chart.

4.2. Experiments of the Dynamic Rescheduling Strategy. In order to verify the effectiveness of our proposed improved genetic algorithm and dynamic rescheduling strategy based on variable interval scheduling, the comparative experiments were done. The case about FJSP from literature [33] is shown in Table 5. The static scheduling Gantt chart obtained by our proposed improved genetic algorithm is shown in Figure 13, and the optimal makespan is 35 which is smaller than the optimal value obtained by Wu et al. [33].

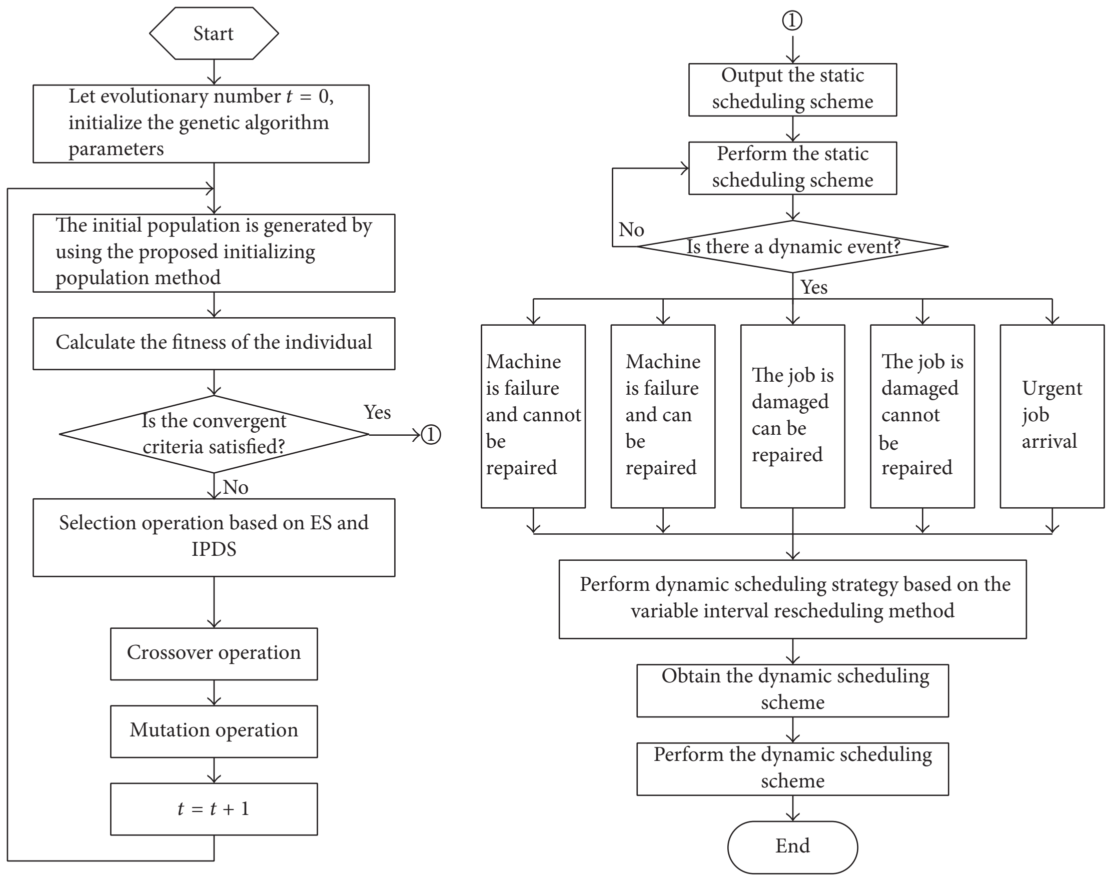
Start
Let evolutionary number $t = 0$, initialize the genetic algorithm parameters
The initial population is generated by using the proposed initializing population method
Calculate the fitness of the individual
Is the convergent criteria satisfied?
Yes
①
No
Selection operation based on ES and IPDS
Crossover operation
Mutation operation
$t = t + 1$
①
Output the static scheduling scheme
Perform the static scheduling scheme
Is there a dynamic event?
No
Yes
Machine is failure and cannot be repaired
Machine is failure and can be repaired
The job is damaged can be repaired
The job is damaged cannot be repaired
Urgent job arrival
Perform dynamic scheduling strategy based on the variable interval rescheduling method
Obtain the dynamic scheduling scheme
Perform the dynamic scheduling scheme
End

FIGURE 9: Flow chart of the proposed improved genetic algorithm and dynamic rescheduling strategy.

