Springer

KSME

Journal of Mechanical Science and Technology 24 (10) (2010) 2119~2115
www.springerlink.com/content/1738-494x
DOI 10.1007/s12206-010-0526-x

Research on flexible job-shop scheduling problem based on a modified genetic algorithm

Wei Sun[1], Ying Pan[1,2,*], Xiaohong Lu[1] and Qinyi Ma[1]

[1]*School of Mechanical Engineering, Dalian University of Technology, Dalian 116024, China*
[2]*Mechanical Engineering Institute, Dalian Fisheries University, Dalian 116023, China*

(Manuscript Received September 21, 2009; Revised April 1, 2010; Accepted April 4, 2010)

Abstract

Aiming at the existing problems with GA (genetic algorithm) for solving a flexible job-shop scheduling problem (FJSP), such as description model disunity, complicated coding and decoding methods, a FJSP solution method based on GA is proposed in this paper, and job-shop scheduling problem (JSP) with partial flexibility and JIT (just-in-time) request is transformed into a general FJSP. Moreover, a unified mathematical model is given. Through the improvement of coding rules, decoding algorithm, crossover and mutation operators, the modified GA's convergence and search efficiency have been enhanced. The example analysis proves the proposed methods can make FJSP converge to the optimal solution steadily, exactly, and efficiently.

Keywords: FJSP; GA; Coding rules; Decoding algorithm

1. Introduction

Scheduling of operations is an essential issue in the planning and managing of manufacturing processes. One of the most difficult problems in this area is the job-shop scheduling problem (JSP), which is a well-known NP-hard problem [1]. The classical JSP schedules a set of jobs on a set of machines with the aim to minimize a certain criterion, subjected to the constraint that each job has a specified processing order through all machines, which are fixed and known in advance. Compared with the traditional Job-shop Scheduling Problem (JSP, the flexible job-shop scheduling problem(FJSP) is an extension of the classical JSP, which is a more complex NP-hard problem. Each operation of FJSP job can be processed on any among a set of available machines and operation on different machines needs different time. Thus, FJSP reduces the machine constraints, enlarges the search range of feasible solution, and hence increases the difficulties greatly. Besides the determination of JSP machining sequences, FJSP has to appoint each operation to a machine among the available ones. Upon that the solution is more complicated.

FJSP has no deterministic algorithm with polynomial time complexity. So far, many relative solution methods are proposed such as dispatching rules, local search and meta-heuristics(such as tabu search), particle swarm optimization algorithm, simulated annealing and GAs [2-6], which can be classified into two main categories: hierarchical approach and integrated approach. The hierarchical approach attempts to solve the problem by decomposing it into a sequence of sub problems with reduced difficulty. For example, the decomposition step is to assign first, then to sequence. Results show that once the assignment is done, the resulting sequencing problem is a JSP. This approach is followed by Brandimarte [3], Paulli [7], Chambers and Barnes [8], among others. They all solve the assignment problem using some dispatching rules, and then solve the resulting JSP using different tabu search heuristics.

The integrated approach is more difficult to solve, but in general it can achieve better results; such reports are seen in Vaessens et al. [9], Dauzére-Pérés and Paulli [10], Hurink et al. [11] and Mastrolilli and Gambardella [2], who all adopt an integrated approach and propose different tabu search to solve FJSP. Among them, Mastrolilli and Gambardella present computational results that their tabu search performs better than any other heuristic developed so far, not only in terms of computation time, but also in solution quality.

Among them, GA uses the biological evolutionary mechanism to optimize and solve the problems by the method of "survival of the fittest, extinction of the unfitness" from an initial solution space. In comparison with other optimization methods, not only does GA have strong optimization ability, but also fast calculation, simple principle and operation, good

† This paper was recommended for publication in revised form by Associate Editor Kim, Dae-Eun
* Corresponding author. Tel.: +86041184707510, Fax: +86041184708812
E-mail address: panying@yahoo.cn

robustness and strong generality without restricted constraints and with implicit parallelism and global solution space searching ability. Therefore, GA is applied widely in production scheduling.

