Int j simul model 13 (2014) 4, 510-522
ISSN 1726-4529 Original scientific paper

AN IMPROVED GENETIC ALGORITHM FOR JOB-SHOP SCHEDULING PROBLEM WITH PROCESS SEQUENCE FLEXIBILITY

Huang, X. W.*; Zhao, X. Y. & Ma, X. L.
Faculty of Management and Economics, Dalian University of Technology, Dalian, P. R. China
E-Mail: huangxuewen@dlut.edu.cn (*Corresponding author)

Abstract
A new scheduling problem considering the sequence flexibility in classical job shop scheduling problem (SFJSP) is very practical in most realistic situations. SFJSP consists of two sub-problems which are determining the sequence of flexible operations of each job and sequencing all the operations on the machines. This paper proposes an improved genetic algorithm (IGA) to solve SFJSP to minimise the makespan, in which the chromosome encoding schema, crossover operator and mutation operator are redesigned. The chromosome encoding schema can express the processing sequence of flexible operations of all the jobs and the processing sequence of the operations on the machines simultaneously. The crossover and mutation operators can ensure the generation of feasible offspring for SFJSP. The simulation results on three practical instances of a bearing manufacturing corporation show that the proposed algorithm is quite efficient in solving SFJSP.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Process Sequence Flexibility, Job Shop Scheduling, Genetic Algorithm

1. INTRODUCTION

1.1 Problem statement

Scheduling is to allocate resources and time to tasks according to some rules and satisfying certain constraints such as operation sequence, the resource capacity and so on [1]. It has been proved that the scheduling problem involving more than two machines and two jobs is generally NP-hard [2]. In the classical job-shop scheduling problem (JSP), a set of independent jobs should be processed on several machines, and each job has an ordered set of operations, each of which must be processed on a predefined machine. The problem aims at finding the appropriate sequence of the operations on each machine to optimise some performance indicators, such as minimizing the maximum completion time over all jobs [3].

The flexibility in process planning includes operation flexibility, process flexibility and sequence flexibility [4]. The operation flexibility implies that each operation can be processed through a set of alternative machines. The process flexibility provides alternative process plans for each job. The sequence flexibility implies that all the operations in the process plan should be performed but the sequence of some operations in the process route of some jobs might not be fixed. This paper considers the sequence flexibility extension to JSP, which is very common as well as realistic. The sequence flexibility job-shop scheduling problem (SFJSP) is more difficult and complicated than the classical JSP in that the operation sequence of each job should be determined according to the real-time scheduling environment.

Due to the complications of the JSP, various methods like linear programming, heuristic algorithm and algorithm based on artificial intelligence, etc. have been introduced to solve the problem. Genetic algorithm is a popular meta-heuristic method used in solving combinatorial optimization problems. The advantage of GA with respect to the other search algorithms is due to the fact that more strategies could be adopted together to find good individuals [5]. GA has been successfully applied to solving the JSP for the optimal performance. In this paper, an

improved genetic algorithm (IGA) for solving SFJSP is proposed, which adopts an improved chromosome representation and a novel initialization approach. The maximum completion time over all jobs (makespan) is viewed as the performance measure to be optimized.

The remainder of the paper is organized as follows. Section 2 provides the literature review related to SFJSP. Section 3 introduces the formulation and constraints of SFJSP. In section 4, the mechanism of IGA for solving the SFJSP is explained in detail. The simulation results with IGA are presented in Section 5. In the final section, the conclusion and some further research suggestions are provided.

1.2 Literature review

The vital impact of flexibility on scheduling has been highlighted gradually since it is more realistic on the shop floor. The research literature concerning SFJSP is quite limited. But there are some works related to the problem considering the other two process planning flexibility in JSP.

The research of JSP with process planning flexibility started with the flexible job-shop scheduling problem (FJSP), which only considered the operation flexibility and has attracted much attention until now. The first research on the FJSP was presented by Brucker and Schlie [6]. A polynomial algorithm was designed for solving the problem with two jobs. For the increase of complexity of FJSP with more jobs, a variety of heuristic methods such as dispatching rules and meta-heuristic methods such as tabu search (TS) [7-10], simulated annealing (SA) [11, 12], genetic algorithm (GA) [13-18] and hybrid approaches [19-22] have been applied for solving FJSP.

With the consciousness of process flexibility in JSP, researches have focused on the integration of the flexible process planning and scheduling. The purpose is to choose an optimal or sub-optimal process plan for each job among the available ones to optimize the scheduling evaluation [23]. Saygin and Kilic [4] presented a framework that integrated the flexible process plan with off-line scheduling. Four stages were involved in the framework. The first stage was to decompose the flexible process plans into several linear alternative process plans (LAPPs). The second stage was to determine the machine tool for each operation in each LAPP among the alternative machine tools. In the third stage, only one process plan would be selected from the LAPPs for each part. After the process plan of each job was determined, in the fourth stage, scheduling was executed by an integer linear program. However, workload of process planning under this framework is very big. Brandimarte [24] proposed a hierarchical approach coping with a multi-objective scheduling problem with processing flexibility, which decomposes the problem into a machine loading problem and a scheduling sub-problem. The machine loading problem to determine the process plan for each job was formalized as an integer programming model using some heuristics to solve it. And then scheduling in the second problem was based on the results of the first sub-problem. Saygin et al. [25] suggested the dissimilarity maximisation method as a real-time decision-making tool for the process routing selection in off-line flexible manufacturing scheduling system. Jain et al. [26] proposed a scheme that consisted of the process plan selection module and scheduling module for the integration of process planning and scheduling. Li et al. [27] established a mathematical model for the flexible process planning selection problem and applied the genetic programming in solving the model. Shao et al. [28] applied the genetic algorithm to find the optimal process plan among alternative process plans to facilitate the integration of the process planning and scheduling functions. Ozguven et al. [29] established two mixed-integer linear programming models dealing with the job-shop scheduling problem with operation flexibility and process plan flexibility respectively. However, for solving the process flexibility job-shop scheduling problem, in most of existing

algorithms, selection of suboptimal process plan from flexible ones and schedule based on the selected process plan of each job are regarded as two separate tasks performed sequentially. Once a process plan is selected for a job, it cannot be changed on the shop floor. So there is still much room for improving the optimization performance by taking advantage of the process flexibility.