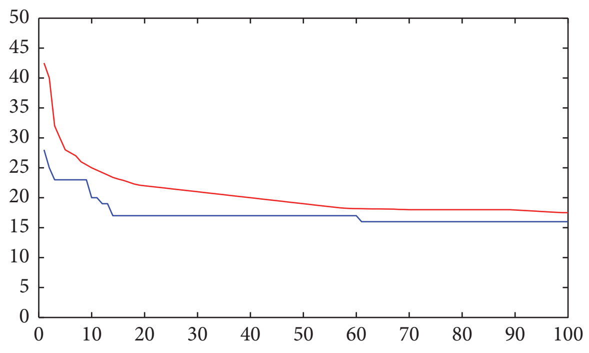

FIGURE 10: Evolutionary curve obtained by random initializing method.

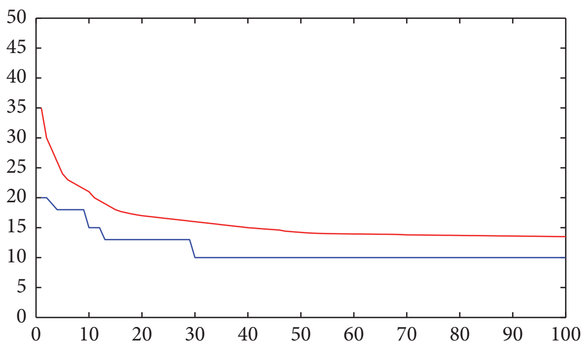

FIGURE 11: Evolutionary curve obtained by proposed initializing method.

The dynamic events often happen in an actual production environment. In this section, the machine failure is considered. Suppose that $M6$ is failure at time 20 s, the rescheduling Gantt chart is shown in Figure 14, and the comparison results for rescheduling are shown in Table 6.

From Table 6, it is not difficult to see that the involving operations of the jobs that need to be processed on the failure machine can be directly transferred to other machines with considerable processing capacity after using the dynamic scheduling strategy based on the variable interval rescheduling method proposed in this paper, and there is no need to wait for the failure machine to continue processing the involving operations after repairing, so that the completion time is greatly reduced.

4.3. Dynamic Rescheduling for Some Other Dynamic Events. In Section 4.2, the machine failure is considered. In order to further verify the effectiveness of our proposed improved genetic algorithm and dynamic rescheduling strategy based on variable interval scheduling, the following dynamic events

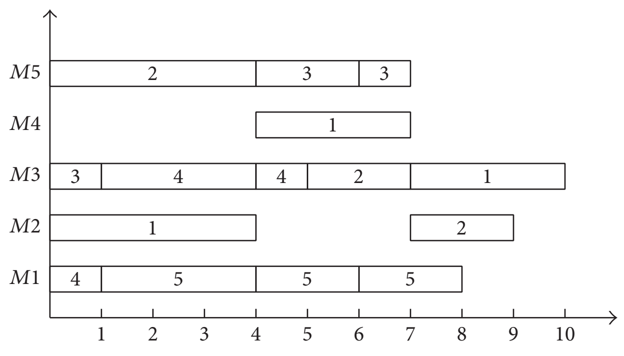

Figure 12: Gantt chart.

