Int J Adv Manuf Technol (2012) 60:291–301
DOI 10.1007/s00170-011-3600-3

ORIGINAL ARTICLE

Interval job shop scheduling problems

Deming Lei

Received: 14 April 2010 / Accepted: 16 August 2011 / Published online: 8 September 2011

Abstract Interval number theory has been applied to many fields; however, its applications to production scheduling are seldom investigated. In this paper, interval theory is used for its low cost in uncertainty modeling and novel interval job shop scheduling problem is proposed. To build the schedule of the problem, the addition and comparison of two interval numbers are first introduced and then a decoding procedure is constructed by using the chromosome of operation-based representation. It is proved that the possible actual objective values are contained in interval objective. An effective genetic algorithm (GA) is presented and tested by using some randomly generated instances. Computational results show the effectiveness of the GA.

Keywords Genetic algorithm · Interval number · Job shop scheduling · Operation-based representation

1 Introduction

Uncertainty is the basic feature of manufacturing process and often described by using fuzzy theory and stochastic theory. In the past decade, fuzzy scheduling and stochastic scheduling have been attracted much attention and many results have been obtained. For fuzzy job shop scheduling problem (FJSSP), Sakawa and Mori [1] presented an efficient genetic algorithm (GA) by incorporating the concept of similarity among individuals. Sakawa and Kubota [2] presented a GA for FJSSP with multiple objectives. Li et al. [3] proposed a GA by adopting two-chromosome presentation and the extended version of Giffler-Thompson procedure for FJSSP with alternative machines. Song et al. [4] presented a combined strategy of GA and ant colony optimization. They also designed a new neighborhood search method and an improved tabu search to intensify the local search ability of the hybrid algorithm. Niu et al. [5] proposed a particle swarm optimization with genetic operators (GPSO) to minimize fuzzy makespan. Lei [6] proposed an efficient Pareto archive particle swarm optimization for the problem with three objectives for obtaining a set of Pareto optimal solutions. Petrovic et al. [7] developed a fuzzy rule-based system to determine the size of lots. Lei [8] developed an efficient decomposition-integration genetic algorithm for FJSSP with processing flexibility. Lei [9] proposed a random key genetic algorithm for the problem. Hu et al. [10] presented a modified differential evolution algorithm for FJSSP with new objective function.

For stochastic job shop scheduling problem, Tavakkoli-Moghaddam et al. [11] proposed a hybrid method based on neural network and simulated annealing (SA), which uses a neural network approach to generate an initial feasible solution and a SA to improve the quality of the initial solution. Lei [12] proposed a RKGA for the JSSP subject to random breakdown. Gu et al. [13, 14] presented two quantum genetic algorithms for the expected value of makespan.

Fuzzy and stochastic theories are commonly used to model the uncertainty of processing conditions. Probability distribution and membership function are needed to be known in advance and a number of data are required to build fuzzy membership function or decide the distribution of stochastic variable. In the actual manufacturing systems, it is expensive or impossible to

D. Lei (✉)
School of Automation, Wuhan University of Technology,
122 Luoshi Road,
Wuhan, Hubei Province, People's Republic of China
e-mail: deminglei11@163.com

produce many processing data for membership function and probability distribution.

Interval number theory is a novel tool and has been applied to describe uncertainty. Obtaining the lower and upper bound of interval is notably simpler than building membership function or distribution function, so the usage of interval has some advantages: the lower and upper bound of interval are only required to indicate uncertain conditions; decision-maker prefers using interval number to describe his expected objective and the obtained interval results can be understood easily.

In this paper, we apply interval number theory to production scheduling and propose a novel scheduling problems named interval job shop scheduling problems (IJSSP). To solve IJSSP with objective of total tardiness, an operation-based representation is first considered and an effective decoding procedure is then built, which can guarantee the inclusion of all possible actual objective values in interval objective, finally, a GA is proposed and applied to some randomly produced instances.

The remainder of the paper is organized as follows. Basic concepts of interval theory are introduced in Section 2 and problem formulation is done in Section 3. Section 4 describes the proposed GA for IJSSP. Numerical test experiments on the proposed algorithm are reported in section 5 and the conclusions are summarized in the final section.

2 Basic concepts of interval theory

Interval number theory was pioneered by Moore [15] as a tool for bounding and rounding errors in computer programs. Since then, interval number theory had been developed into a general methodology for investigating numerical uncertainty in numerous fields such as data mining [16], automatic control [17], and optimization [18].

Definition 1 A interval is a set of the form $A = [\underline{A}, \overline{A}]$, where $\underline{A}$ and $\overline{A}$ are the left and right limit of the interval A, respectively. If $\underline{A} = \overline{A}$, A is a real number.

For interval $A = [\underline{A}, \overline{A}]$ and $B = [\underline{B}, \overline{B}]$, the addition of A and B is defined as follows.

$$A + B = [\underline{A} + \underline{B}, \overline{A} + \overline{B}] \tag{1}$$

The comparison of any two interval numbers is importation to the application of interval theory to real problems including production scheduling. Sengupta and Pal [19] presented a brief survey of the existing works on comparing and ranking interval numbers. For example, Ishibuchi and Tanaka [20] defined the following method to rank interval numbers.

$$A \leq_{\mathrm{LR}} B \text{ iff } \underline{A} \leq \underline{B} \text{ and } \overline{A} \leq \overline{B} \tag{2}$$

$$A <_{\mathrm{LR}} B \text{ iff } A \leq_{\mathrm{LR}} B \text{ and } A \neq B \tag{3}$$

