Flowshop with Multiple Processors Dynamic Scheduling Problem based on Hybrid Genetic-Particle Swarm Optimization

Wang Li[1,*], Zhao Chuanyong[1] and Li Dawei[2]

[1]*School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, 114051, P.R. China;* [2]*School of Science, University of Science and Technology Liaoning, Anshan, Liaoning, 114051, P.R. China*

Abstract: Flowshop scheduling problem is a kind of typical scheduling problem, and has wide application background. While common in the flowshop environment, the objective of the flowshop with multiple processors dynamic scheduling problem is minimize the total cost of earliness/tardiness and deviation punishment. In this paper, a mathematical model of the problem is presented and a hybrid genetic-particle swarm optimization algorithm is constructed. The simulation results show that this is a better solving method for complex flowshop multiple processors dynamic scheduling problem.

Keywords: Dynamic scheduling, genetic algorithm, hybrid algorithm, multiple processor, particle swarm optimization.

1. INTRODUCTION

Flowshop scheduling problem (FSP, for short) was first proposed by Johnson in 1954 [1]. The problem can be described as: In a processing and manufacturing system, there are *N* jobs, each of which has the same processing route, and needs *S* processes. In all processes, at least one has multiple parallel processors. The goal is to determine the allocation of the parallel processors and the jobs processing sequence on each processor.

FSP caused extensive concerns of many scholars and researchers since it was put forward, and had been proved that it is a NP-complete problem, so it is difficult to get the optimal solution.

Flowshop with multiple processors (FSMP) is the extension upon the common FSP. Its typical characteristics is in parallel processors exist in some processes. So it is more complex than the common FSP on the solving method.

Salvador first defined the FSMP in 1973 [2]. In the next 20 years, many scholars and researchers studied on it, but main research results were about minimizing the total flow time, makespan, et al, in the static environment [3]. Rarely results were found about earliness/tardiness and deviation punishment in the dynamic environment [4, 5].

In recent years, intelligent algorithms such as artificial neural network (ANN), genetic algorithm (GA), particle swarm optimization (PSO), et al, have been used to solve FSP [6-8]. While GA and PSO are widely used because they can fast search the near optimization solutions. But these two algorithms have some weakness. For example, GA has lower local searching ability, not enough uses the feedback information in system, and prone to “premature” convergence in practical application [9]. While for PSO, it has lower searching precision, prone to fall into local optimal, the global optimal solution may not easy to be searched as the result [10].

Against disadvantages of the above two algorithms, this paper proposes a hybrid GA-PSO. The basic process of hybrid GA-PSO is that GA first obtains near optimal, which took as the best position in PSO, then updating particle’s velocity and position by formula. The optimal solution may be obtained, if all points are traversed. The simulation results show that the hybrid algorithm has advantages of solving complex scheduling problems.

2. PROBLEM DESCRIPTION

The essence of dynamic FSMP is that production systems are disturbed frequently by uncertainty factors, which exist in the whole production process. According to the need of actual information, production systems must re-schedule the production plan on time, in order to ensure the implementation of the original production plan, and as far as possible to reduce the disturbance loss.

According to the types of disturbances, time and the production conditions, jobs are divided into processed and unprocessed ones. The schedule can be determined by assigning priorities for processed jobs, and re-do the production plan for unprocessed ones.

This paper mainly studies about equipment disturbance, according to the actual production background of a manufacturing plant. A detailed description as follows:

After a processor is breakdown in processing process, the processed and unprocessed jobs which in the original scheduling plan must be determined, under the consideration of process constraints and the current resource states.

In the dynamic scheduling model, as one of the main consideration of the variables, the jobs can be divided into two kinds, one is known as processed, which are finished or already start to be processed but haven't, another is known as unprocessed, which haven't start to be processed. At this time, the starting time of some jobs must be adjusted, due to the presence of the processors breakdown. It is likely to make a lots time conflicts on a certain process. In the dynamic scheduling, in order to ensure the system stability, it requires that the deviation of the starting process time in the re-scheduled with the original and the changes in processing station as small as possible. On the other hand, in order to maintain the stability and efficiency of the enterprises, according to the real-time information after disturbance, to make the minimum total cost of earliness /tardiness and deviation punishment of each job.