Recently, the solutions of FJSP with GA have mostly adopted integration methods for coding and various decoding methods. Chen et al. [12] separate one chromosome into two parts. The first part represents process routing; the second part represents machining sequence of each machine. Jia et al. [13] present an improved genetic algorithm to solve distributed scheduling problems which also can solve FJSP. Ho and Tay [14] propose an efficient methodology called GENACE based on a cultural evolutionary architecture for solving FJSP with recirculation. Kacem et al. [15, 16] combine both routing and sequencing information to present one chromosome, and evolve an approach through localization method to find promising initial assignments, and then apply dispatching rules to sequence the operations. Once the initial population is found, crossover and mutation operators are applied to jointly modify assignments and sequences. As the generations go by, better individuals will be produced. F.Pezzella et al. [17] integrate different strategies for generating the initial population, selecting the individuals for reproduction and reproducing new individuals to get better results. They are all integrated approaches, and different from each other for different coding scheme, initial population generation, chromosome selection and offspring generation strategies. However , the commonness of the above-mentioned GA methods for solving FJSP is a relatively complex coding scheme leading to complicated decoding, which induces increased operation time.

In this paper, a modified GA for FJSP is presented, in which the coding scheme adopts the direct coding method, the decoding algorithm is designed to be executed simply, and what's more, crossover and mutation operators are brief. For FJSP is hard to get optimal solution. Practical applications also require stable and efficient algorithms to quickly converge to an optimal solution. According to the above characteristics, such a practical GA is used for standard FJSP. Computation shows that this GA has strong stability and high efficiency; it can find the optimal solution in a short time. Additionally, we attempt to transform the partial flexibility and JIT request into standard problems so as to solve FJSP with a unified algorithm. Thus the application field and practicality of this GA can be extended and reinforced.

The remainder of this paper is organized as follows. In section 2 we give a standard FJSP description to unify the previous different mathematical descriptions. Section 3 gives a method to transform the partial flexibility and JIT request into standard FJSP. In section 4 we present the algorithm, by detailing the strategies to the coding scheme, the decoding algorithm, and the GA operators adopted to generate the offspring, the GA operation steps and example verification. Some conclusions are given in section 5.

2. The mathematical description of standard FJSP

The FJSP can be stated as follows. It is given a set $J = \{J_1, J_2, \ldots, J_n\}$ of independent jobs. A job (workpiece) J_i has a sequence $O_{i1}, O_{i2}, \ldots, O_{im_i}$ of operations to be performed one after the other according to the given sequence. It is given a set $U = \{M_1, M_2, \ldots, M_m\}$ of machines. Each operation O_{ij} can be executed on any among a subset $U_{ij} \subseteq U$ of compatible machines. The processing time of each operation is machine-dependent. We denote with T_{ij}^k the processing time of operation O_{ij} when executed on machine M_k. Table 1 is taken for an example. $p \times n$ operations O_{ij} are required to assign to m machines, and processing sequence of each operation should be decided as well as the beginning process time. Under the conditions of obeying the process sequence of the same workpiece, the production cycle, total load of machines or other operation index are to be optimized. Table 1 gives the data set of a 3×5 FJSP having 3 jobs operating on 5 machines. It's obviously a problem with total flexibility, and every job has equal number of operations. We term FJSP of this kind as the standard one, and will transform other more general ones into this standard form and solve them using the same unified method.

Following is the mathematical model for the goal of the optimal production cycle.

Design variables:

$$F_{ij}^k \in \{0,1\} \quad i \in N, j \in P, k \in M \tag{1}$$

$$\mathrm{ORD}_{ij} \in \{1, 2, \ldots, pn\} \quad i \in N, j \in P \tag{2}$$

0-1 variable F_{ij}^k presents choice relation between processes and machines. $F_{ij}^k = 1$ means that operation O_{ij} is distributed to machine M_k; integer variable ORD_{ij} is the processing sequence of O_{ij}, and the operation with smaller value is ahead of the bigger one. The starting process time is represented by ST_{ij}. The ending time is ET_{ij}. The optimization mathematical model is as follows:

$$\min \quad z = \max(ET_{ij}) \quad i \in N, j \in P \tag{3}$$

$$\text{St. } ET_{ij} = ST_{ij} + \sum_{k=1}^{m} F_{ij}^k T_{ij}^k \quad i \in N, j \in P \tag{4}$$