They suggested another order relation $\leq_{\mathrm{mw}}$ where $\leq_{\mathrm{LR}}$ cannot be applied,

$$A \leq_{\mathrm{mw}} B \text{ iff } m(A) \leq m(B) \text{ and } w(A) \geq w(B) \tag{4}$$

$$A <_{\mathrm{mw}} B \text{ iff } A \leq_{\mathrm{mw}} B \text{ and } A \neq B \tag{5}$$

$$P(A \leq B) = \begin{cases} 0 & \underline{A} \geq \overline{B} \\ 0.5 \cdot \dfrac{\overline{B} - \underline{A}}{\overline{A} - \underline{A}} \cdot \dfrac{\overline{B} - \underline{A}}{\overline{B} - \underline{B}} & \underline{B} \leq \underline{A} < \overline{B} \leq \overline{A} \\ \dfrac{\underline{B} - \underline{A}}{\overline{A} - \underline{A}} + 0.5 \cdot \dfrac{\overline{B} - \underline{B}}{\overline{A} - \underline{A}} & \underline{A} < \underline{B} < \overline{B} \leq \overline{A} \\ \dfrac{\underline{B} - \underline{A}}{\overline{A} - \underline{A}} + \dfrac{\overline{A} - \underline{B}}{\overline{A} - \underline{A}} \cdot \dfrac{\overline{B} - \overline{A}}{\overline{B} - \underline{B}} + 0.5 \cdot \dfrac{\overline{A} - \underline{B}}{\overline{A} - \underline{A}} \cdot \dfrac{\overline{A} - \underline{B}}{\overline{B} - \underline{B}} & \underline{A} < \underline{B} \leq \overline{A} < \overline{B} \\ \dfrac{\overline{B} - \overline{A}}{\overline{B} - \underline{B}} + 0.5 \cdot \dfrac{\overline{A} - \underline{A}}{\overline{B} - \underline{B}} & \underline{B} \leq \underline{A} < \overline{A} < \overline{B} \\ 1 & \overline{A} \leq \underline{B} \end{cases} \tag{6}$$

The possibility degree of interval number denotes certain degree that one interval number is larger or smaller than another. Jiang et al. [18] proposed a possibility degree-based ranking method, which is shown in Eq. 6.

The above possibility degree-based methods have some features.

(1) $0 \leq P(A \leq B) \leq 1$
(2) If $P(A \leq B) = \alpha$, then $P(B \leq A) = 1 - \alpha$
(3) If $P(A \leq B) = P(B \leq A)$, then $A=B$

If $A \equiv B$, which means $\underline{A} = \underline{B}$, $\overline{A} = \overline{B}$, then $P(A \le B) = P(B \le A) = 0.5$, so we define $A <_{pd} B$ iff $P(A \le B) > 0.5$ or $P(B \le A) < 0.5$; $A <_{pd} B$ iff $P(A \le B) < 0.5$ or $P(B \le A) > 0.5$; $A <_{pd} B$ iff $P(A \le B) = 0.5$.

For $A = [\underline{A}, \overline{A}]$ and $B = [\underline{B}, \overline{B}]$, $A \vee B = [\max\{\underline{A}, \underline{B}\}, \max\{\overline{A}, \overline{B}\}]$, so $A \vee B$ is still interval number.

3 Problem descriptions

IJSSP is the extended version of JSSP by introducing interval processing conditions including interval processing times. $n \times m$ IJSSP is composed of n jobs $J_i (i = 1, 2, \cdots, n)$ and m machines $M_k (k = 1, 2, \cdots, m)$. Each job consists of m operations. Operation 0_{ij} indicates the jth operation of J_i processed on a machine and its processing time is represented as interval number $p_{ij} = [\underline{p}_{ij}, \overline{p}_{ij}]$. The due date of job J_i is indicated as $d_i = [\underline{d}_i, \overline{d}_i]$, where $\underline{d}_i$ is the maximum time meeting customer's requirement on delivery and $\overline{d}_i$ is the minimum unacceptable time of delivery, that is, if the delivery is done after time $\overline{d}_i$, customers may refuse to accept the delivery.

Other constraints of JSSP are still suitable to IJSSP, such as:

Each machine can process at most one operation at a time
No jobs may be processed on more than one machine at a time
Operation cannot be interrupted
Operations of a given job have to be processed in a given order and setup times and remove times are included in the processing times et al. In this paper, total tardiness is considered.

$$\sum T = \sum_{i=1}^{n} T_i \tag{8}$$

where $T_i = [\max\{0, \underline{C}_i - \overline{d}_i\}, \max\{0, \overline{C}_i - \underline{d}_i\}]$ and $C_i = [\underline{C}_i, \overline{C}_i]$ are the tardiness and interval completion time of job J_i, respectively.

For job J_i, if $\max\{0, \underline{C}_i - \overline{d}_i\} > 0$, then the delivery of J_i has to delay under any possible actual processing time. If $\max\{0, \overline{C}_i - \underline{d}_i\} = 0$, job J_i always can be delivered before due date. The bigger $\max\{0, \overline{C}_i - \underline{d}_i\}$ is, the higher the possibility of delay in delivery is and the bigger the tardiness is. Table 1 describes an example of IJSSP with four jobs and three machines.

4 GA for IJSSP

4.1 The construction of interval schedule

Operation-based representation has been extensively applied to JSSP [21] and FJSSP [3, 5]. This representation also can be used in IJSSP. For $n \times m$ IJSSP, the chromosome of operation-based representation is an integer string $(p_1, p_2, \cdots, p_{mn})$, in which serial number of each job occurs m times. To build the schedule of scheduling problems, the chromosome is first translated into an ordered operation list and then each operation in the list is allocated sequentially a best available beginning time by adopting some operations of interval numbers shown in section 2.

4.1.1 Raw interval schedule

Let $\sigma_{ij} = [\underline{\sigma}_{ij}, \overline{\sigma}_{ij}]$, $\tau_{ij} = [\underline{\tau}_{ij}, \overline{\tau}_{ij}]$, and $\eta_k = [\underline{\eta}_k, \overline{\eta}_k]$ indicate, respectively, the beginning time, the completion time of 0_{ij}, and the available time of machine M_k. Firstly, We build a raw interval schedule using the following principles: each operation may be inserted into the idle time interval of machines; for 0_{ij} $(j>1)$ processed on machine M_k, if $\tau_{i(j-1)} <_{pd} \eta_k$, $\sigma_{ij} = \eta_k$; else $\sigma_{ij} = \tau_{i(j-1)}$, for 0_{i1} on machine M_k, $\sigma_{ij} = \eta_k$.

For the example shown in Table 1, the chromosome may be (1,3,2,3,3,4,1,2,4,1,2,4) and the corresponding list of the ordered operations is o_{11}, o_{31}, o_{32}, o_{33}, o_{12}, o_{22}, o_{41}, o_{42}, o_{13}, o_{23}, o_{43}. The raw interval schedule is built in the following way.