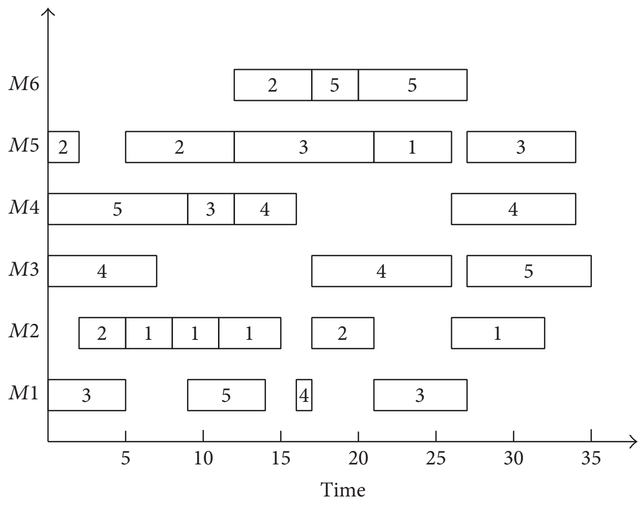

Figure 13: Static scheduling Gantt chart.

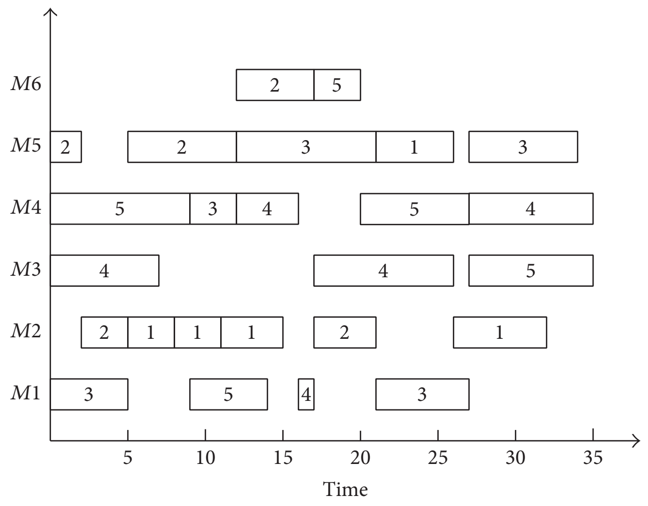

Figure 14: Rescheduling Gantt chart after $M6$ failure.

Table 5: Processing information for 5-job and 6-machine FJSP.

Job	Operation	Processing time					
		$M1$	$M2$	$M3$	$M4$	$M5$	$M6$
	O11	2	3	4	—	—	—
	O12	—	3	—	2	4	—
$J1$	O13	1	4	5	—	—	—
	O14	4	—	—	3	5	—
	O15	—	6	8	7	6	9
	O21	3	—	5	—	2	—
	O22	4	3	—	—	6	—
$J2$	O23	—	—	4	—	7	11
	O24	—	5	—	7	—	5
	O25	4	5	7	—	5	—
	O31	5	6	—	—	—	—
	O32	—	4	—	3	5	—
$J3$	O33	—	—	13	—	9	12
	O34	6	5	7	4	8	—
	O35	8	6	—	—	7	8
	O41	9	—	7	9	—	—
	O42	—	6	—	4	—	5
$J4$	O43	1	—	3	—	—	3
	O44	6	—	9	7	5	4
	O45	—	8	7	8	8	—
	O51	4	3	7	9	3	6
	O52	5	6	—	4	—	5
$J5$	O53	6	—	4	—	—	3
	O54	—	5	—	7	—	7
	O55	7	—	8	7	8	—

are considered: (1) machine failure: it can be repaired after a period of time; here we name this kind of dynamic events as DE1; (2) the job is damaged and it cannot be repaired; here we name this kind of dynamic events as DE2; (3) the job is damaged and it can be repaired; here we name this kind of dynamic events as DE3; (4) urgent job arrival; here we name this kind of dynamic events as DE4.

The processing information in Table 5 is still used in our experiments; the dynamic event descriptions are shown in Table 7. The static scheduling scheme is shown in Figure 13.