Table 1. Processing time of total FJSP.

workpiece	operation	Processing machine and time				
		M_1	M_2	M_3	M_4	M_5
J_1	O_{11}	1	3	4	5	4
	O_{12}	4	5	3	4	3
	O_{13}	2	2	5	3	2
J_2	O_{21}	3	3	5	7	3
	O_{22}	4	4	3	4	2
	O_{23}	7	3	2	5	3
J_3	O_{31}	4	5	3	5	4
	O_{32}	2	5	2	3	4
	O_{33}	3	4	7	4	3

$$\sum_{k=1}^{m} F_{ij}^{k} = 1 \qquad i \in N, j \in P \tag{5}$$

$$ST_{ij+1} \geq ST_{ij} \qquad i \in N, j \in \{1,2,...,p-1\} \tag{6}$$

$$ORD_{ij} \neq ORD_{i'j'} \qquad i,i' \in N, j,j' \in P, i \neq i', j \neq j' \tag{7}$$

$$ST_{ij} \geq \mathrm{sgn}(ORD_{ij} - ORD_{i'j'}) F_{ij}^{k} F_{i'j'}^{k} ST_{i'j'}$$
$$k \in M, i,i' \in N, j,j' \in P, i \neq i', j \neq j' \tag{8}$$

Formula (3) takes the finish time of the last operation as optimization objective, i.e., makespan; formula (4) is used to calculate the ending time of each operation; formula (5) ensures each operation can occupy only one machine; formula (6) keeps the right sequence of every operation for each workpiece; in formula (7), ORD_{ij} is guaranteed to obtain the combination from 1 to p × n; In formula (8), the starting process time of each operation is ensured to be zero or be after the completion of the former operation in the same machine. As can be seen from these formulas, FJSP is a nonlinear integer programming with huge search space.

3. General FJSP treatment

The above model is aimed at standard FJSP. In practice, FJSP usually manifests as more general forms. The following Table 2 corresponds to the problems with partial flexibility, JIT request and inconsistent operation number of each workpiece. The basic connotation of JIT production mode is as follows. From the view of enterprise economic benefit, product processing must meet a consignment period, which means processing completion is not the sooner, the better. Certainly, it should not be tardy too. On the one hand, completion ahead of schedule will increase inventory cost; on the other hand, tardiness will be punished according to contract. Workpiece completion at their respective delivery time is the ideal situation. So the scheduling target must ensure workpieces are produced when needed. Generally, minimum earliness/tardiness cost is taken as the performance index.

3.1 The treatment of partial flexibility

Some machine k cannot perform procedure O_{ij}, which is so-called partial flexibility. In this case, the process time is changed to a relative great value artificially, such as 100. Thus, the algorithm will automatically avoid the machine assignment of such operations.

3.2 The introduction of dummy operations and dummy machines

Dummy operations are introduced, and artificially all the processing time on all machines is set to zero, which can be used to guarantee the consistent operation number of each workpiece. The introduction of dummy preparatory operations and machines can meet the requirements of initial processing time, meeting the JIT requirements for a workpiece. For example, O_{11} of J_1 is changed to dummy preparatory activity.

Table 2. Processing time of general FJSP.

workpiece	JIT request		operation	Processing machine and time				
	ST⩾	ET⩽		M_1	M_2	M_3	M_4	M_5
J_1	2	8	O_{11}	1	3	4	-	-
			O_{12}	-	5	3	-	3
			O_{13}	-	2	5	3	2
J_2	0	10	O_{21}	3	-	5	-	3
			O_{22}	-	4	3	4	2
			O_{23}	7	3	2	5	3
J_3	1	9	O_{31}	4	5	-	5	-
			O_{32}	-	-	2	3	4

Table 3. Processing time after transforming into standard FJSP.

workpiece	operation	Processing machines and time					
		M_1	M_2	M_3	M_4	M_5	M_6
J_1	O_{11}	100	100	100	100	100	2
	O_{12}	1	3	4	100	100	100
	O_{13}	100	5	3	100	3	100
	O_{14}	100	2	5	3	2	100
J_2	O_{21}	100	100	100	100	100	0
	O_{22}	3	100	5	100	3	100
	O_{23}	100	4	3	4	2	100
	O_{24}	7	3	2	5	3	100
J_3	O_{31}	100	100	100	100	100	1
	O_{32}	0	0	0	0	0	100
	O_{33}	4	5	100	5	100	100
	O_{34}	100	100	2	3	4	100