The operation o_{11} is first allocated, σ_{11}=[0,0], $\tau_{11} = \sigma_{11} + p_{11}$=[1,3], η_1=[1,3], the next is o_{31}, σ_{31}=[0,0], $\tau_{31} = \sigma_{31} + p_{31}$=[2,5], η_3=[2,5]. For operation o_{21}, its beginning time

Table 1 An illustration of IJSSP

Job	Operations											
	1	2	3	1	2	3	1	2	3	1	2	3
	Interval processing time			Processing routes			Actual processing time			Actual processing time		
1	[1,3]	[3,5]	[2,3]	M_1	M_2	M_3	3	4	3	3	4	3
2	[2,4]	[4,6]	[3,5]	M_1	M_3	M_2	4	4	4.5	3	3	4
3	[2,5]	[3,7]	[4,6]	M_3	M_1	M_2	4	7	6	4	7	5
4	[5,6]	[3,5]	[2,4]	M_3	M_2	M_1	6	5	2.5	5	4	4

σ_{21}=[1,3]and the completion time τ_{21}=[3,7], η_1=[3,7]. The operation $o_{32}$is considered, σ_{32}=[3,7], τ_{32}=[6,14].

For operation o_{33}, σ_{33}=[6,14] and τ_{32}=[10,20]. For operation o_{12}, if σ_{12}=[1,3], then τ_{12}=[4,8], if the ranking method of *Jiang* et al. [18] is used, $P([4,8] \leq [6,14]) = 0.833$, τ_{12}=[4,8]$<_{pd}$ σ_{33}, o_{12}can allocated in the idle time interval before the beginning time of o_{33}. Operation $o_{22}$is then considered, σ_{22}=[3,7], τ_{22}=[7,13]. The next is operation o_{41}, σ_{41}=[7,13], τ_{41}=[12,19]. For operation o_{42}, its processing machine is M_2, η_2=[10,20] is compared with τ_{41}, $P([10,20] \leq [12,19]) = 0.529$, $\eta_2<_{\text{pd}}$ τ_{41}, so the beginning time of o_{42} is $[12,19] \subset [10,20]$ and τ_{42}=[15,24].

For operation o_{13}, τ_{12}=[4,8] is compared with the machine available time η_3=[12,19], $\tau_{12}<_{\text{pd}}\eta_3$, so $\sigma_{13}=\eta_3$ and τ_{13}=[14,22]. For operation o_{23}, τ_{22}=[7,13]$<_{\text{pd}}\eta_2$= [15,24] according to Eq. (7), so σ_{23}=[15,24] and τ_{23}= [18,29]. The final operation is o_{43}, σ_{43}=[15,24] for $\tau_{42}>_{\text{pd}}$ η_1=[6,14] and τ_{43}=[17,28]. Figure 1 shows the constructing procedure of raw interval schedule, in which the isosceles triangle under the line represents the beginning time of operation and the isosceles triangle on the line indicates the completion time of operation.

Definition 2 $f(x) = f(x_1, x_2, \cdots, x_n)$ is the real valued function of real variables $x_1, x_2, \cdots, x_n$, where $x = (x_1, x_2, \cdots, x_n)$. If all real variables are replaced with interval number variables $X_1, X_2, \cdots, X_n$, $x_1 \quad X_1$, $x_2 \quad X_2$, $\cdots$, $x_n \in X_n$, and four arithmetic operations of interval number variables substitute for corresponding operations of real number, then $F(X) = f(X_1, X_2, \cdots, X_n)$ has the monotony of inclusion relation, that is, if $X \subset Y$, then $F(X) \subset F(Y)$. Function F is called the natural interval extension of f.

Theorem 1 If function F is the natural interval extension of f and has the monotony of inclusion relation, then $\widehat{f}(X_1, X_2, \cdots, X_n) \subseteq F(X_1, X_2, \cdots, X_n)$. *where* $\widehat{f}(X_1, X_2, \cdots, X_n) = \{f(x_1, x_2, \cdots, x_n) | \forall x_i \in X_i, i = 1, 2, \cdots, n\}$.

Once the raw interval schedule is built, for each job J_i, its raw interval completion time $\widetilde{C}_i = \sum_{o_{ij} \in \theta_i} p_{ij}$, θ_i is the set of