2. PROBLEM DESCRIPTION

2.1 Sequence flexibility description

The sequence flexibility can be defined as that job J_i consists of a sequence of n_i operations $\{O_{i,1}, O_{i,2}, \ldots, O_{i,j}, \ldots, O_{i,k}, \ldots, O_{i,n}\}$ and the sequence of the operations between $O_{i,j}$ and $O_{i,k}$ is not determinate. Then $<O_{i,j}, O_{i,k}>$ $(j < k)$ is called a flexible operation section, and the operations in the section are called flexible operations. The length of the flexible operation section can be expressed as $k-j+1$. Any flexible operation section is composed of one or more operations that can be placed at any position in the section, which is called free operation. Commonly, there are three types of flexible operation sections on the shop floor:

- T_1: O_s is a proper subset of operations in section $<O_{i,j}, O_{i,k}>$. Each operation in set O_s is a free operation that can be placed at any position in $<O_{i,j}, O_{i,k}>$. The process sequence of other operations besides O_s in section $<O_{i,j}, O_{i,k}>$ is fixed. Section of this type can be expressed as $T_1<<O_{i,j}, O_{i,k}>, O_s>$, which has $p = P_{k-j+1}^{|O_s|}$ kinds of permutation and combination on the operation sequence, where $|O_s|$ is the number of elements in O_s;
- T_2: in this type of flexible operation section all the operations are free operations, so the process sequence of them can be exchanged randomly. Section of this type can be expressed as $T_2<O_{i,j}, O_{i,k}>$, and the operations in the flexible operation section $<O_{i,j}, O_{i,k}>$ are called parallel operations, which has $(k-j+1)!$ kinds of permutation and combination;
- T_3: flexible operation section of this type is a hybrid of sections of type T_1 and T_2 for that in this type of section some flexible operation sections of type T_2 are included in one flexible operation section of type T_1. In order to distinguish from type T_1 and T_2, they are expressed as T_{31} and T_{32} respectively. Then the flexible operation section of type T_3 can be expressed as $T_3<T_{31}<<O_{i,j}, O_{i,k}>, O_s,>, T_{32}<O_{i,r}, O_{i,t}>\ldots T_{32}<O_{i,u}, O_{i,v}>>$, $(j \le r < t < \ldots < u < v \le k)$. $<<O_{i,j}, O_{i,k}>, O_s,>$ is the flexible operation section of type T_{31} and $<O_{i,r}, O_{i,t}>\ldots<O_{i,u}, O_{i,v}>$ are the flexible operation sections of type T_{32}.

The three types of flexible operation sections could describe the sequence flexibility completely. In order to make the sequence flexibility problem more understandable, there is an example of the inner ring grinding process plan for a spherical roller bearing, the sequence flexibility of which is very typical in bearing manufacture industry. The process plan with sequence flexibility is shown in Table I, which involves eleven operations. The third and the fourth operations are parallel operations, so the processing sequence of them can be exchanged. Then <3, 4> can be expressed as T_2 <3, 4>. The sixth operation can be placed at any position between the fifth and the eleventh operation, and the eighth and ninth operations are parallel operations. So this flexible operation section can be expressed as:

$$T_3 < T_{31} << 6, 10>, \{6\}>, T_{32} <8, 9>>.$$

The sequence flexibility of this parts can be expressed as:

$$T_2 <3, 4>, T_3 < T_{31} << 6, 10>, \{6\}>, T_{32} <8, 9>>.$$

Table I: Grinding process plan of a spherical roller bearing inner ring.

Operation no.	Operation name
1	Ring face grinding
2	Outside surface rough grinding
3	Inside surface rough grinding
4	Raceway rough grinding
5	Temper treatment
6	Chamfer brightening
7	Outside surface fine grinding
8	Inside surface fine grinding
9	Raceway fine grinding
10	Rib grinding
11	Surface brightening

2.2 SFJSP description

The SFJSP can be stated as follows: there are a set of jobs to be processed $J = \{J_1, J_2, \ldots, J_n\}$ and a set of machines $M = \{M_1, M_2, \ldots, M_m\}$. A number of n_i operations $\{O_{i1}, O_{i2}, \ldots, O_{ii}, \ldots, O_{ik}, \ldots, O_{in}\}$ has to be performed to complete job J_i, in which several different types of flexible operation sections might be included.

The SFJSP can be considered as consisting of many unique JSPs because of the sequence flexibility. For example, suppose that there are m_1 flexible operation sections of type T_1 and T_{31} as well as m_2 flexible operation sections of type T_2 and T_{32} in the problem, and the length of them are $n_{11}, n_{12}, \ldots, n_{1m1}$ and $n_{21}, n_{22}, \ldots, n_{2m2}$ respectively, then the SFJSP can be viewed as a set of $p_1 * p_2 * \ldots * p_i * \ldots * p_{m1} * n_{21}! * n_{22}! * \ldots * n_{2m2}!$, where p_i is the kinds of permutation and combination of the j^{th} flexible operation section of type T_1 and T_{31}. While JSP consists of only a sequencing problem because the process plan of each job is given in advance, the SFJSP turns into a routing problem to determine the operations sequence of each job due to the real-time scheduling environment as well as a sequencing problem to order operations on each machine. In this paper, the objective is to minimize the makespan, so the objective function is defined as $C = \min(\max C_k)$, $k = 1, 2, \ldots, m$, where C_k is the completion time of machine k.