Symbols description:

Job *i*, $i = 1, 2, \cdots, n$; process *j*, $j = 1, 2, \cdots, m$.

s_{ijk} , s''_{ijk} : job *i*'s starting time of the *j*th process in the *k*th station after/before dynamic schedule;

A_{im} : the allowed starting processing time of job *i* in the *m*th process;

c_{im} : the complete time of job *i* after dynamic schedule;

δ_{ijk} : the deviation punishment of job *i*'s processing time of the *j*th process in the *k*th station;

Ω_{ijk} : job *i*'s the earliest starting time of the *j*th process in the *k*th station;

$\Phi^{i}_{j,j+1}$: job *i*'s waiting form the *j*th process to the next process;

β_{im} : the earliness punishment of job *i* in the *m*th process;

γ_{im} : the tardiness punishment of job *i* in the *m*th process;

p_{im} : the processing time of job *i* in the *m*th process;

d_{im} : after disturbance, the latest complete time, or the due date of job *i*;

The decision variables are:

$x_{ijk}, x''_{ijk} = 1$, if job *i* is allocated to the *j*th process in the *k*th station after/before dynamic schedule, else $x_{ijk}, x''_{ijk} = 0$.

Objective 1: minimize the cost of earliness/tardiness punishment.

$$\min f_1 = \sum_{i=1}^{n} (\beta_{im} \max\{0, d_{im} - c_{im}\} + \gamma_{im} \max\{0, c_{im} - d_{im}\}) \quad (1)$$

Objective 2: minimize the deviation punishment after dynamic schedule.

A new schedule plan is necessary to generate, if disturbance occurs in processing process. Dynamic scheduling must guarantee the system stability, and reduce the degree of modification or change, such that the deviations of starting time of unprocessed in static/dynamic schedules and the changes in stations are as small as possible.

Aiming at the above, the objective function is established as fellows:

$$\min f_2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{k_j} \delta_{ijk} \left| x_{ijk} s_{ijk} - x''_{ijk} s''_{ijk} \right| \quad (2)$$

The total objective is $f = a_1 f_1 + a_2 f_2$, i. e. minimize the weighted sum of the earliness/ tardiness and the deviation punishment before and after dynamic schedule. That is:

$$\min f = a_1 f_1 + a_2 f_2 \quad (3)$$

The initial weighted values are given by experience, and $a_1 \le a_2$.

Constrain conditions:

$$\sum_{i=1}^{n} x_{ijk} = 1 \quad j = 1, 2, \cdots, m \quad (4)$$

$$\sum_{k=1}^{k_j} x_{ijk} = 1 \quad j = 1, 2, \cdots, m \quad (5)$$

$$s_{ijk} + p_{ij} \le s_{i(j+1)k} \quad j = 1, 2, \cdots, m-1 \quad (6)$$

$$s_{ijk} + p_{ij} \le s_{(i+1)jk} \quad i = 1, 2, \cdots, n-1 ; j = 1, 2, \cdots, m \quad (7)$$

$$\sum_{k=1}^{k_j} \sum_{i=1}^{n} x_{ijk} = n \quad j = 1, 2, \cdots, m \quad (8)$$

$$s_{(i+1)mk} = s_{imk} + p_{im} \quad i = 1, 2, \cdots, n-1 \quad (9)$$

$$s_{imk} \ge \mathrm{A}_{im} \quad (10)$$

$$s_{ijk} + p_{ij} = \Omega_{(i+1)jk} \quad i = 1, 2, \cdots, n-1 ; j = 1, 2, \cdots, m \quad (11)$$

$$\Omega_{i(j+1)k} = s_{ijk} + p_{ij} + \Phi^{i}_{j,j+1} ; \; j = 1, 2, \cdots, m-1 \quad (12)$$

At above, $i = 1, 2, \cdots, n$, and $k = 1, 2, \cdots, k_j$, from formulae (4) to (12).

Objective (3) is to minimize the punishment costs of earliness/tardiness and the changes of total jobs after disturbance. Formula (4) expresses that each process in each station can only process one job at the same time. Formula (5) expresses that each job can only be processed by one station in each process. Formula (6) expresses that each job can starts to the next process, if the current process has been finished. Formula (7) expresses that the next process can start, if the prior process has been finished. Formula (8) expresses that the jobs processed in all stations of each process must equal the total jobs. Formula (9) expresses that the processors continuous processing after dynamic scheduling. Formula (10) expresses that the job's starting time after dynamic scheduling must behind the allowed starting processing time. Formula (11) is process constraint to avoid waiting. And formula (12) is to calculate the time from prior process to the next.