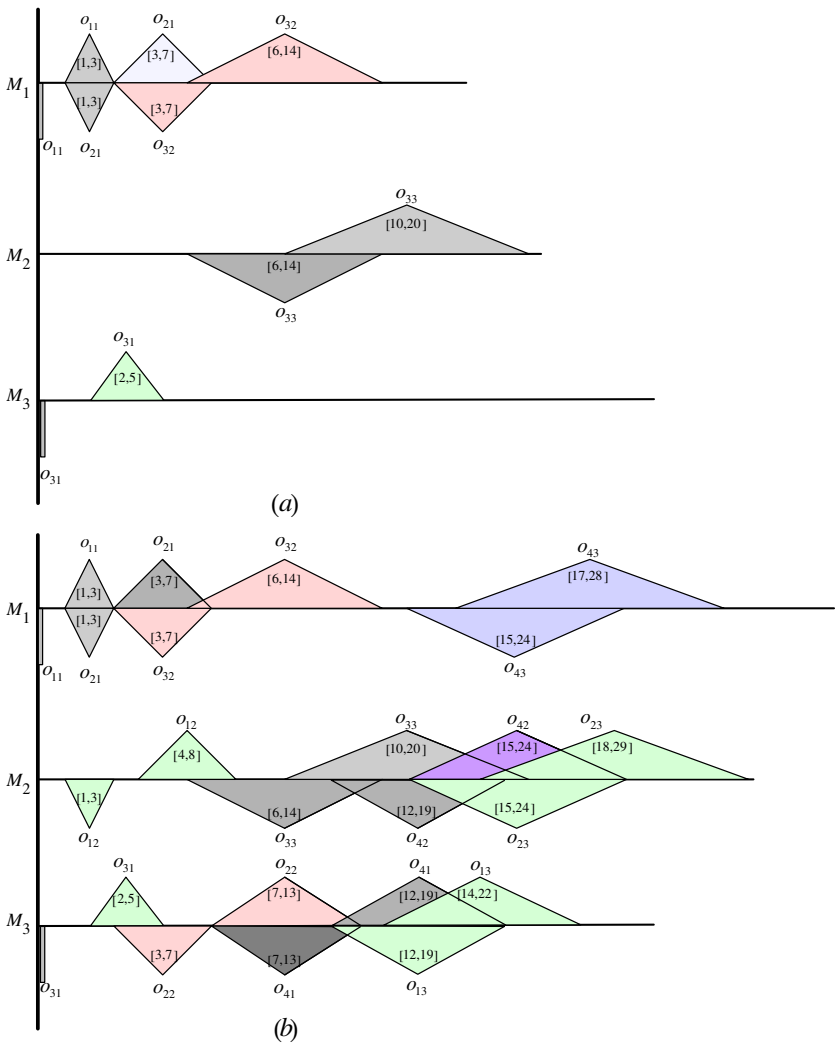

Fig. 1 The construction procedure of raw interval schedule

some operations for computing $\widetilde{C}_i$. For a set of actual processing time $\widehat{p}_{ij}$ of any operation o_{ij}, $\widehat{p}_{ij} \in p_{ij}$, the actual raw completion time of $\widehat{C}_i = \sum_{o_{lj} \in \theta_i} \widehat{p}_{lj}$. It can be proved that $\widetilde{C}_i$ is the natural interval extension of $\widehat{C}_i$ under the same operation sequence. So for any possible value of $\widehat{p}_{ij}$, $\widehat{C}_i = \sum_{o_{lj} \in \theta_i} \widehat{p}_{lj} \in \widetilde{C}_i$.

4.1.2 The computation of interval completion time

After the raw interval schedule is constructed, we do some adjustments for each operation o_{ij} meeting the conditions: $\sigma_{lj} \subset \eta_k$ or $\eta_k \subset \sigma_{lj}$, and $\sigma_{1j} > \eta_k$ in the following way: compute the increment $\Delta\underline{\sigma}_{lj} = \max\left\{\underline{\sigma}_{lj}, \underline{\eta}_k\right\} - \underline{\sigma}_{lj}$, $\Delta\overline{\sigma}_{lj} = \max\left\{\overline{\sigma}_{lj}, \overline{\eta}_k\right\} - \overline{\sigma}_{lj}$, calculate the new lower and upper limit of σ_{1j}, which are $\Delta\underline{\sigma}_{ij} + \underline{\sigma}_{ij}$ and $\Delta\overline{\sigma}_{ij} + \overline{\sigma}_{ij}$, by adding the increment, and adjust the interval beginning time and completion time of all related operations, as a result, the lower and upper limit of $\widetilde{C}_i$ increases only if the related $\Delta\underline{\sigma}_{lj}$ and $\Delta\overline{\sigma}_{lj}$ are greater than 0. Finally, the interval completion time is obtained for each job J_i, $i = 1, 2, \cdots, n$. Figure 2 shows the final interval schedule.

When a set of actual processing times are considered, we first calculate the raw completion time $\widehat{C}_i$, and then add the increment of actual beginning time and adjust the beginning time and completion time of all related operations. The actual completion time can be obtained finally. For any possible value of $\widehat{p}_{ij}$, $\widehat{C}_i \in \widetilde{C}_i$, the increment of the actual beginning time is always equal to one of two increments $\Delta\underline{\sigma}_{lj}$ and $\Delta\overline{\sigma}_{lj}$. It can be concluded that any possible actual completion time always lies in interval completion time. This feature can guarantee that interval tardiness of each job includes all possible actual tardiness.

For any operations, $\max\left\{\underline{\sigma}_{ij}, \underline{\eta}_k\right\} = \Delta\underline{\sigma}_{ij} + \underline{\sigma}_{ij}$ and $\max\left\{\overline{\sigma}_{ij}, \overline{\eta}_k\right\} = \Delta\overline{\sigma}_{ij} + \overline{\sigma}_{ij}$. We can directly obtain the final interval procedure in Fig. 2 by using the equation $\underline{\sigma}_{ij} = \max\left\{\underline{\tau}_{i(j-1)}, \underline{\eta}_k\right\}$ and $\overline{\sigma}_{ij} = \max\left\{\overline{\tau}_{i(j-1)}, \overline{\eta}_k\right\}$, that is, $\sigma_{ij} = \tau_{i(j-1)} \vee \eta_k$ for operation o_{ij} (j>1) processed on machine M_k.

As shown in Figs. 1 and 2, $\widetilde{C}_1 = [14, 22]$, $\widetilde{C}_2 = [18, 29]$, $\widetilde{C}_3 = [10, 20]$, $\widetilde{C}_4 = [17, 28]$. Operation o_{42} meets the conditions: $\sigma_{42} \subset \eta_2$ and $\sigma_{42} > \eta_2$, *so* $\Delta\underline{\sigma}_{42} = 0$, $\Delta\overline{\sigma}_{42} = 1$, $\sigma_{42} = \tau_{41} \vee \eta_2 = [12, 20]$, with the variation of σ_{42}, τ_{42} is equal to [15,25] and the interval beginning time and completion time of o_{23}, o_{43} are adjusted, σ_{23}=[15,25], τ_{23}=[18,30], σ_{43}=[15,25], τ_{43}=[17,28]. Finally, C_1=[14,22], C_2=[18,30], C_3=[10,20], and C_4=[17,29]. If d_1=[20,25], d_2=[21,24], d_3=[17,22], and d_4=[15,20], then T_1=[0,2], T_2=[0,9], T_3=[0,3], and T_4=[2,14]. Σ^T=[2,28].

In Table 1, two sets of possible $\widehat{p}_{ij}$ are considered, the first set is composed of data in the first bracket of data grid and the second consists of all data in the second bracket. Figure 3 shows two possible actual schedules using the data in Table 1. It can be found that the possible actual completion time of each operation is contained in its interval completion time. Similarly, all possible values of actual total tardiness are located in the interval total tardiness, so we can compare different solutions in terms of their interval total tardiness. The smaller the interval total tardiness is, the better the solution is.