The following assumptions are made for SFJSP:

(1) Jobs are independent of each other. There are no precedence constraints among the operations of different jobs.

(2) Machines are independent of each other. Each operation has to be processed on a predefined machine.

(3) All jobs and machines are available at time 0.

(4) Setup times and transportation times between operations are negligible.

(5) Each operation must be completed without interruption once it starts.

(6) A machine can process only one operation at a time, and one operation can only be processed on one machine at a time.

3. PROPOSED GENETIC ALGORITHM

The overall procedure of the proposed IGA is given in Fig. 1. There are two sub-problems in SFJSP: determination of flexible operation sequence and operations sequence on the machines. So IGA should be capable of solving the two sub-problems at the same time. First,

the initial population of IGA that is composed of a number of individual chromosomes is randomly generated. Each chromosome consists of two segments to deal with the two sub-problems. Each individual should be decoded to get the fitness measured by makespan. A new population will be formed by selecting the fitter individuals from the parent population. For every two individuals in the population, if a generated random value is less than a predetermined crossover probability p_c, then the crossover should be applied to both segments in the chromosomes and two offspring would be generated to replace the parent individuals in the population. After crossover, for each individual in the population, if a generated random value is less than a predetermined mutation probability p_m, the mutation will also be applied to both segments of the chromosome and a new offspring will be generated to replace the old individual. After the above operations, the population of a new generation is formed. For each generation, it is examined whether the algorithm has converged. The convergence is examined by both the expected objective value and the number of iteration predefined. If the algorithm has converged, the best individual of the current population is output as the optimal or sub-optimal solution. In general, except the selection mechanism, IGA has redesign the chromosome encoding schema、crossover operator and mutation operator Specifically for SFJSP, which are explained in detail in the following parts of this section.

```
begin
   I = Initialize(popsize);
   while(termination condition is not met) do
         F = Fitness(I);
         I = Select(I,F);
         for(i=0;i<popsize-1;i++)
             P1 = I(i);
             P2 = I(++i);
                if(random value < p_c )
                     [Off1,Off2]= Crossover(P1,P2);
                     replace P1 and P2 with Off1 and Off2 in I;
                end;
         end;
         foreach chromosome Ch    in I
              if(random value < p_m )
                   Mch = Mutation(Ch);
                   replace Ch with Mch in I;
              end;
         end;
       until converged;
   end;
   output the best solution;
end;
```

Figure 1: Pseudocode for IGA procedure.

3.1 Encoding scheme

A solution to SFJSP can be expressed by the processing sequence of flexible operations of all the jobs and the processing sequence of the operations on the machines. In IGA, a chromosome consists of the following two segments:

(1) Sequence of flexible operations of all the jobs

This part of chromosome is to determine the processing sequence of the flexible operations of each job, so that the process plan of each job can be determined. Genes of this part are composed of symbols and natural numbers. Suppose that there are s_j flexible operation sections in the process plan of job J_j, and the length of the flexible operation sections are $l_1, l_2, \ldots, l_{si}$ respectively. Then the total length of this chromosome segment is:

$$\sum_{j=1}^{n}\left(s_j+\sum_{p=1}^{s_j} l_p\right).$$

The expression of this segment is $T_i, O_{j,f1}, \ldots, O_{j,fn}, T_i, \ldots, T_i, O_{n,f1}, \ldots, O_{n,fn}$. T_i is the type of flexible operation section $<O_{j,f1}, O_{j,fn}>$, $T_i \in \{T_1, T_2, T_{31}, T_{32}\}$, and in the chromosome it is a symbol whose length is 1. $O_{j,f1}, \ldots, O_{j,fn}$ represents the processing sequence of the flexible operation section $<O_{j,f1}, O_{j,fn}>$, and in the chromosome they are natural numbers. For example, the expression $T_2, 5, 4, 7, 6$ means that the processing sequence of the flexible operations in section $T_2 < 4, 7>$ is $5, 4, 7, 6$.

(2) Sequence of operations on all the machines

This part of chromosome is to determine the processing sequence of all the operations on the machines. Operation-based representation is used for this part. Genes of this part are natural numbers. Gene value is the sequence number of a job. The total occurrence number of the job sequence number in this part equals to the total number of the job's operations. So the length of this part of chromosome equals to the total number of operations of all the jobs, which can be expressed as $\sum_{i-1}^{n} n_i$. The j^{th} appearance of a job sequence number from left to right represents the j^{th} operation in the process plan determined by the first chromosome segment of the job. For example, the expression $(1, 2, 2, 3, 3, 1, 1, 2)$ represents processing sequence of the operations $O_{1,1}$ - $O_{2,1}$ - $O_{2,2}$ - $O_{3,1}$ - $O_{3,2}$ - $O_{1,2}$ - $O_{1,3}$ - $O_{2,3}$. This encoding scheme can avoid generating infeasible individuals.