(1) Dynamic Rescheduling for DE1. When DE1 happens, $M6$ cannot be used between time 20 and time 30. The normal scheduling result is shown in Figure 15. However, if dynamic rescheduling strategy based on variable interval scheduling is carried out, the rescheduling result is shown in Figure 16. From Figures 15 and 16, it can be seen that the scheduling time is reduced 10 units after performing our proposed dynamic rescheduling strategy.

(2) Dynamic Rescheduling for DE2. When DE2 happens at time 20, job 4 is damaged and it cannot be repaired; therefore job 4 needs to be rescheduled. The normal scheduling result is shown in Figure 17. However, if the dynamic rescheduling strategy is carried out, the rescheduling result is shown in Figure 18. From Figures 17 and 18, it can be seen that the scheduling time is reduced by 9 units after performing our proposed dynamic rescheduling strategy.

Table 6: Comparison results for rescheduling.

	Actual enterprise's scheduling result	Method proposed in [33]	Our proposed method
Failure time	20	20	20
Machine repair time	30	0	0
Ideal makespan	37	37	35
Actual makespan	67	37	35

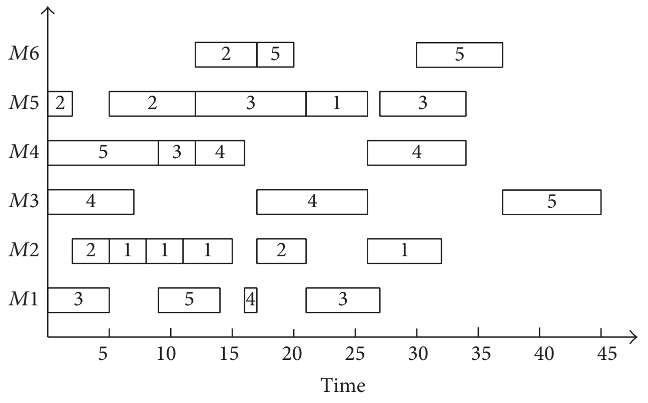

Figure 15: Scheduling Gantt chart by waiting for $M6$ repair.

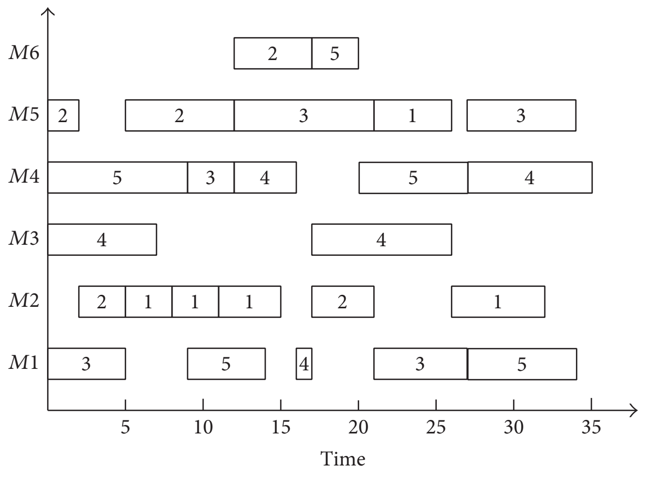

Figure 16: Rescheduling Gantt chart by using dynamic rescheduling strategy.

Table 7: Information for dynamic events.

Type of dynamic events	Dynamic event descriptions
DE1	At time 20, $M6$ is failure, and it is repaired after time 30
DE2	At time 20, job 4 is damaged and it cannot be repaired
DE3	At time 20, the fourth operation of job 4 is damaged and it can be repaired
DE4	At time 20, an urgent job 6 arrives

Table 8: Processing information of job 6.

Job	Operation	Processing time					
		$M1$	$M2$	$M3$	$M4$	$M5$	$M6$
$J6$	$O61$	6	7	3	6	2	5
	$O62$	6	7	2	6	7	2
	$O63$	2	2	5	5	2	3
	$O64$	3	4	7	7	2	3
	$O65$	3	3	5	7	5	5

Table 9: Comparisons of dynamic events.