4.2 Algorithm description

An efficient GA is proposed based on the above procedure to build interval schedule. The GA produces new solutions by using tournament selection, generalized order crossover (GOX) and swap mutation. In the

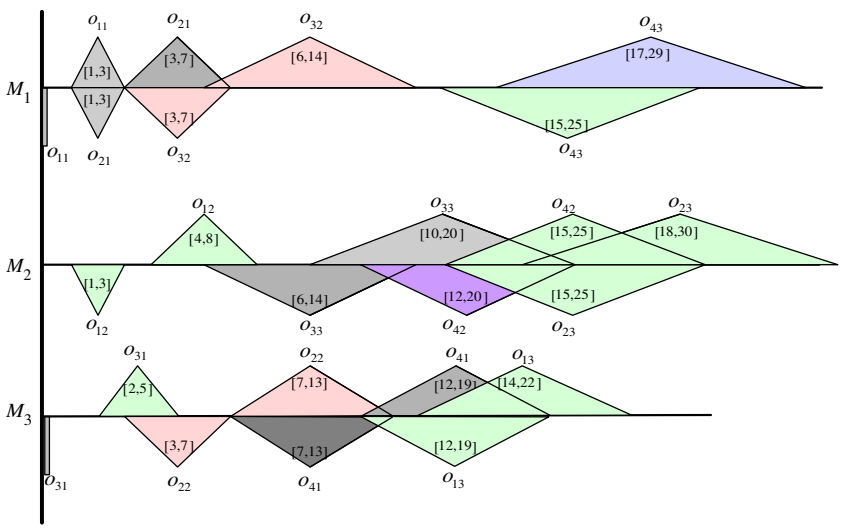

Fig. 2 The final interval schedule

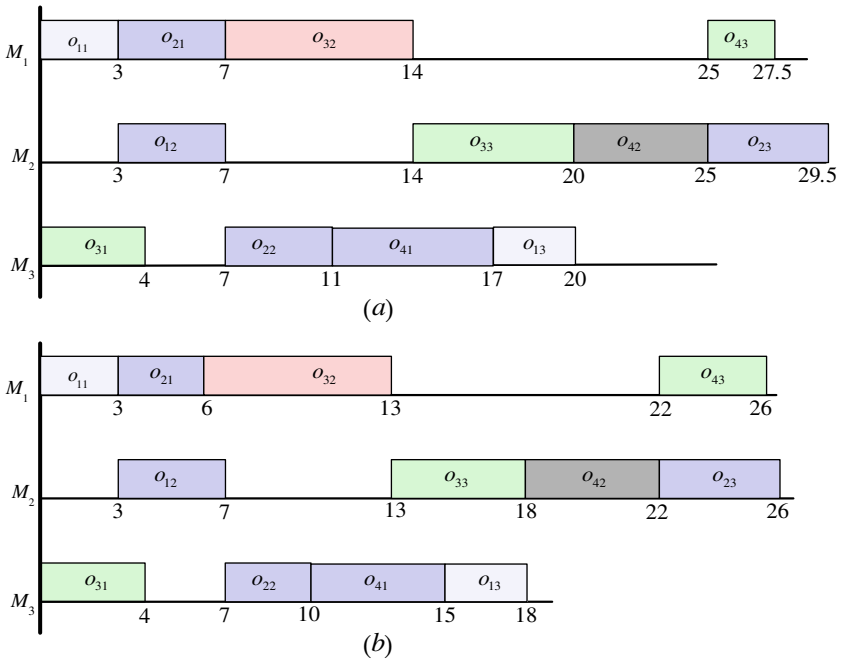

Fig. 3 The actual schedules obtained by using two sets of actual processing time

whole process of the GA, no illegal interval schedules occur and any possible values of actual objective are always contained in the corresponding interval objective.

The GA is described as follows.

1. Randomly generate an initial population P with N individuals and determine the solution with the best fitness as elite individual
2. Perform binary tournament selection on P and obtain a new population P
3. Perform GOX and swap on population P
4. Calculate fitness of each individual and maintain the elite individual if possible;
5. If the predetermined number of generations is met, terminates its search; else, go to step 2.

For individual i, its fitness function $\text{fit}(i) = \sum T^i$. where $\sum T^i$ is the interval total tardiness of individual i . The usual elite strategy is used, in which the optimal solution produced by the GA is stored as an elite individual; moreover, the elite individual is always added into population before reproduction.

Roulette wheel reproduction and breeding pool reproduction cannot be applied for the interval feature of objective, so tournament selection [22] is used and described in the following way: first two individuals are randomly selected from the population, and then an individual is chosen if the individual has smaller fitness than the other individual. Finally, the selected individuals go back to the population and can be chosen again.

GOX [23] is described as follows: first randomly select a substring ℓ of the first parent, determine the position of the first element of ℓ on the second parent and remove the substring ℓ from the second parent. By inserting ℓ into the position of its first element, the offspring is obtained. The swap mutation is adopted and described as follows: randomly select two genes from the chosen individual, and then exchange them.

5 Computational experiments

In this section, the impact of interval ranking methods on interval schedule is first discussed and then the GA is compared with GPSO and SA by using seven random generated instances. The instances 1, 2, 3, and 4 are composed of 10 jobs and 10 machines and each of the other instances has 15 jobs and 10 machines. The processing data are shown in Appendix. All experiments are implemented by using Microsoft Visual C++ 6.0 and run on PC with 1 G RAM 2.5 G CPU.

5.1 Analyses on interval ranking methods

In the decoding procedure, some operations may be inserted into the idle time of machines, and the number of these insertable operations may be different for the same chromosome in the deterministic case and interval case. A set of actual processing time are randomly chosen from the interval time. For each instance, 300 initial populations are produced and each individual of these populations are

decoded in the deterministic case and interval case, respectively. Det indicates the results obtained in the deterministic case and Int denotes the results of interval case. Table 2 shows the results of the number of the insertable operations in two cases.

$$insnum_i = \frac{1}{N}\sum_{j=1}^{N} opr_{ij},$$

$$avgnum = \frac{1}{300}\sum_{i=1}^{300} ins\_num_i,$$

$$\min num = \min\{insnum_i | i = 1, 2, \cdots 300\},$$

$$\max num = \max\{insnum_i | i = 1, 2, \cdots 300\},$$

where opr_{ij} indicates the number of the insertable operations of the *j*th individual in the *i*th initial population, N is *size* of initial population, N=100, and $insnum_i$ is the average number of the insert-able operation in the *i*th initial population.