Labor hour from M_1 to M_5 is 100. In the optimization process, the algorithm will automatically choose dummy machine M_6. The labor hour of M_6 is 2, which ensures O_{12} of J_1, that is the real operation O_{11} in Table 1, starts its processing after 2. Such method meets the starting time requirements of JIT for the workpiece. Thus, the original problem is changed from Table 2 to Table 3, and can be solved using unified GA.

4. A GA for FJSP

The key to FJSP solution is to decide the assignment between process and machine, and to arrange the order of the operations on each machine. Then, starting time and ending time of each operation are solved as well as other indexes. According to the problem's characteristics and requirement, a GA for FJSP is developed in this paper.

4.1 Coding

Coding methods of GA for JSP are divided into three categories: direct coding, indirect coding and coding with special knowledge of application problems. Among them, indirect coding is used widely. However, the shortcoming of indirect coding is its limited scope of application; thus we cannot get global optimization. Yuan et al. [18] use matrixes to represent

chromosomes, and put forward the GOR coding method. But this method belongs to direct coding; correspondingly, decoding is easy too. Without considering gene "0", they get a one-to-one correspondence between the chromosomes and the solution of practical problems. However, if this coding method is taken, random combination of genes will produce many illegal chromosomes. Furthermore, if the operators are selected unsuitably, many infeasible solutions will occur, which will increase calculation work and storage space, and will affect the convergence rate. Zhang et al. [19] adopt a bi-level coding scheme characterizing the FJSP problem. First part is coding based on operations to make sure the process sequence of operations; second one is based on machine assignment to select the process machine of each operation. Synthesis of the two parts coding can get the feasible solution of FJSP. But the course of this method is complex, leading to inconvenience of decoding.

n this paper, GA coding is composed of two parts also. First part is coding based on operations to make sure the process sequence of operations; second part is based on machine assignment to select the process machine for each operation. And the coding rule is one-level coding that succinctly expresses adoption of machine and process sequence of each operation for different workpieces. Correspondingly, the decoding is brief and intuitive, and will not produce an infeasible solution. In accordance with FJSP in Table 1, the following is a coding example.

$O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}|O_{11},O_{12},O_{13},O_{21},O_{22},O_{23},O_{31},O_{32},O_{33}$

（2，4，6，3，5，8，1，7，9 ‖1，2，2，3，3，3，4，2，5）

This coding represents operation O_{31} is arranged firstly and processed on M_4; the second arranging operation is O_{11} which is processed on M_1, and so on. Obviously, such a coding method can show that every process sequence of the same workpiece does not accord with the requirements. This problem will be adjusted successfully during the decoding course so as to guarantee the coding effectiveness.

4.2 Decoding

Decoding falls into three steps. First, each operation is assigned to corresponding machine according to the selection between sequence and machines in genes; then, the processing time of each operation is counted according to constraints described above; Thus, the starting time ST_{ij} and the ending time ET_{ij} of each process, working time MWT_k and ending time MET_k of each machine are all obtained. Lastly, the fitness function is generated, according to single-objective or multi-objective requirements. In Fig. 1 the flow chart of the decoding process is as follows. PUTIN(x,y) is the inserted matrix, x represents the inserted sequence. PUTIN(x,1), PUTIN(x,2) and PUTIN(x,3), respectively, represent workpiece number, process number and machine number; ST(i,j), ET(i,j), MET(k) and MWT(k), respectively, represent the starting time of operation, the ending time of operation, the machine idle time and the working time of machine; F(i,j) marks the process-inserting is finished; $p \times n$ is the total number of operations. This decoding method is faster and more concise than the inserted greedy decoding algorithm [20], and overall convergence is fast too, which meets the requirements of NP problems.