3.2 Decoding scheme

A feasible and active schedule could be obtained by decoding the chromosome. The first part of a chromosome should be firstly decoded to determine the process plan of each job. For the flexible operation section of type T_1 and T_2, the gene sequence in the chromosome is the sequence of the flexible operations in the section. For the flexible operation section of type T_3, the sequence of flexible operations can be firstly determined as the gene sequence of flexible operation sections of type T_{31} in the chromosome. And for that there are also some flexible operation sections of type T_{32} in the section of type T_{31}, so the operation sequence should be adjusted according to the gene sequence of flexible operation sections of type T_{32} in the chromosome after this section. Take the job shown in Table I for an example, suppose that the gene sequence of the flexible operation section $T_3 < T_{31} << 6, 10>, \{6\}>, T_{32} < 8, 9>>$ is $T_{31}, 7, 6, 8, 9, 10, T_{32}, 9, 8$, then operation sequence of the flexible operations in this section after decoded is $7, 6, 9, 8, 10$.

For the second part of a chromosome, from left to right, the operation that each gene represents can be determined by the decoded result of the first chromosome segment. For example, if the operation sequence of J_2 determined by the first chromosome segment is $1, 3, 2$, the sequence number corresponding to $O_{2,2}$ in expression $O_{1,1}$ - $O_{2,1}$ - $O_{2,2}$ - $O_{3,1}$ - $O_{3,2}$ - $O_{1,2}$ - $O_{1,3}$ - $O_{2,3}$ is 3. In the same way, the sequence number corresponding to $O_{2,3}$ is 2. Then the processing sequence of all the operations on each machine can be determined. Each operation will be placed on the appointed machine at the earliest possible start time.

3.3 Initial population

In order to keep the diversity of the population, each individual in the population is generated randomly. For the first part, the way of gene sequence generation depends on the type of the flexible operation section. For the flexible operation sections of type T_1 or T_{31}, the former operation of each free operation in the section is randomly chosen from a feasible region to determine the gene sequence. Take the flexible section $T_{31} << 6, 10>, \{6\}>$ for an example, the former operation sequence number of the sixth operation can be selected from the set $\{5, 7, 8,$

9, 10}. If the random generated operation sequence number is 7, the gene sequence of this section is 7, 6, 8, 9, 10. For the sections of type T_2 or T_{32}, a random sequence of the flexible operations in the section is generated to determine the gene sequence in the chromosome. For the second part, the occurrence of a job sequence number should equal to the number of the operations in its process plan to avoid generating unfeasible chromosome. This procedure satisfies the flexible relationship between the flexible operations as well as the precedence relationships between other operations.

3.4 Selection

The individual with higher fitness should have a higher probability of being chosen over others for the new population. In this paper, since the objective of the SFJSP is to minimize the makespan, the individual with less completion time should have higher fitness. Suppose that the population size is N and Ob_i means the maximum completion time of individual i, $Ob_{max} = \max\{Ob_k \mid k = 1, 2, \ldots, N\}$ and $F_i = Ob_{max} - Ob_i$. Then the fitness of individual i can be expressed as Eq. (1):

$$f_i = F_i \Big/ \sum_{j=1}^{N} F_j \tag{1}$$

Once the fitness value of each individual has been assessed, the roulette wheel selection is implemented to select individuals for the new population.

3.5 Crossover operator

For the different encoding schemes for the two segments of the chromosome, the crossover operations for the two parts are different. First, two chromosomes are selected from the current population as parents, and some feasible subsets of genes of the two segments should be selected independently to be swapped between two parents for generating two new chromosomes as offspring. The crossover operations for the two parts are as follows:

For the first part, several flexible operation sections need to be firstly selected by random for crossover. Genes in the selected sections are copied from one parent to the same positions of the offspring. The remaining genes are inherited from the other parent beginning with the first position.

For the second part, the partial matching crossover operation is applied [30, 31]. Suppose the parent individuals that need crossover are P_1 and P_2 respectively.

$$P_1 = (1,2,2,1,3,2,1,3) = (O_{1,1}, O_{2,1}, O_{2,2}, O_{1,2}, O_{3,1}, O_{2,3}, O_{1,3}, O_{3,2})$$

$$P_2 = (2,1,1,3,2,1,3,2) = (O_{2,1}, O_{1,1}, O_{1,2}, O_{3,1}, O_{2,2}, O_{1,3}, O_{3,2}, O_{2,3})$$

The crossover procedure of the second part is explained as follows:

Step 1: Select a gene section of P_1. Consider the selected section is 2, 1, 3, and the operations that genes in this section represent are $O_{2,2}$, $O_{1,2}$ and $O_{3,1}$;

Step 2: Genes in the selected section are copied to the same positions of offspring Off_1;

Step 3: The remaining genes of Off_1 are copied from genes on P_2 from left to right one by one except the genes that represent the operations $O_{2,2}$, $O_{1,2}$ and $O_{3,1}$.

Off_2 is generated in the same way. Suppose the selected gene section of P_2 is 3, 2, 1 and the corresponding operations are $O_{3,1}$, $O_{2,2}$ and $O_{1,3}$. Then the offspring Off_1 and Off_2 are as follows:

$$Off_1 = (2,1,2,1,3,1,3,2) = (O_{2,1}, O_{1,1}, O_{2,2}, O_{1,2}, O_{3,1}, O_{1,3}, O_{3,2}, O_{2,3})$$

$$Off_2 = (1,2,1,3,2,1,2,3) = (O_{1,1}, O_{2,1}, O_{1,2}, O_{3,1}, O_{2,2}, O_{1,3}, O_{2,3}, O_{3,2})$$

In this way, unfeasible chromosomes after crossover could be avoided. So there is no need for chromosome correction.