As shown in Table 2, there are no significant differences between the results obtained in two cases. The values of avgnum of interval case are nearly equal to those of the deterministic case for all instances. The results of minnum of the deterministic case are close to those of interval case for all instances. The number of the insertable operations does not increase or decrease notably when processing time of each operation is converted from a deterministic value into an interval number, so the chosen ranking methods are appropriate to build interval schedule.

5.2 Algorithms comparison

The GA is compared with GPSO and SA to test its performance on IJSSP. GPSO also uses operation-based representation and produces new solutions by two-point crossover and inversion mutation. We use the following parameters for GPSO: population scale of 20 and the maximum generation of 300 for 10×10 instances and 500 for other instances. With respect to SA, we randomly produce a chromosome of operation-based representation and translate it into a schedule as the initial solution, then the initial solution is improved by exchanging a pair of randomly chosen operations on a machine and the acceptance probability is $\min\left\{1, \exp\left(-\frac{m(\sum T^c)-m(\sum T)}{T_k}\right)\right\}$, where $\sum T^c$ and $\sum T$ are total tardiness of the the new solution and the current solution. The initial temperature $T_0 = -(m_{\text{worst}} - m_{\text{best}})/\ln p_r$., m_{worst} and m_{best} are the biggest and smallest mid-point of solutions. P_r=0.02, $T_k = \lambda T_{k-1}$, λ=0.90, the number of the movements in each temperature is 30 for all instances. The parameters of GA are as follows, population scale of 100, crossover probability of 0.9, mutation probability of 0.1, the maximum generation of 300 for 10×10 instances, and 500 for other instances.

Three algorithms have the same number of objective function evaluation for the same problem. The above parameters are obtained experimentally and can yield the best results of each algorithm for most of instances. All algorithms randomly run 20 times on each instance. Table 3 shows the computational results and computational times of three algorithms, in which *min* indicates the best solution obtained in 20 runs, *max* represents the worst solution in 20 runs and $\text{avg} = \frac{1}{20}\sum_{i=1}^{20}\sum T^{i,\min}$, where $\sum T^{i,\min}$ denotes the best solution obtained in *ith* run.

From Table 3, the GA outperforms GPSO and SA on seven instances. The GA obtains the best solutions of seven instances and the worst solutions produced by the GA are always better than those of GPSO and SA for six instances. For instances 1, 2, 3, and 4, avg.$GPSO \odot$ avg.GA is close to interval [0,9], for instances 5, 6, and 7, avg.$GPSO \odot$ avg.GA is greater than [0,60], especially for instance 7, avg.$GPSO \odot$ avg.$GA > [100, 300]$. The GA produces better average results than SA for all instances, especially for three 15×10 instances, avg.$SA \odot$ avg.$GA > [0, 90]$, where avg.GPSO, avg.SA, and avg.GA indicate the avg of GPSO, SA, and GA, $A \odot B = [\min\{\underline{A} - \underline{B}, \overline{A} - \overline{B}\}, \max\{\underline{A} - \underline{B}, \overline{A} - \overline{B}\}]$.

The GA uses similar computational time with GPSO; however, the performance GA is better than that of GPSO because the GA converges to the best solutions of seven

Table 2 Results on the number of the insertable operations

Instance		Minnum	Maxnum	Avgnum	Instance		Minnum	Maxnum	Avgnum
1	Det	22.97	28.95	26.02	3	Det	23.61	29.22	26.92
	Int	22.82	28.19	25.90		Int	23.02	29.10	26.25
2	Det	20.92	27.25	25.67	4	Det	20.91	26.74	24.49
	Int	20.58	27.17	25.23		Int	20.67	26.79	24.21
5	Det	42.87	50.71	47.50	6	Det	45.42	52.34	48.34
	Int	42.75	50.20	46.56		Int	45.38	52.23	48.21
7	Det	44.81	52.31	48.40	7	Int	44.50	52.27	48.12

Table 3 Computational results and times of three algorithms

Instance	Algorithm	Min	Max	Avg	Time/s
1	GA	[5,321]	[5,340]	[4.1,328.1]	6.54
	GPSO	[12,322]	[7,340]	[5.7,335.2]	6.50
	SA	[5,327]	[5,345]	[5.9,339.5]	6.00
2	GA	[0,493]	[0,547]	[0.6,527.3]	6.59
	GPSO	[0,512]	[0,556]	[1.3,536.8]	6.71
	SA	[0,518]	[0,569]	[1.9,544.1]	6.04
3	GA	[7,459]	[0,522]	[2.7,484.1]	6.40
	GPSO	[0,480]	[0,528]	[4.5,494.3]	6.45
	SA	[7,474]	[2,535]	[6.7,507.2]	5.99
4	GA	[4,451]	[0,530]	[3.1,487.3]	6.33
	GPSO	[7,468]	[4,515]	[3.7,505.8]	6.35
	SA	[3,479]	[3,518]	[2.9,512.5]	6.05
5	GA	[79,1678]	[108,1831]	[116.3,1771.5]	21.5
	GPSO	[115,1727]	[151,2072]	[122.9,1866.5]	21.4
	SA	[159,1774]	[169,2098]	[145.2,1898.3]	20.6
6	GA	[0,1048]	[0,1195]	[1.8,1135.8]	22.0
	GPSO	[0,1137]	[1,1270]	[3.3,1192.2]	21.6
	SA	[0,1171]	[0,1308]	[4.5,1223.3]	20.7
7	GA	[69,1524]	[335,2031]	[232.5,1765.4]	21.1
	GPSO	[246,1913]	[588,2154]	[388.7,2016.3]	20.9
	SA	[321,1922]	[595,2190]	[431.5,2065.3]	19.7

instances and GPSO cannot approximate to the best solution of any instance. Although the main part of GPSO is crossover and mutation, the parents for crossover are not randomly chosen and some pairs of parents may be the same solution, in each generation, five objective function evaluations are used to choose a new particle, if GPSO and the GA have the same population scale, GPSO can only evolve one fifth of generations of GA under the same objective evaluations. These features limit the convergence ability of GPSO to the best solutions and the performance of GPSO. SA spends less time than GA and GPSO; however, its performance is also worse than GA and GPSO for its limited optimization ability, in SA only one kind of neighborhood search is done.