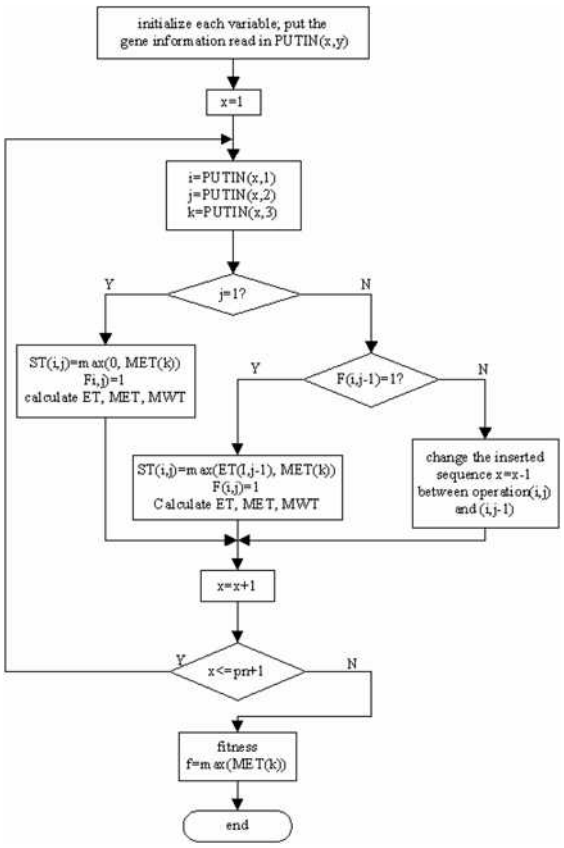

Fig. 1. Flow chart of decoding.

4.3 Crossover and mutation

Crossover operation changes some genes between two individuals at random to produce a new gene combination. Such operation is expected to compose beneficial genes so as to improve the local search ability of the algorithm, maintain the population diversity and prevent the premature phenomenon.

A. Crossover operators

Two parents are selected. Random positions of left half-side p_1, p_2 are taken; right half-side is p_3.

The left half-side, that is, sequence-inserted section, adopts one of the two parents to inverse the genes of p_1- p_2 as corresponding genes of offspring, and others are kept invariant; the right half-side, namely machine-selected part, uses the gene combination before and after p_3 position of two parents as offspring's genes.

(1, 5, 6, 2, 3, 7, 4, 9, 8‖3, 4, 5, 2, 4, 1, 3, 2, 3)

(2, 4, 6, 3, 5, 8, 1, 7, 9‖1, 2, 2, 3, 3, 3, 4, 2, 5)

p_1 p_2 p_3

(2, 8, 5, 3, 6, 4, 1, 7, 9‖1, 2, 2, 2, 4, 1, 3, 2, 3)

B. Mutation operators

One parent is chosen. Random positions of left half-side p_1, p_2 are taken; right half-side are $q_1$and q_2. Offspring will be obtained as long as the genes p_1/ p_2 and q_1/ q_2 of parents are exchanged.

(2, 4, 6, 3, 5, 8, 1, 7, 9‖1, 2, 2, 3, 3, 3, 4, 2, 5)

p_1 p_2 q_1 q_2

(2, 4, 8, 3, 5, 6, 1, 7, 9‖1, 2, 2, 4, 3, 3, 3, 2, 5)

As can be seen, the above operators can ensure the effectiveness of offspring's gene values. In this way, it is guaranteed that the ineffective genes cannot be generated during the algorithm operation; thus the reliability and efficiency of the algorithm are improved.

4.4 Operation steps of GA

The authors adopt GATOOL of MATLAB as the framework. Selection operation uses Elite mechanism. Namely, the Elite-Count chromosome individuals with best fitness would be inherited to next generation. The parents for crossover and mutation operation are generated from the individuals' remainder after Elite mechanism selection. Tournament is adopted, that is, from Tournament–Size individuals selected randomly, the best individuals are chosen as parents of crossover and mutation. Among the generated parents, crossover is carried out with crossover fraction proportion. The remainder part performs a mutation operation. The following are the steps.

(1) Initialize population, and generate population-size individuals at random;
(2) Decode, estimate individual fitness;
(3) Judge termination, output the results if Stopping criterion is satisfied;
(4) Select individuals according to Elite mechanism, reproduce to next generation directly;
(5) Generate parents using Tournament;
(6) Carry out crossover operation with crossover rate; the remainder part has mutation operations;
(7) Produce new population and return to (2).