3.6 Mutation operator

The mutation operations for the two different chromosome segments are also different:

For the first chromosome segment, a flexible operation section is randomly selected. If type of the selected section is T_1 or T_{31}, a new former operation sequence number of the free operation is randomly selected from a feasible region, then a new gene sequence for this section could be generated. Else if type of the selected section is T_2 or T_{32}, reverse mutation operation is implemented. This means that the number gene sequence in the section should be reversed.

For the second chromosome segment, the reverse mutation operation is adopted. First, a gene section is randomly selected, and gene sequence in this section is reversed.

Obviously, the feasibility of the resulting offspring after mutation could be ensured.

4. SIMULATIONS AND ANALYSES

4.1 Simulation data

Due to the lack of availability of SFJSP benchmark problems, simulation experiments have been performed on three instances selected from a bearing company located in Dalian, China. Each instance can be characterized by number of jobs (n) and maximum number of operations within all the jobs (O_{max}).

4.1.1 Problem 5×5

A small scale SFJSP instance of 5×5 with 22 operations and 4 machines is shown in Table II. Each operation can only be processed on a predefined machine with a processing time. The data of Table II is the processing equipment and time of the j^{th} operation of job J_i. The symbol '–' indicates that J_i doesn't have that operation. The flexible operation sections of each job are shown in the second column of Table III. This problem can be viewed as $3!*3!*P_2^1 = 72$ JSPs.

Table II: Problem 5×5 with 22 operations and 4 machines.

Job	O_1	O_2	O_3	O_4	O_5
J_1	$M_1(60)$	$M_2(103)$	$M_3(50)$	$M_3(80)$	$M_4(57)$
J_2	$M_1(63)$	$M_2(84)$	$M_3(194)$	$M_2(243)$	$M_4(84)$
J_3	$M_1(72)$	$M_2(110)$	$M_3(63)$	$M_4(70)$	—
J_4	$M_1(72)$	$M_2(90)$	$M_4(285)$	—	—
J_5	$M_1(72)$	$M_2(97)$	$M_3(168)$	$M_2(140)$	$M_3(105)$

Table III: Flexible operation sections and operations order after IGA scheduling for problem 5×5.

Job	**Flexible operation sections**	**Operations order after IGA scheduling**
J_1	$T_2<2,4>$	1-3-4-2-5
J_2	$T_2<2,4>$	1-4-3-2-5
J_5	$T_2<3,4>$	1-2-4-3-5

4.1.2 Problem 10×10

A middle scale SFJSP instance of 10×10 with 80 operations and 9 machines is depicted in detail in Table IV and Table V. This problem could be viewed as 3×2^{12} JSPs.

Table IV: Problem 10×10 with 80 operations and 9 machines.

Job	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}
J_1	$M_1(63)$	$M_2(105)$	$M_3(50)$	$M_4(63)$	$M_3(57)$	$M_4(50)$	$M_5(42)$	—	—	—
J_2	$M_1(63)$	$M_2(84)$	$M_6(194)$	$M_7(194)$	$M_4(84)$	$M_8(168)$	$M_7(97)$	$M_7(84)$	$M_5(42)$	—
J_3	$M_1(72)$	$M_2(110)$	$M_3(63)$	$M_4(70)$	$M_3(70)$	$M_4(63)$	$M_5(50)$	—	—	—
J_4	$M_1(72)$	$M_2(90)$	$M_6(229)$	$M_7(229)$	$M_4(90)$	$M_8(180)$	$M_7(105)$	$M_7(97)$	$M_5(50)$	—
J_5	$M_1(72)$	$M_2(126)$	$M_3(126)$	$M_4(105)$	$M_3(105)$	$M_4(84)$	$M_5(63)$	—	—	—
J_6	$M_1(72)$	$M_2(97)$	$M_6(168)$	$M_9(126)$	$M_4(74)$	$M_8(168)$	$M_9(84)$	$M_5(63)$	—	—
J_7	$M_1(72)$	$M_2(126)$	$M_3(126)$	$M_4(105)$	$M_3(105)$	$M_4(84)$	$M_5(63)$	—	—	—
J_8	$M_1(72)$	$M_2(97)$	$M_6(168)$	$M_9(126)$	$M_4(74)$	$M_8(168)$	$M_9(84)$	$M_5(63)$	—	—
J_9	$M_1(72)$	$M_2(360)$	$M_6(504)$	$M_4(280)$	$M_2(126)$	$M_8(420)$	$M_4(229)$	$M_5(70)$	—	—
J_{10}	$M_1(72)$	$M_2(90)$	$M_6(229)$	$M_7(229)$	$M_5(50)$	$M_4(90)$	$M_8(180)$	$M_7(105)$	$M_7(97)$	$M_5(50)$

Table V: Flexible operation sections and operations order after IGA scheduling for problem 10×10.

Job	**Flexible operation sections**	**Operations order after IGA**
J_2	$T_2<2,3>$, $T_2<5,6>$	1-3-2-4-6-5-7-8-9
J_4	$T_2<2,3>$, $T_2<5,6>$	1-3-2-4-5-6-7-8-9
J_6	$T_2<3,4>$, $T_2<5,6>$	1-2-3-4-5-6-7-8
J_8	$T_2<3,4>$, $T_2<5,6>$	1-2-4-3-6-5-7-8
J_9	$T_1<<4,6>$, $\{5\}>$	1-2-3-4-6-5-7-8
J_{10}	$T_2<3,4>$, $T_3<T_{31}<<6,9>$, $\{6\}>$, $T_{32}<7,8>>$	1-2-4-3-5-7-8-9-6-10

4.1.3 Problem 15×10

A large scale SFJSP instance of 15×10 with 124 operations and 8 machines is depicted in detail in Table VI and Table VII. This problem can be viewed as 3×2^{24} JSPs.