6 Conclusions

The uncertain scheduling problems have attracted much attention in the past decade. The literature of uncertain scheduling focuses on fuzzy scheduling and stochastic scheduling and the main concern is how to obtain uncertain schedule using various methods. The modeling cost of uncertainty is hardly considered. In fact, a number of processing data are required to obtain fuzzy membership function or stochastic distribution, and it is expensive or impossible to implement this purpose.

In this paper, interval number theory is applied to production scheduling and a new type shop scheduling problem named IJSSP is proposed. We describe the decoding procedure of operation-based representation by using the addition, max $\vee$ operator and ranking method of interval and prove that all possible actual completion time is contained in interval completion time, so we can compare solutions according to its objective value. We design an efficient GA by using GOX, swap, binary tournament selection et al. The computational experiments are executed and the results demonstrate the effectiveness of the GA.

Interval scheduling has some different features from fuzzy or stochastic scheduling: it is easy to obtain interval processing conditions such as interval processing time, the operations for building schedule are simpler than fuzzy or stochastic operations used to form schedule, the interval objective is easier to meet the requirement of decision maker than fuzzy or stochastic objective. In the near future, we will focus on interval scheduling with some actual processing constraints such as batches and flexibility. We will also investigate the new scheduling algorithms for the high quality solutions of the problems.

Acknowledgement This paper is supported by China National Science Foundation (70901064)

Appendix

job 1	[2,4]8	[3,6]6	[2,5]5	[4,6]2	[1,3]1	[3,6]3	[2,5]9	[1,3]4	[3,5]7	[2,4]10
job 2	[2,4]10	[2,5]7	[2,5]4	[1,3]6	[4,6]8	[2,7]3	[3,4]2	[1,4]1	[2,4]5	[3,5]9
job 3	[2,5]6	[1,3]9	[3,5]10	[2,4]8	[3,6]1	[1,4]7	[1,5]4	[2,4]2	[2,5]5	[1,5]3
job 4	[1,3]1	[3,5]5	[1,5]8	[2,6]9	[2,5]10	[1,4]6	[3,5]7	[1,5]2	[1,6]4	[1,4]3
job 5	[2,4]2	[1,4]7	[1,4]3	[2,3]5	[3,5]8	[4,8]9	[3,7]10	[1,4]6	[3,5]1	[3,4]4
job 6	[2,6]4	[2,4]2	[2,5]3	[4,7]5	[1,5]6	[3,6]8	[3,5]7	[2,5]9	[2,5]10	[3,6]1
job 7	[2,4]3	[1,5]5	[1,5]4	[3,5]1	[2,4]9	[3,5]7	[1,3]2	[3,6]10	[5,8]8	[2,3]6
job 8	[3,5]7	[1,3]1	[3,5]9	[2,5]6	[1,4]10	[2,4]2	[1,3]5	[2,5]3	[3,6]4	[2,5]8
job 9	[3,5]9	[1,4]4	[1,5]10	[2,4]2	[3,6]3	[2,5]6	[1,4]8	[3,5]1	[1,3]5	[3,5]7
job 10	[2,5]7	[1,3]5	[3,7]2	[2,4]4	[3,5]1	[4,7]8	[2,5]10	[3,5]6	[1,3]3	[2,4]9

Fig. 4 Processing data of instance 1

job 1	[10,16]4	[4,9]6	[10,13]7	[5,7]8	[6,9]9	[7,12]2	[10,15]5	[5,7]3	[2,5]1	[10,18]10
job 2	[3,6]2	[9,13]1	[5,9]3	[9,16]10	[5,9]6	[7,12]7	[9,14]5	[8,16]9	[2,6]8	[4,10]4
job 3	[9,14]7	[10,14]10	[5,8]9	[3,6]4	[4,8]5	[3,7]2	[3,6]8	[1,4]3	[5,9]1	[9,13]6
job 4	[10,16]5	[1,4]7	[6,10]10	[1,5]6	[7,11]4	[5,10]1	[9,14]8	[4,8]9	[4,10]2	[2,5]3
job 5	[9,15]5	[8,14]6	[10,17]10	[5,9]1	[2,5]2	[1,5]4	[7,10]7	[3,6]9	[9,13]8	[1,3]3
job 6	[4,9]8	[10,15]2	[3,5]3	[10,18]4	[5,9]1	[10,16]5	[10,15]9	[8,12]6	[5,9]10	[4,10]7
job 7	[8,13]5	[2,6]10	[10,18]8	[5,9]1	[4,8]6	[4,7]9	[7,11]7	[10,12]2	[10,15]3	[9,13]4
job 8	[10,15]2	[5,9]6	[1,4]3	[6,12]8	[4,9]4	[7,14]1	[7,13]9	[6,11]5	[8,13]10	[7,13]7
job 9	[2,6]9	[2,5]6	[2,4]2	[4,7]4	[6,9]10	[8,14]3	[4,9]1	[8,14]7	[1,3]8	[3,6]5
job 10	[5,9]7	[6,9]8	[8,16]3	[6,12]1	[7,13]9	[10,14]5	[7,11]4	[3,7]2	[3,6]10	[8,15]6

Fig. 5 Processing data of instance 2

job 1	[3,5]7	[10,13]9	[4,10]2	[5,9]5	[10,16]4	[10,13]1	[4,10]6	[4,8]3	[4,7]8	[9,13]10
job 2	[10,15]10	[9,14]3	[7,14]5	[7,13]4	[2,6]8	[8,13]1	[7,12]2	[8,14]7	[6,10]6	[10,15]9
job 3	[5,10]5	[2,5]7	[1,4]4	[7,12]3	[8,12]1	[2,5]8	[4,7]6	[7,10]2	[9,14]9	[8,15]10
job 4	[5,10]5	[4,8]6	[7,14]4	[3,8]7	[4,6]8	[8,12]2	[2,4]1	[2,5]9	[6,11]3	[6,11]10
job 5	[7,10]2	[5,7]7	[10,14]4	[2,4]1	[9,13]3	[5,9]5	[5,8]9	[1,4]6	[3,6]8	[2,6]10
job 6	[1,4]7	[3,5]3	[3,6]8	[5,8]2	[8,13]4	[9,14]9	[4,8]10	[1,4]1	[2,5]6	[6,12]5
job 7	[10,14]10	[2,4]8	[9,12]6	[9,11]3	[4,6]7	[3,7]2	[1,4]9	[2,5]4	[8,13]1	[7,11]5
job 8	[7,15]6	[9,15]2	[5,9]8	[8,13]4	[6,9]7	[8,10]3	[9,13]9	[1,4]5	[8,14]10	[6,12]1
job 9	[4,10]7	[3,6]2	[6,10]8	[3,6]6	[8,12]9	[5,10]4	[4,9]10	[1,4]1	[3,7]3	[10,15]5
job 10	[1,3]4	[8,13]1	[7,9]9	[6,12]10	[9,15]5	[7,15]2	[10,18]6	[1,5]3	[1,4]8	[2,5]7