Dynamic event	Makespan before optimization	Makespan after optimization
DE1	45	35
DE2	52	43
DE3	37	35
DE4	50	37

(3) Dynamic Rescheduling for DE3. When DE3 happens at time 20, the fourth operation of job 4 is damaged and it can be repaired; therefore this operation needs to be reprocessed. The normal scheduling result is shown in Figure 19. However, if the dynamic rescheduling strategy is carried out, the rescheduling result is shown in Figure 20. From Figures 21 and 22, it can be seen that the scheduling time is reduced by 2 units after performing our proposed dynamic rescheduling strategy.

(4) Dynamic Rescheduling for DE4. When DE4 happens at time 20, urgent job 6 arrives, and it is necessary to be added to the current scheduling sequences. The processing information of job 6 is shown in Table 8. The normal scheduling result is shown in Figure 21. However, if the dynamic rescheduling strategy is carried out, the rescheduling result is shown in Figure 22. From Figures 21 and 22, it can be seen that the scheduling time is reduced by 13 units after performing our proposed dynamic rescheduling strategy.

The scheduling results for these dynamic events are shown in Table 9. It can be seen from Table 9 that the makespans obtained by using our proposed method for all these dynamic events are better than that before optimization. Therefore, we firmly believe that the proposed method is feasible and effective for static and dynamic events.

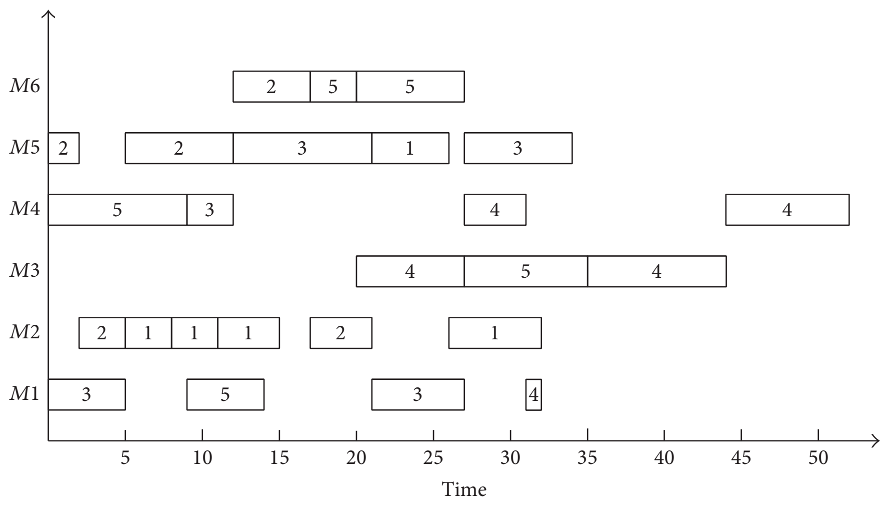

Figure 17: Scheduling Gantt chart for DE2 before optimizing.

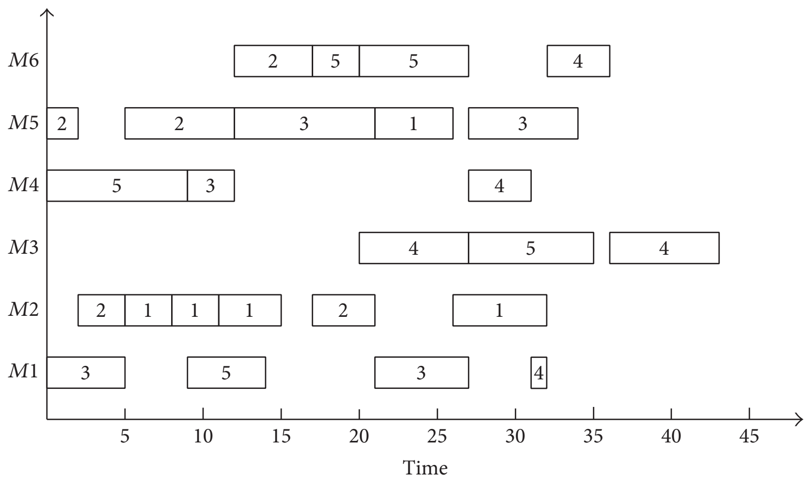

Figure 18: Scheduling Gantt chart for DE2 after optimizing.