4.5 Performance verification of GA

To test the performance of the algorithm in this paper, the data collected are same as that of KACEM et al.[16], Zhang et al.[4] and Zhang et al.[19]. These data are shown in Table 4. Here is a total FJSP for 10 workpieces with 10 machines. The available processing machine number for each operation of such a scheduling problem equals the total machine number.

Table 4. Test example for total FJSP (10×10).

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	O_{11}	1	4	6	9	3	5	2	8	9	5
	O_{12}	4	1	1	3	4	8	10	4	11	4
	O_{13}	3	2	5	1	5	6	9	5	10	3
J_2	O_{21}	2	10	4	5	9	8	4	15	8	4
	O_{22}	4	8	7	1	9	6	1	10	7	1
	O_{23}	6	11	2	7	5	3	5	14	9	2
J_3	O_{31}	8	5	8	9	4	3	5	3	8	1
	O_{32}	9	3	6	1	2	6	4	1	7	2
	O_{33}	7	1	8	5	4	9	1	2	3	4
J_4	O_{41}	5	10	6	4	9	5	1	7	1	6
	O_{42}	4	2	3	8	7	4	6	9	8	4
	O_{43}	7	3	12	1	6	5	8	3	5	2
J_5	O_{51}	7	10	4	5	6	3	5	15	2	6
	O_{52}	5	6	3	9	8	2	8	6	1	7
	O_{53}	6	1	4	1	10	4	3	11	13	9
J_6	O_{61}	8	9	10	8	4	2	7	8	3	10
	O_{62}	7	3	12	5	4	3	6	9	2	15
	O_{63}	4	7	3	6	3	4	1	5	1	11
J_7	O_{71}	1	7	8	3	4	9	4	13	10	7
	O_{72}	3	8	1	2	3	6	11	2	13	3
	O_{73}	5	4	2	1	2	1	8	14	5	7
J_8	O_{81}	5	7	11	3	2	9	8	5	12	8
	O_{82}	8	3	10	7	5	13	4	6	8	4
	O_{83}	6	2	13	5	4	3	5	7	9	5
J_9	O_{91}	3	9	1	3	8	1	6	7	5	4
	O_{92}	4	6	2	5	7	3	1	9	6	7
	O_{93}	8	5	4	8	6	1	2	3	10	12
J_{10}	O_{101}	4	3	1	6	7	1	2	6	20	6
	O_{102}	3	1	8	1	9	4	1	4	17	15
	O_{103}	9	2	4	2	3	5	2	4	10	23

The above GA is programmed in MATLAB, and implemented under the frame of GATOOL. The running environment is Pentium Dual-Core@3.0GHz，2GB memory. The GA parameters are as follows.

Population=100; Elite Count=4; Tournament Size=17; Crossover Fraction=0.5.

Table 5 compares our GA to the algorithms proposed by heuristic rules, traditional GA, methods of Kacem et al. and Zhang et al. in the case of different performance indexes. These index functions include the maximum makespan $C_{\max}$ and maximum workload of each machine $W_{\max}$. Each algorithm are programmed and implemented in same circumstance (MATLAB) and on same PC.

Seen from Table 5, GA in this paper obtains better results in both objective functions of $C_{\max}$ and $W_{\max}$ for 10×10 test example(In this example,7 and 5 are the best solution for $C_{\max}$ and $W_{\max}$ respectively), and have the shortest running time. Fig. 2 was created by GATOOL in MATLAB, showing the

Table 5. Comparison of test results (10×10).

	Heuristic rules	Classic GA	Localization +CGA	Zhang et al.	GA in this paper
C_{max}	16	16	7	7	7
time/s	0.1	-	23.3	22.1	13.3
W_{max}	16	14	6	5	5
time/s	0.1	-	17.6	20.3	11.5