3. HYBRID GENETIC-PARTICLE SWARM OPTIMIZATION

Based upon the basic GA and PSO, this paper proposes a hybrid GA-PSO algorithm, according to the advantages and disadvantages of two algorithms. The thought of hybrid algorithm is that GA first gains near optimal, which took as the best colony position in PSO (*gbest*). Then random initializes particle swarm in the feasible solution space and speed space, i. e. determines initial position and speed of particles. Finally, updates each particle's velocity and position by formula, and determines the best objective of each particle (*gbest*). After traversing all points, the particles with higher fitness are operated crossover and mutation of GA. The best solutions may be obtained, after these operations. The simulation results show that the hybrid algorithm has advantages of solving complex scheduling problems.

Hybrid algorithm, using GA's feature of global fast convergence, integrates GA into the each iteration of PSO, in order to improve the quality of the solution [11-13].

The procedure of solving FSMP dynamic scheduling by hybrid algorithm is as follows:

Step 1 Initializes the control parameters. Let M be the number of subpopulation, in which with L randomly generated individuals, and Q be the evolution generation.

Step 2 Chromosome presentation is by random key. This method can make the genetic operator to produce viable offspring, and the amount of computation will not increase with the amount of scheduling problems.

Chromosome is consists of S segments, each of which represents a process, and contains N genes. Gene a_{ij} in segment j $(1 \le j \le S)$ represents the ith process processed by the $\mathrm{INT}(a_{ij})$ th processor. Digit 0 is used to separate adjacent segments. So the chromosome length is $S \times N + N - 1$. Gene a_{ij} contains two parts, one is the integer in set $\{1, 2, \cdots, m_j\}$, another is the random decimal between interval (0, 1). The integer part presents the processor allocated to the job, and decimal part presents the job sequence in each processor.

For instance, a FSMP schedule problem with 3 jobs, 3 processes with 3, 2, and 1 parallel processors, respectively. A chromosome by random key is:

$\alpha = [2.3, 1.8, 2.5, 0, 2.8, 2.5, 1.7, 0, 1.1, 1.2, 1.6]$

The meaning of chromosome α is that in the first process, job 1 and job 3 are processed by processor 2, but because 2.3<2.5, job 3 is processed first, and then job 1, while job 2 is processed by processor 1. In the second process, job 3 is processed in processor 1, while job 1 and job 2 are processed in processor 2, but the processing sequence must be determined by job 1 and job 2's complete time in the first process. If job 1's complete time in the first process is earlier than job 2's, job 2 must be processed in the second processor, although 2.8>2.5. While job 1's complete time in the first process is the same as job 2's, job 1 must be processed in the second processor, because of 2.8>2.5.

Step 3 Randomly generated R initial population, and let evolution generation point be $P = 1$.

Step 4 Each subpopulation i $(1 \le i \le M)$ is carried out the following operation, and a new subpopulation may be generated.

(1) Definition fitness function $f(k)$.

(2) Select individual by roulette wheel, and the replication probability is:

$$P_k = f_k / \sum_{j=1}^{L} f_j \tag{13}$$

(3) Single point crossing is used to exchange the gene segments in the selected individuals.

Crossover operation is to exchange the gene segments of two chromosomes. The single point crossover is adopted based upon random key code.

For example, parent chromosomes A_1 and A_2 are:

A_1=[$\underline{2.2, 2.7}$, 2.3, 0, 1.5, 2.4, 2.2]

A_2=[$\underline{1.1, 2.1}$, 1.9, 0, 2.1, 1.4, 1.6]

Then the offspring chromosomes B_1 and B_2 are:

B_1=[$\underline{1.1, 2.1}$, 2.3, 0, 1.5, 2.4, 2.2]

B_2=[$\underline{2.2, 2.7}$, 1.9, 0, 2.1, 1.4, 1.6]

In where, the items with underlines express the segments to be exchanged. And new solution will be generated after exchanging. It can be seen that before crossing, job 1, job 2, and job 3 are all processed in processor 2, but because of 2.7>2.3>2.2, the processing sequence is 2→3→1. After single point crossing, job 1's first process is processed in processor 1, and the first process of job 2, and job 3 are processed in processor 2, but because of 2.1<2.3, job 3 is prior job 2. As the same reason, for parent A_2 after crossing, the first process of job 1 and job 2 is processed in processor 2, but job 2 is prior job 1. Job 3 is processed in processor 1.

(4) Mutation operation

Mutation operation may be carried out, if the offspring's fitness isn't evolved, premature convergence occurs, or the terminal condition isn't satisfied. Mutation operation is helpful to increase the diversity of population. Swap operation is adopted in this paper, and in order to ensure that the solution is still feasible, just swap the different genes on the same segment.