4.2 Simulation results and analyses

Experiments of the three instances are computed by the IGA algorithm, which was implemented in MATLAB2011R and run on a PC with 2.4 GHz and 2 GB of RAM memory. After several rounds of testing, the probability for the crossover operator and mutation operator are set as $p_c = 0.8$ and $p_m = 0.05$ respectively and the other parameters are set as follows: for problem 5×5, the pop size $Psize = 50$ and the number of iterations $T = 50$; for problem 10×10 and problem 15×10, $Psize = 100$ and $T = 100$. The optimal process plans of the jobs with flexible operations for the three instances are shown in the third column of Table III, Table V and Table VII.

Table VI: Problem 15×10 with 124 operations and 8 machines.

Job	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}
J_1	$M_1(63)$	$M_2(105)$	$M_3(50)$	$M_4(63)$	$M_3(57)$	$M_4(50)$	$M_5(42)$	—	—	—
J_2	$M_1(63)$	$M_2(84)$	$M_6(194)$	$M_7(194)$	$M_4(84)$	$M_8(168)$	$M_7(97)$	$M_7(84)$	$M_5(42)$	—
J_3	$M_1(63)$	$M_2(84)$	$M_6(194)$	$M_6(242)$	$M_4(84)$	$M_8(168)$	$M_8(122)$	$M_7(84)$	$M_5(42)$	—
J_4	$M_1(72)$	$M_2(90)$	$M_6(229)$	$M_7(229)$	$M_5(50)$	$M_4(90)$	$M_8(180)$	$M_7(105)$	$M_7(97)$	$M_5(50)$
J_5	$M_1(72)$	$M_2(110)$	$M_3(63)$	$M_4(70)$	$M_3(70)$	$M_4(63)$	$M_5(50)$	—	—	—
J_6	$M_1(72)$	$M_2(90)$	$M_6(229)$	$M_7(229)$	$M_4(90)$	$M_8(180)$	$M_7(105)$	$M_7(97)$	$M_5(50)$	—
J_7	$M_1(72)$	$M_2(90)$	$M_6(229)$	$M_6(285)$	$M_4(90)$	$M_8(180)$	$M_8(132)$	$M_7(97)$	$M_5(50)$	—
J_8	$M_1(72)$	$M_2(126)$	$M_3(126)$	$M_4(105)$	$M_3(105)$	$M_4(84)$	$M_5(63)$	—	—	—
J_9	$M_1(72)$	$M_2(97)$	$M_6(168)$	$M_6(155)$	$M_4(74)$	$M_8(168)$	$M_8(105)$	$M_5(63)$	—	—
J_{10}	$M_1(72)$	$M_2(97)$	$M_6(168)$	$M_2(140)$	$M_4(74)$	$M_8(168)$	$M_4(93)$	$M_5(63)$	—	—
J_{11}	$M_1(72)$	$M_2(126)$	$M_3(126)$	$M_4(105)$	$M_3(105)$	$M_4(84)$	$M_5(63)$	—	—	—
J_{12}	$M_1(72)$	$M_2(97)$	$M_6(168)$	$M_2(140)$	$M_4(74)$	$M_8(168)$	$M_8(105)$	$M_5(63)$	—	—
J_{13}	$M_1(72)$	$M_2(97)$	$M_6(168)$	$M_6(157)$	$M_4(74)$	$M_8(168)$	M_4 (93)	$M_5(63)$	—	—
J_{14}	$M_1(72)$	$M_2(360)$	$M_6(504)$	$M_4(280)$	$M_2(126)$	$M_8(420)$	$M_4(229)$	$M_5(70)$	—	—
J_{15}	$M_1(72)$	$M_2(90)$	$M_6(229)$	$M_7(229)$	$M_5(50)$	$M_4(90)$	$M_8(180)$	$M_7(105)$	$M_7(97)$	$M_5(50)$

Table VII: Flexible operation sections and operations order after IGA scheduling for problem 15×10.

Job	**Flexible operation sections**	**Operations order after IGA**
J_2	$T_2<2,3>$, $T_2<5,6>$	1-3-2-4-6-5-7-8-9
J_3	$T_2<2,3>$, $T_2<5,6>$	1-3-2-4-5-6-7-8-9
J_4	$T_2<3,4>$, $T_3<T_{31}<<6,9>$, $\{6\}>$, $T_{32}<7,8>>$	1-2-4-3-5-7-6-8-9-10
J_6	$T_2<2,3>$, $T_2<5,6>$	1-2-3-4-5-6-7-8-9
J_7	$T_2<2,3>$, $T_2<5,6>$	1-3-2-4-6-5-7-8-9
J_9	$T_2<3,4>$, $T_2<5,6>$	1-2-4-3-5-6-7-8
J_{10}	$T_2<3,4>$, $T_2<5,6>$	1-2-3-4-5-6-7-8
J_{12}	$T_2<3,4>$, $T_2<5,6>$	1-2-3-4-5-6-7-8
J_{13}	$T_2<3,4>$, $T_2<5,6>$	1-2-4-3-6-5-7-8
J_{14}	$T_1<<4,6>$, $\{5\}>$	1-2-3-4-6-5-7-8
J_{15}	$T_2<3,4>$, $T_3<T_{31}<<6,9>$, $\{6\}>$, $T_{32}<7,8>>$	1-2-4-3-5-6-7-8-9-10

By the way, when neglecting the sequence flexibility, the SFJSP can be viewed as a classical JSP. It is obvious that the classical JSP can also be solved by IGA. This is because that the first segment of the chromosomes is empty and IGA becomes a complete unification algorithm with the classical GA algorithm for JSP [30, 31]. Therefore, through comparing with scheduling results for SFJSP's corresponding JSP, the performance of IGA algorithm for SFJSP can be obtained.