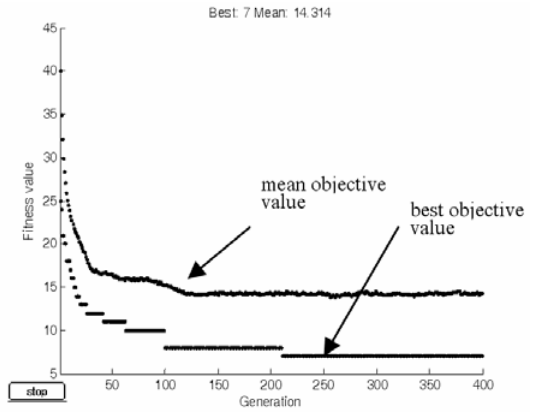

Fig. 2. The best and mean function value in each generation versus iteration number.

searching and converge ting course in solving the 10×10 example. It can be found that optimum individual converges towards optimal value rapidly within first 50 generations during the genetic process. At the same time, population gets better distribution, which demonstrates this algorithm has well stability and fast convergence.

Fig. 3 and Fig. 4 show a Gantt chart for the scheduling solution when C_{max} =7 and W_{max} =5 for 10 × 10 test example, respectively, which are drawn according to the information decoding from the best-fit gene individuals. From Fig. 1, we can tell that on machine 1 first operation of workpiece 1 and the second operation of workpiece 9 are performed in turn. Likely, the three operations of workpiece 2 are performed on machines 2, 4 and 10. It is obvious that Fig. 4 is sparser than Fig. 3, which is because in minimizing W_{max} we use a single objective model without considering C_{max} simultaneously. These results show that compared with other algorithms, the solution quality of the GA we use is improved greatly.

5. Conclusions

We have developed a genetic algorithm (GA) for the flexible job-shop scheduling problem (FJSP). The present GA for FJSP solution has lots of problems, such as a not unified description model, complicated coding and decoding methods. In this paper, the effective FJSP solution method is presented based on GA. Furthermore, a unified mathematical model is built and a new coding scheme of GA is given too. This approach solves the complex coding problems of the former research on FJSP, designs a simple decoding algorithm and crossover and mutation operators, which guarantees the gene

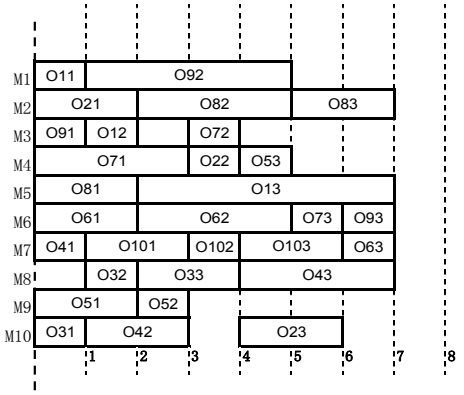

Fig. 3. Solution to C_{max} index (Makespan =7).

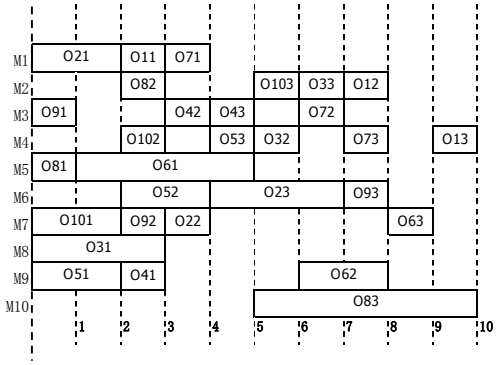

Fig. 4. Solution to W_{max} index (Wmax =5).

effectiveness when the algorithm runs. At the same time, a method is proposed which transforms the general FJSP with partial flexibility and JIT (just-in-time) request to standard FJSP. Through the test examples of the proposed GA and comparison with other algorithms, results show that such an algorithm can obtain the optimal solution steadily, efficiently and quickly. It accords with the requirements of FJSP for optimization algorithm.

Acknowledgment

This research supported by the Program for New Century Excellent Talents in University (NCET-05-0285), Ministry of Education, China.

References

[1] J. Blazewicz, G. Finke and G. Haopt, New trends in machine scheduling, *European Journal of Operational Research*, 37 (1988) 303-317.

[2] M. Mastrolilli andL. M. Gambardella, Effective neighbourhood functions for the flexible job shop problem, *Journal of Scheduling*, 3 (1) (1996) 3-20.

[3] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, 41 (1993) 157-83.