Table 1. The processing time of each job.

Job	Forming (H)	CNC (H)	Assembly (H)
1	43	35	30
2	45	40	30
3	52	35	31
4	45	30	33
5	45	30	35
6	50	35	40
7	50	33	34
8	35	30	35
9	40	40	35
10	40	35	25
11	44	30	32
12	40	50	35
13	50	35	40
14	40	40	36

For example, parent chromosome is:

X_1=[$\underline{1.1}$, 2.1, $\underline{2.3}$, 0, 1.5, $\underline{2.4}$, $\underline{2.2}$]

The offspring after mutation operation is:

Y_1=[$\underline{2.3}$, 2.1, $\underline{1.1}$, 0, 1.5, $\underline{2.2}$, $\underline{2.4}$]

In where, the items with underlines express the genes to be mutated. And new solution will be generated after mutation. Before mutation, the job 1's first process is processed in processor 1, and the first process of job 2 and job 3 are processed in processor 2, but job 3 is prior to job 2 because of 2.1<2.3. After mutation, the first process of job 1 and job 2 are processed in processor 2, job 1 is prior to job 2. Job 3's first process is processed in processor 1.

Step 5 Finding the best and the worst individuals in the current *M* subpopulation, and inserting the best to the *M* subpopulation to replace the worst one.

Step 6 If the convergence condition is satisfied, then return the optimal *R*, and go to step 14, else return to step 4.

Step 7 Let *R* be the population best position *gbest* in PSO.

Step 8 Randomly initializing the particle's position and velocity, and let c_1 and c_2 be the learning factors.

Step 9 Fitness calculating. Selecting the suitable fitness function, according to disturbance, and calculates each individual's fitness.

Step 10 Comparing the present objective value of each particle with *R*, i. e. *gbest*. Updating *pbest*. If terminal condition is satisfied, then go to step 11, else go to step 10.

Step 11 Updating new particle's position and velocity.

Step 12 Comparing the objective values of *pbest* and *gbest*, and then updating *gbest*.

Step 13 Terminal condition. If satisfies, then go to next step, else go to step 10.

Step 14 Output the optimal solution *gbest*.

By above hybrid GA-PSO algorithm, the FSMP problem may be solved better.

4. SIMULATION RESULTS

In order to verify the FSMP model and solving algorithm, an example is used to simulation. The detailed data are as follows.

There are 14 jobs, each of which needs 3 processes, namely forming → CNC → assembly. 3, 3 and 2 processors are equipped in 3 processes, numbered from 1 to 14, respectively. The processing time of each job's process is list in Table **1**.

The disturbance is occurred in forming 3 after 120min, as in Figs. (**1**) and (**2**). The maintain time of forming 3 is 400min, that means the latest complete time (min) are 120, 160, 190, 220, 270, 310, 350, 130, 160, 200, 240, 280, 320, and 360, respectively. Using dynamic scheduling by the hybrid GA-PSO, the simulation results are list in Tables **2-4**.

Some parameters are set as:

crossover probability is set as 0.6, mutation probability is set as 0.001, population size is set as 30, iteration times is set as 1000, $a_1 = a_2 = 0.5$, $d_{im} = c_{im} = 0.5$.