The comparison results are shown in Table VIII. The four rows signify the average computational time, the best makespan, the mean value of makespan and standard deviation of makespan obtained by twenty experiments. For each problem, the best makespan of SFJSP

solved by IGA is shorter than the corresponding JSP result, as shown in Fig. 2. Take problem 10×10 for example, the best makespan of SFJSP solved by IGA is 2461, which is 115 shorter than the JSP result 2576. But the computational time of IGA solving SFJSP is only about 1s longer than the time of IGA solving JSP. The comparison results show that taking the sequence flexibility of process plan into consideration, a more optimal scheduling result will be obtained only at small computational time cost. Both of the standard deviations of the two problems are smaller than 8.1, which can prove the validity and stability of IGA.

Table VIII: Comparisons of the results.

	Problem 5×5		**Problem 10×10**		**Problem 15×10**	
	SFJSP	JSP	SFJSP	JSP	SFJSP	JSP
AV(CPUS)	3.455	3.268	8.679	7.868	16.679	14.868
Best makespan	987	1011	2461	2576	3920	4111
Mean value	987	1011	2467.2	2581.2	3933.3	4121.7
Standard deviation	0	0	7.846	7.665	8.067	6.035

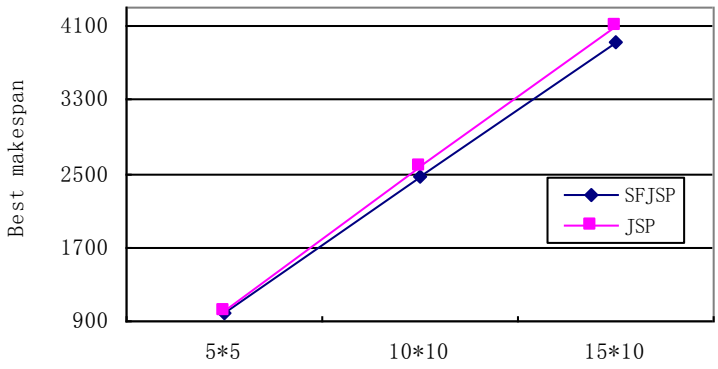

Figure 2: The comparisons of best makespan between SFJSP and JSP.

5. CONCLUSIONS AND FUTURE STUDY

This paper proposes an improved genetic algorithm to solve the job shop scheduling problem considering the operation sequence flexibility (SFJSP) which is more practical to some extent but has received less attention until now. The SFJSP includes a routing problem to select the optimal and suboptimal process sequence of flexible operations of each job and a sequencing problem to order operations on each machine. The proposed IGA is capable of solving the two problems of SFJSP simultaneously by improved chromosome encoding schema, crossover operator and mutation operator. The performance of IGA was firstly proved by a Simulation experiment with a benchmark JSP problem. Then, computational experiments on the realistic problems of a bearing enterprise showed that taking the sequence flexibility into consideration while scheduling, a more satisfying scheduling result could be achieved, and IGA could efficiently find a good approximate solution to SFJSP in a reasonable amount of computation time.

Future work will propose a number of heuristic approaches for improving the ability of local search of the genetic algorithm. In addition, in most job-shop scheduling problems other constraints exist, such as operation flexibility and processing flexibility. Hybrid algorithm based on genetic algorithm and heuristic algorithm might be developed to solve JSP with more flexibility.

6. ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation Monumental Project of China under Award Number 61034003, the National Science and Technology Plan of China under Award Number 2013BAF02B03 and the National Natural Science Foundation of China under Award Number 70772086.

REFERENCES

[1] Peng, S. O.; Stephen, F. S. (1988). Viewing scheduling as an opportunistic problem-solving process, *Annals of Operation Research*, Vol. 12, No. 1, 85-108, doi:10.1007/BF02186362

[2] Garey, M. R.; Johnson, D. S.; Sethi, R. (1976). The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, Vol. 1, No. 2, 117-129, doi:10.1287/moor.1.2.117

[3] Jain, A. S.; Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future, *European Journal of Operational Research*, Vol. 113, No. 2, 390-434, doi:10.1016/S0377-2217(98)00113-1

[4] Saygin, C.; Kilic, S. E. (1999). Integrating flexible process plans with scheduling in flexible manufacturing systems, *International Journal of Advanced Manufacturing Technology*, Vol. 15, No.4, 268-280, doi:10.1007/s001700050066

[5] Pezzella, F.; Morganti, G.; Ciaschetti, G. (2007). A genetic algorithm for the flexible job-shop scheduling problem, *Computers & Operations Research*, Vol. 35, No. 10, 3202-3212, doi:10.1016/j.cor.2007.02.014

[6] Brucker, P.; Schlie, R. (1990). Job-shop scheduling with multi-purpose machines, *Computing*, Vol. 45, No. 4, 369-375, doi:10.1007/BF02238804

[7] Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, Vol. 41, No. 3, 157-183, doi:10.1007/BF02023073

[8] Dauzere-Peres, S.; Roux, W.; Lasserre, J. B. (1998). Multi-resource shop scheduling with resource flexibility, *European Journal of Operational Research*, Vol. 107, No. 2, 289-305, doi:10.1016/S0377-2217(97)00341-X