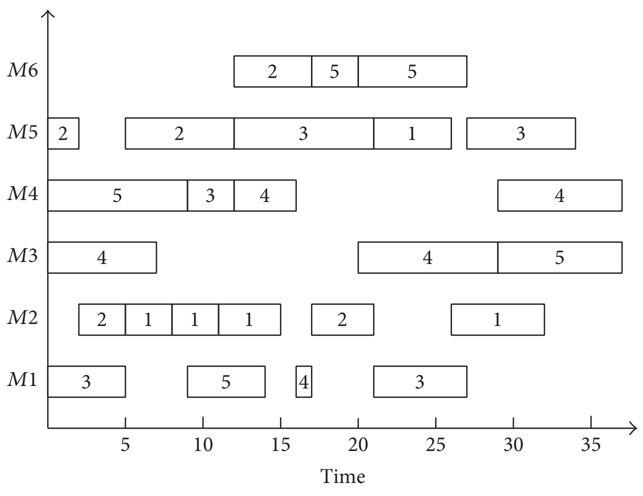

Figure 19: Scheduling Gantt chart for DE3 before optimizing.

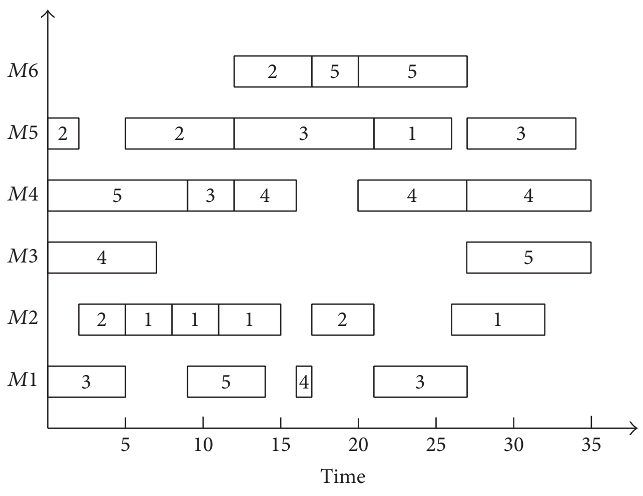

Figure 20: Scheduling Gantt chart for DE3 after optimizing.

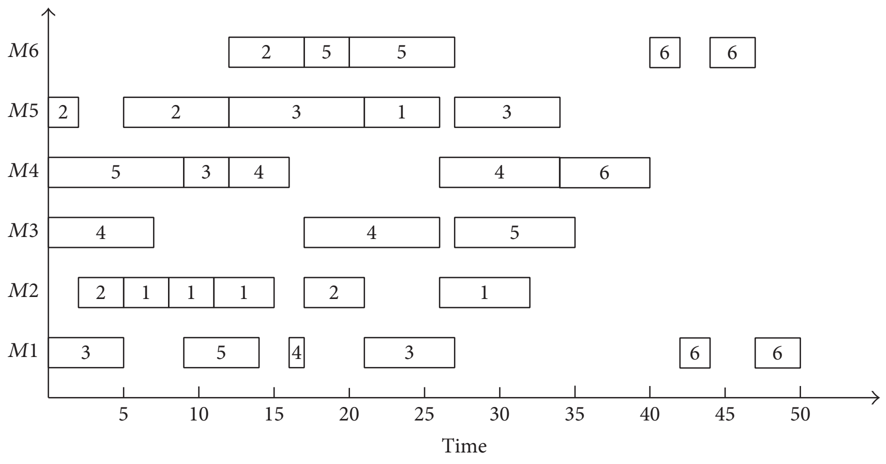

Figure 21: Scheduling Gantt chart for DE4 before optimizing.