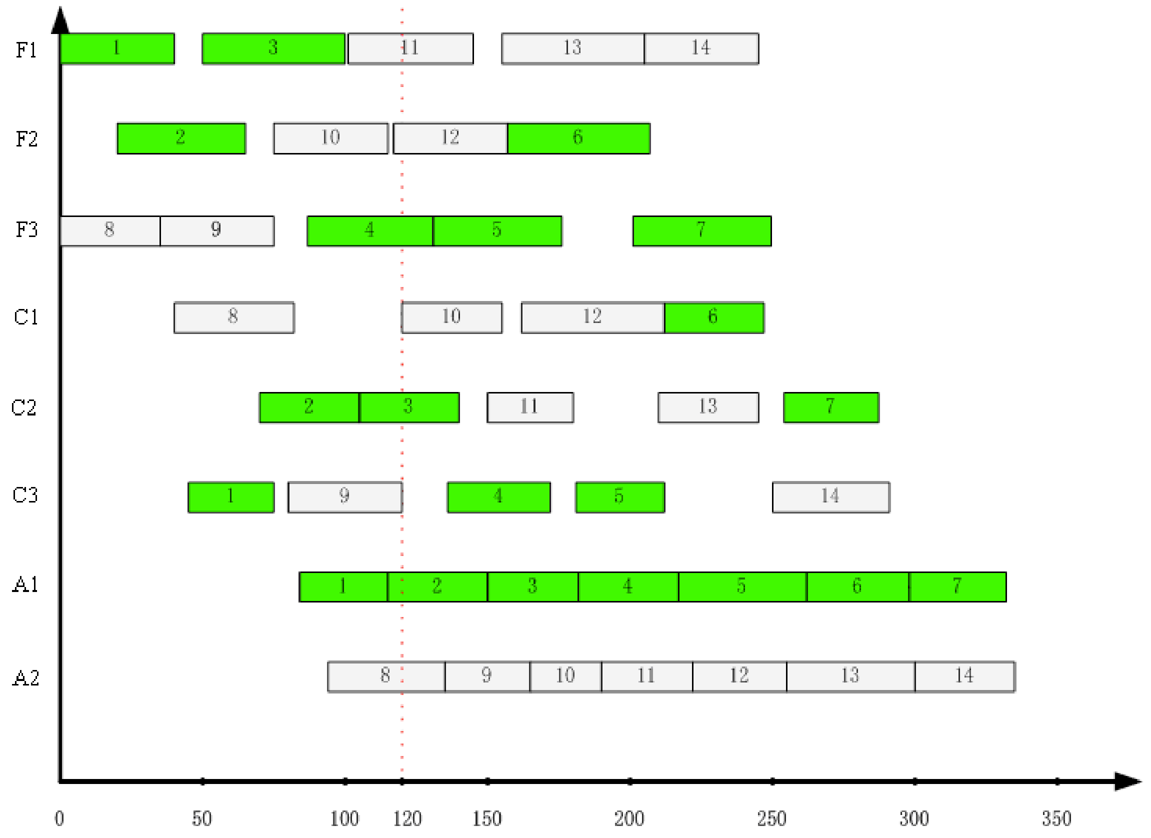

Fig. (1). Gantt chart before dynamic schedule in stations.

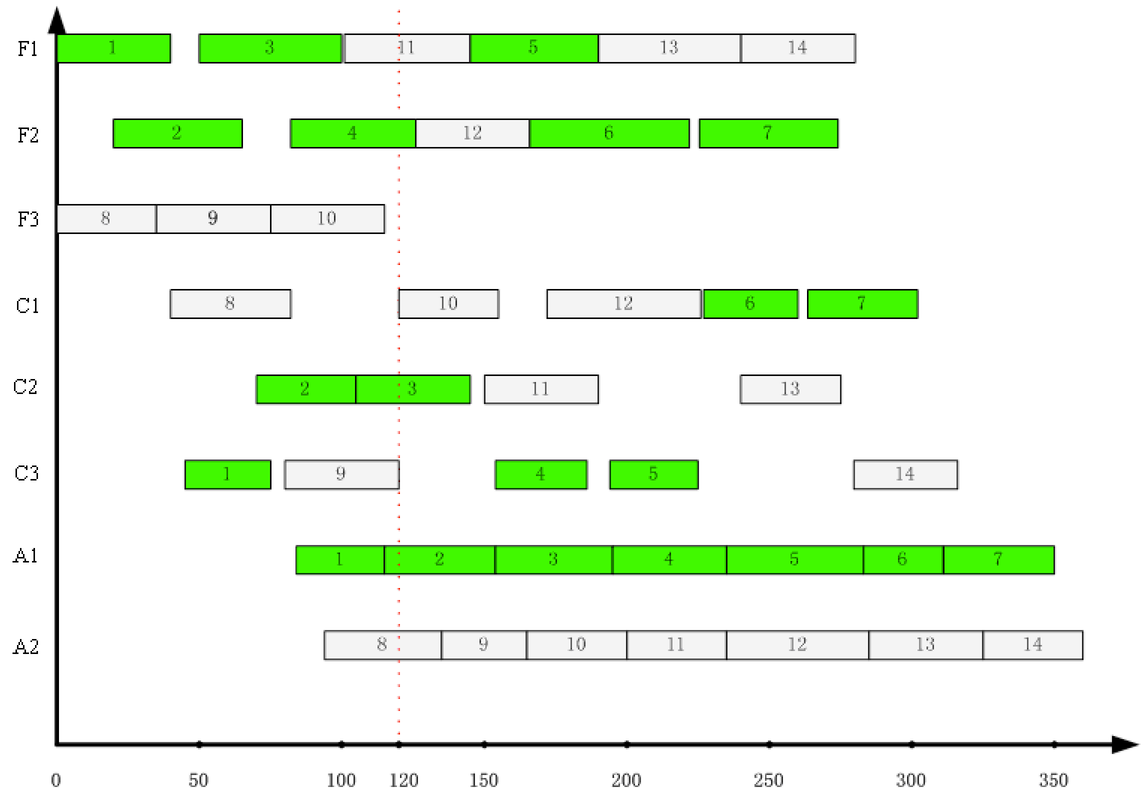

Fig. (2). Gantt chart after dynamic schedule in stations.