[4] H. P. Zhang andM. Gen, Multistage-based genetic algorithm

for flexible job-shop scheduling problem, *Complexity International*, 11 (2005) 223-232.

[5] J. Gao, M. Gen, L. Y. Sun and X. H. Zhao, A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems, *Computers & Industrial Engineering*, 53 (2007) 149-162.

[6] G. H. Zhang, X. Y. Shao , P. G. Li and L. Gao, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering* 56 (2009) 1309-1318.

[7] J. A. Paulli, Hierarchical approach for the FMS scheduling problem, *European Journal of Operational Research* 86 (1) (1995) 32- 42.

[8] J. W. Barnes and J. B. Chambers, Flexible Job Shop Scheduling by tabu search. Graduate program in operations research and industrial engineering, *Technical Report* ORP 9609, University of Texas, Austin, 1996. _http://www.cs.utexas.edu/users/jbc/_.

[9] R. J. M. Vaessens, E. H. L. Aarts and J. K. Lenstra, Job Shop Scheduling by local search, *COSOR Memorandum* 94-05, Eindhoven University, 1994.

[10] S. Dauzére-Pérés and J. Paulli, An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search, *Annals of Operations Research*, 70 (1997) 281-306.

[11] J. Hurink, B. Jurish and M. Thole, Tabu search for the job shop scheduling problem with multi-purpose machines, *OR-Spektrum* 15 (1994) 205-15.

[12] H. Chen, J. Ihlow and C. A. Lehmann, Genetic algorithm for flexible Job-shop scheduling, *IEEE international conference on robotics and automation*, Detroit (1999) 1120- 1128.

[13] H. Z. Jia, A. Y. C. Nee, J. Y. H. Fuh and Y. F. Zhang, A modified genetic algorithm for distributed scheduling problems, *International Journal of Intelligent Manufacturing*, 14 (2003) 351- 362.

[14] N. B. Ho and J. C. Tay, GENACE: an efficient cultural algorithm for solving the Flexible Job-Shop Problem, *IEEE international conference on robotics and automation* (2004) 1759-1766.

[15] 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, 32 (1) (2002) 1- 13.

[16] I. Kacem, S. Hammadi and P. Borne, Pareto-optimality Approach for Flexible Job - shop Scheduling Problems, *Hybridization of Evolutionary Algorithms and Fuzzy Logic*, Mathematics and Computers in Simulation, 60 (2) (2002) 245-276.

[17] F. Pezzella, G. Morganti and G. Ciaschetti, A genetic algorithm for the Flexible Job-shop Scheduling Problem , *Computers & Operations Research*, 35 (2008) 3202-3212.

[18] K. Yuan and J. Y. Zhu, Improved Genetic Algorithm for the Flexible Job - shop Scheduling with Multi-object, *China Mechanical Engineering,* 18 (12) (2007) 156-160.

[19] C. Y. Zhang, Y. Q. Rao, P. G. Li and X. Y. Shao, Bi-level genetic algorithm for the flexible job-shop scheduling problem, *Chinese Journal of Mechanical Engineering*, 43 (4) (2007) 119-124.

Wei Sun, born in 1967, is currently a professor and a PhD candidate supervisor in the School of Mechanical and Engineering, Dalian University of Technology, China. His main research interests include production scheduling, CIMS and optimization of complex mechanical equipment, mechanical transmission and structure CAD/CAE, management of product design resource and process. E-mail: sunwei@dlut.edu.cn

Ying Pan is currently a PhD candidate in the School of Mechanical and Engineering, Dalian University of Technology, China. Meanwhile, she serves as a lecturer in Mechanical Engineering Institute, Dalian Fisheries University, China. Her research interests include production scheduling, production engineering, manufacturing execution system and multi-agent system. E-mail: panying@yahoo.cn

Xiaohong Lu is currently a postdoctoral in the School of Mechanical and Engineering, Dalian University of Technology, China. Her main research interests include production scheduling, CIMS, production engineering, mechanical transmission and structure CAD/CAE, management of product design resource and process. E-mail: xiaohonglu@yahoo.cn

Qinyi Ma is currently a PhD candidate in School of Mechanical and Engineering, Dalian University of Technology, China. Her research interests focus on knowledge-based product digital design. E-mail: onlypony@163.com