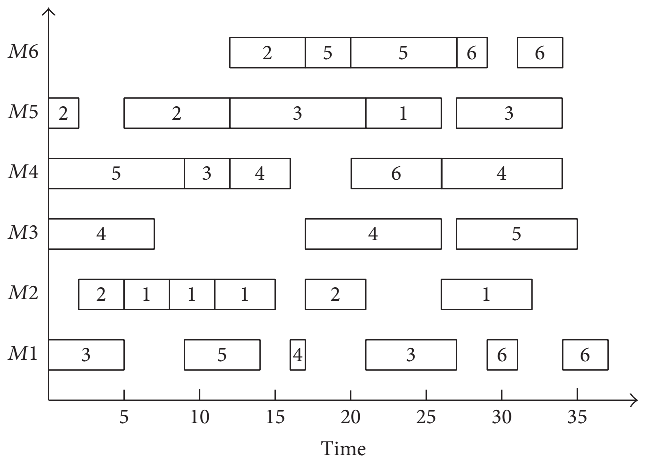

Figure 22: Scheduling Gantt chart for DE4 after optimizing.

5. Conclusions

The flexible job shop scheduling problem is known as NP-hard combinatorial optimization, and dynamic flexible job shop scheduling is very important to the implementation of the real-world manufacturing systems. In order to respond to dynamic flexible job shop scheduling problem for minimizing makespan, this paper develops an improved genetic algorithm based on variable interval rescheduling strategy for generating new schedules. Several dynamic events are carried out to evaluate the performance of the proposed method. Experimental results for static and several dynamic events in the FJSP demonstrate that our method is feasible and effective.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper is supported by Anhui Provincial Natural Science Foundation (Grant no. 1708085ME129), the National Natural Science Foundation of China (Grant no. 51305001), and Key Support Projects for Outstanding Young Talents of Anhui Province Universities (Grant no. gxyqZD2016125).

References

[1] W. Teekeng and A. Thammano, "Modified genetic algorithm for flexible job-shop scheduling problems," *Procedia Computer Science*, vol. 12, no. 12, pp. 122–128, 2012.

[2] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, 2011.

[3] X.-N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Information Sciences*, vol. 298, pp. 198–224, 2015.

[4] P. Fattahi, M. S. Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 18, no. 3, pp. 331–342, 2007.

[5] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[6] N. Shivasankaran, P. S. Kumar, and K. V. Raja, "Hybrid sorting immune simulated annealing algorithm for flexible job shop scheduling," *International Journal of Computational Intelligence Systems*, vol. 8, no. 3, pp. 455–466, 2015.

[7] G. Vilcot and J.-C. Billaut, "A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem," *International Journal of Production Research*, vol. 49, no. 23, pp. 6963–6980, 2011.

[8] A. Rossi, "Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships," *International Journal of Production Economics*, vol. 153, pp. 253–267, 2014.

[9] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Computers in Industry*, vol. 81, pp. 82–95, 2016.

[10] L. Wang, G. Zhou, Y. Xu, S. Wang, and M. Liu, "An effective artificial bee colony algorithm for the flexible job-shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 1–4, pp. 303–315, 2012.

[11] I. Driss, K. N. Mouss, and A. Laggoun, "A new genetic algor ithm for flexible job-shop scheduling problems," *Journal of Mechanical Science and Technology*, vol. 29, no. 3, pp. 1273–1281, 2015.

[12] S. H. A. Rahmati, M. Zandieh, and M. Yazdani, "Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 64, no. 5–8, pp. 915–932, 2013.

[13] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9–12, pp. 1159–1169, 2011.

[14] L. De Giovanni and F. Pezzella, "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem," *European Journal of Operational Research*, vol. 200, no. 2, pp. 395–408, 2010.

[15] Q. Zhang, H. Manier, and M.-A. Manier, "A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times," *Computers & Operations Research*, vol. 39, no. 7, pp. 1713–1723, 2012.