Table 2. The results of forming process.

Job	Forming (H)	Starting Time
1	1	0
2	2	20
3	1	50
4	2	81
5	1	145
6	2	165
7	2	225
8	3	0
9	3	35
10	3	75
11	1	101
12	2	126
13	1	190
14	1	235

Table 3. The results of CNC process.

Job	CNC (H)	Starting Time
1	6	45
2	5	70
3	5	105
4	6	155
5	6	195
6	4	226
7	4	267
8	4	40
9	6	80
10	4	120
11	5	150
12	4	171
13	5	240
14	6	280

Table 3. The results of assembly process.

Job	Assembly (H)	Starting Time
1	7	85
2	7	115
3	7	155
4	7	195
5	7	235
6	7	271
7	7	311
8	8	92
9	8	130
10	8	165
11	8	200
12	8	235
13	8	285
14	8	325

The Gantt charts before and after dynamic schedule in stations please see Figs. (**1**) and (**2**).

CONCLUSION

It is obvious seen compared before and after the dynamic scheduling that:

(1) The results after dynamic scheduling are still ensure the jobs without waiting for processing in each process and processed continuously.

(2) The schedules of forming 1 and 2 are more compact and all equipments use relative equilibrium, after disturbance occurs.

(3) The optimal solution obtained by hybrid GA-PSO is 950.5. It is can be seen from dynamic schedule Gantt chart that most jobs are processed in the original processors, changes are very limited, after disturbance occurred. This is better for arranging production.

In conclusion, this paper puts forward the dynamic scheduling model and the algorithm is feasible and effective.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work was financially supported by Liaoning Province University Innovation Team Project (2008T092) and the Natural Science Foundation of Liaoning Province (20092177).

REFERENCES

[1] S. Johnson, “Optimal Two-and Three-stage Production Schedules with Set-up Times Included”, *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61-68, 1954.

[2] M. S. Salvador, A Solution of Special Class of Flowshop Scheduling and its Application, Berlin: Springer-Verlag, 1973.

[3] Xu Jugang, Dai Guozhong and Wang Hongan, “An Overview of Theories and Methods of Production Scheduling”, *Journal of Compu-ter Research and Development*, vol. 41, no. 2, pp. 257-267, 2004.

[4] Li Suping, “The Production System is Living the Application in the Business”, *Coal Mine Machinery*, vol. 3, pp. 116-117, 2005.

[5] Xu Wenyan and Ye Wenhua, “Research and Implement of Production Scheduling System in MES”, *Machine Building and Automation*, vol. 34, no. 1, pp. 84-86, 2005.

[6] Y. Xin, “A Review of Evolutionary Artificial neural network”, *International Journal of Intelligent Systems*, vol. 8, no. 3, pp. 539-565, 1993.

[7] T. Murata, H. Ishibuchi, and H. Tanaka, “Genetic Algorithms for Flowshop Scheduling Problems”, *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 1061-1071, 1996.

[8] Jia Zhaohong, “Application and Research on Particle Swarm Optimization Algorithm in Flexible Job Shop Scheduling Problems”, Ph. D thesis, University of Science and Technology of China, Heifei, China, 2008.

[9] A. Bellabdaoui and J. Teghem, “A Mixed Integer Linear Programming Model for the Continuous Casting Planning”, *International Journal of Production Economics*, vol. 103, no. 2, pp. 102-120, 2006.

[10] Zhou Ming and Sun Shudong, Genetic Algorithm Principle and Applications, Beijing: Defense Industry Press, 2009.

[11] L. Davis, Genetic Algorithms and Simulated Annealing, London: Pitman, 1987.

[12] G. E. Vieira, J. W. Herrmann and E. Lin, “Rescheduling Manufacturing Systems: a Frame Work of Strategies, Policies, and Methods”, *Journal of Scheduling*, vol. 6, pp. 39-62, 2003.

[13] Fan Na and Yun Qingxia, “Particle Swarm Optimization and its Applications”, Information Technology, vol. 1, pp. 53-56, 2006.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015