[9] Scrich, C. R.; Armentano, V. A.; Laguna, M. (2004). Tardiness minimization in a flexible job shop: A tabu search approach, *Journal of Intelligent Manufacturing*, Vol. 15, No. 1, 103-115, doi:10.1023/B:JIMS.0000010078.30713.e9

[10] Saidi-Mehrabad, M.; Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms, *International Journal of Advanced Manufacturing Technology*, Vol. 32, No. 5-6, 563-570, doi:10.1007/s00170-005-0375-4

[11] Low, C.; Wu, T.-H. (2001). Mathematical modelling and heuristic approaches to operation scheduling problems in an FMS environment, *International Journal of Production Research*, Vol. 39, No. 4, 689-708, doi:10.1080/00207540150504403

[12] Loukil, T.; Teghem, J.; Fortemps, P. (2007). A multi-objective production scheduling case study solved by simulated annealing, *European Journal of Operational Research*, Vol. 179, No. 3, 709-722, doi:10.1016/j.ejor.2005.03.073

[13] Chen, H.; Ihow, J.; Lehmann, C. (1999). A genetic algorithm for flexible job-shop scheduling, *Proceedings of the IEEE International Conference on Robotics and Automation*, 1120-1125

[14] Kacem I. (2003). Genetic algorithm for the flexible job-shop scheduling problem, *IEEE International Conference on Systems*, 3464-3469

[15] Zhang, G.; Gao, L.; Li, P.; Zhang, C. (2009). Improved genetic algorithm for the flexible job-shop scheduling problem, *Chinese Journal of Mechanical Engineering*, Vol. 45, No. 7, 145-151, doi:10.3901/JME.2009.07.145

[16] Lestan, Z.; Brezocnik, M.; Buchmeister, B.; Brezovnik, S.; Balic, J. (2009). Solving the job-shop scheduling problem with a simple genetic algorithm, *International Journal of Simulation Modelling*, Vol. 8, No. 4, 197-205, doi:10.2507/IJSIMM08(4)2.138

[17] De Giovanni, L.; Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, *European Journal of Operational Research*, Vol. 200, No. 2, 395-408, doi:10.1016/j.ejor.2009.01.008

[18] Zhang, G.; Gao, L.; Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem, *Expert Systems with Applications*, Vol. 38, No. 4, 3563-3573, doi:10.1016/j.eswa.2010.08.145
[19] Low, C.; Yip, Y.; Wu, T.-H. (2006). Modelling and heuristics of FMS scheduling with multiple objectives, *Computers & Operations Research*, Vol. 33, No. 3, 674-694, doi:10.1016/j.cor.2004.07.013
[20] Fattahi, P.; Saidi Mehrabad, M.; Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, *Journal of Intelligent Manufacturing*, Vol. 18, No. 3, 331-342, doi:10.1007/s10845-007-0026-8
[21] Gao, J.; Sun, L.; Gen, M. (2008). A hybrid genetic and variable neighbourhood descent algorithm for flexible job shop scheduling problems, *Computers & Operations Research*, Vol. 35, No. 9, 2892-2907, doi:10.1016/j.cor.2007.01.001
[22] Vilcot, G.; Billaut, J.-C. (2008). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem, *European Journal of Operational Research*, Vol. 190, No. 2, 398-411, doi:10.1016/j.ejor.2007.06.039
[23] Hutchinson, G. K.; Pflughoeft, K. A. (1994). Flexible process plans: their value in flexible automation systems, *International Journal of Production Research*, Vol. 32, No. 3, 707-719, doi:10.1080/00207549408956962
[24] Brandimarte, P. (1999). Exploiting process plan flexibility in production scheduling: A multi-objective approach, *European Journal of Operational Research*, Vol. 114, No. 1, 59-71, doi:10.1016/S0377-2217(98)00029-0
[25] Saygin, C.; Chen, F. F.; Singh, J. (2001). Real-time manipulation of alternative routings in flexible manufacturing systems: A simulation study, *International Journal of Advanced Manufacturing Technology*, Vol. 18, No. 10, 755-763, doi:10.1007/s001700170019
[26] Jain, A.; Jain, P. K.; Singh. I. P. (2006). An integrated scheme for process planning and scheduling in FMS, *International Journal of Advanced Manufacturing Technology*, Vol. 30, No. 11-12, 1111-1118, doi:10.1007/s00170-005-0142-6
[27] Li, X. Y.; Shao, X. Y.; Gao, L. (2008). Optimization of flexible process planning by genetic programming, *International Journal of Advanced Manufacturing Technology*, Vol. 38, No. 1-2, 143-153, doi:10.1007/s00170-007-1069-x
[28] Shao, X.; Li, X.; Gao, L.; Zhang, C. (2009). Integration of process planning and scheduling – A modified genetic algorithm-based approach, *Computer & Operations Research*, Vol. 36, No. 6, 2082-2096, doi:10.1016/j.cor.2008.07.006
[29] Ozguven, C.; Ozbakir, L.; Yavuz, Y. (2010). Mathematical models for job-shop scheduling problem with routing and process plan flexibility, *Applied Mathematical Modelling*, Vol. 34, No. 6, 1539-1548, doi:10.1016/j.apm.2009.09.002
[30] Gu, F.; Chen, H.-P.; Lu, B. Y. (2006). The solution for multi-objective flexible job shop scheduling based on genetic algorithm, *Operations Research and Management Science*, Vol. 15, No. 1, 134-139
[31] Amirthagadeswaran, K. S.; Arunachalam,V. P. (2006). Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction, *International Journal of Advanced Manufacturing Technology*, Vol. 28, No. 5-6, 532-540, doi:10.1007/s00170-004-2403-1