[16] G. Zhang, L. Gao, and Y. Shi, "A novel variable neighborhood genetic algorithm for multi-objective flexible job-shop scheduling problems," *Advanced Materials Research*, vol. 118-120, pp. 369–373, 2010.

[17] D. Rahmani and R. Ramezanian, "A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: a case study," *Computers and Industrial Engineering*, vol. 98, pp. 360–372, 2016.

[18] B. Liu, Y. Fan, and Y. Liu, "A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem," *Computers and Industrial Engineering*, vol. 87, pp. 193–201, 2015.

[19] J. Xiong, L.-N. Xing, and Y.-W. Chen, "Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns," *International Journal of Production Economics*, vol. 141, no. 1, pp. 112–126, 2013.

[20] N. Al-Hinai and T. Y. Elmekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm," *International Journal of Production Economics*, vol. 132, no. 2, pp. 279–291, 2011.

[21] L. Liu, H. Y. Gu, and Y. G. Xi, "Robust and stable scheduling of a single machine with random machine breakdowns," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7-8, pp. 645–654, 2007.

[22] F. Zhao, J. Wang, J. Wang, and J. Jonrinaldi, "A dynamic rescheduling model with multi-agent system and its solution method," *Strojniski Vestnik/Journal of Mechanical Engineering*, vol. 58, no. 2, pp. 81–92, 2012.

[23] L. Nie, L. Gao, P. Li, and X. Shao, "Reactive scheduling in a job shop where jobs arrive over time," *Computers and Industrial Engineering*, vol. 66, no. 2, pp. 389–405, 2013.

[24] M. A. Adibi, M. Zandieh, and M. Amiri, "Multi-objective scheduling of dynamic job shop using variable neighborhood search," *Expert Systems with Applications*, vol. 37, no. 1, pp. 282–287, 2010.

[25] P. Fattahi and A. Fallahi, "Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability," *CIRP Journal of Manufacturing Science and Technology*, vol. 2, no. 2, pp. 114–123, 2010.

[26] N. Kundakci and O. Kulak, "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem," *Computers and Industrial Engineering*, vol. 96, pp. 31–51, 2016.

[27] E. Ahmadi, M. Zandieh, M. Farrokh, and S. M. Emami, "A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms," *Computers and Operations Research*, vol. 73, pp. 56–66, 2016.

[28] X. Qiu and H. Y. K. Lau, "An AIS-based hybrid algorithm with PDRs for multi-objective dynamic online job shop scheduling problem," *Applied Soft Computing Journal*, vol. 13, no. 3, pp. 1340–1351, 2013.

[29] L. Zhang, L. Gao, and X. Li, "A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3516–3531, 2013.

[30] A. A. R. Hosseinabadi, H. Siar, S. Shamshirband, M. Shojafar, and M. H. N. Md. Nasir, "Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in Small and Medium Enterprises," *Annals of Operations Research*, vol. 229, no. 1, pp. 451–474, 2014.

[31] M.-S. Lu and R. Romanowski, "Multicontextual dispatching rules for job shops with dynamic job arrival," *International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1-4, pp. 19–33, 2013.

[32] H. Karimi, S. H. A. Rahmati, and M. Zandieh, "An efficient knowledge-based algorithm for the flexible job shop scheduling problem," *Knowledge-Based Systems*, vol. 36, no. 6, pp. 236–244, 2012.

[33] Z. Wu, H. He, and C. Huang, "Flexible job shop dynamic scheduling problem research with machine fault [J]," *Machine Design and Research*, vol. 31, no. 3, pp. 94–98, 2015.

[34] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 32, no. 1, pp. 1–13, 2002.

[35] J. Tang, G. Zhang, B. Lin, and B. Zhang, "A hybrid algorithm for flexible job-shop scheduling problem," *Procedia Engineering*, vol. 15, no. 1, pp. 3678–3683, 2011.

[36] X. Li and L. Gao, "An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem," *International Journal of Production Economics*, vol. 174, pp. 93–110, 2016.