Fig. 6 Processing data of instance 3

job 1	[7,12]2	[5,9]9	[4,8]10	[7,15]8	[8,13]6	[7,12]3	[6,9]1	[2,4]5	[4,8]7	[2,5]4
job 2	[3,6]6	[1,3]1	[1,4]5	[10,14]8	[8,14]2	[2,5]4	[5,10]7	[8,11]10	[2,5]3	[4,8]9
job 3	[9,12]9	[8,13]7	[7,10]3	[3,7]6	[10,17]2	[2,6]5	[8,13]4	[3,5]8	[1,3]1	[4,7]10
job 4	[8,13]2	[2,5]1	[4,8]8	[4,7]4	[5,8]3	[6,10]9	[2,4]10	[8,14]7	[5,8]6	[4,9]5
job 5	[4,7]3	[4,9]9	[6,12]7	[2,5]10	[1,5]4	[9,12]6	[10,12]5	[5,9]1	[8,13]2	[4,9]8
job 6	[7,9]9	[4,8]4	[4,7]1	[4,8]3	[6,9]7	[3,7]10	[9,15]8	[7,12]2	[6,12]5	[5,9]6
job 7	[5,8]6	[5,9]2	[4,7]9	[8,10]8	[9,14]4	[4,8]3	[7,12]10	[3,6]5	[9,15]1	[5,10]7
job 8	[7,14]6	[3,6]4	[10,14]3	[7,11]10	[7,13]8	[5,10]1	[7,13]9	[10,16]5	[10,16]7	[6,10]2
job 9	[2,5]10	[1,4]6	[4,8]7	[10,16]3	[3,6]8	[1,4]5	[1,3]1	[3,6]9	[5,9]2	[8,14]4
job 10	[6,9]4	[5,9]7	[5,8]5	[3,7]9	[2,5]1	[1,3]2	[8,13]3	[9,16]8	[4,8]10	[2,5]6

Fig. 7 Processing data of instance 4

job 1	[2,7]3	[9,16]4	[5,11]6	[8,15]10	[10,18]5	[7,14]7	[9,17]1	[7,15]9	[14,17]2	[6,12]8
job 2	[8,15]4	[10,12]3	[4,8]1	[11,20]2	[12,21]10	[9,15]9	[8,16]7	[5,9]6	[8,13]5	[7,15]8
job 3	[7,12]2	[6,9]1	[9,17]4	[15,20]5	[5,12]7	[11,19]10	[8,16]9	[7,13]6	[8,17]3	[14,18]8
job 4	[10,18]5	[15,21]3	[9,17]9	[8,16]6	[7,12]4	[11,19]8	[14,18]2	[13,17]7	[9,18]10	[3,8]1
job 5	[7,13]9	[8,15]10	[8,15]3	[9,16]5	[10,17]4	[9,16]1	[10,17]8	[10,15]7	[11,17]2	[12,16]6
job 6	[15,21]9	[9,18]8	[8,16]7	[10,16]10	[7,14]3	[9,17]2	[8,15]6	[10,17]5	[12,18]1	[12,20]4
job 7	[6,12]5	[7,13]6	[11,15]4	[9,16]10	[11,14]1	[8,14]9	[10,16]7	[7,15]8	[7,11]3	[5,11]2
job 8	[9,16]6	[7,13]5	[8,14]3	[6,13]7	[4,9]2	[9,17]8	[8,13]1	[7,11]4	[8,12]10	[6,10]9
job 9	[5,11]2	[7,14]6	[8,13]1	[6,8]4	[4,8]3	[7,11]8	[8,13]9	[9,14]7	[7,12]10	[6,12]5
job 10	[4,8]3	[7,12]6	[8,16]7	[5,11]10	[3,8]2	[4,9]4	[6,12]9	[7,12]1	[5,9]8	[10,17]5
job 11	[7,11]2	[9,12]5	[3,8]1	[5,10]3	[6,11]10	[8,13]9	[7,11]6	[4,7]4	[7,12]8	[9,17]7
job 12	[6,11]6	[4,10]10	[5,9]1	[6,12]5	[5,10]7	[3,9]4	[4,9]3	[5,12]2	[6,12]9	[4,10]8
job 13	[3,9]6	[7,12]10	[5,9]9	[6,11]8	[4,9]5	[8,13]7	[9,15]4	[7,13]1	[5,9]2	[7,10]3
job 14	[5,11]2	[5,8]9	[4,8]1	[7,14]3	[6,12]10	[5,10]4	[4,6]6	[8,14]7	[6,13]5	[5,11]8
job 15	[8,15]5	[7,12]4	[6,10]7	[5,10]6	[7,12]3	[8,13]9	[4,9]2	[4,10]10	[6,11]8	[3,8]1

Fig. 8 Processing data of instance 5

job 1	[5,8]10	[9,13]6	[4,8]5	[7,11]3	[8,11]8	[5,9]4	[6,10]2	[4,9]1	[8,14]9	[4,9]7
job 2	[3,8]4	[6,10]3	[7,12]5	[4,8]2	[2,6]10	[5,11]1	[6,12]7	[4,7]6	[3,6]8	[6,11]9
job 3	[6,10]9	[3,7]8	[2,6]3	[4,8]1	[5,9]10	[7,11]6	[5,9]7	[4,8]4	[1,5]2	[7,13]5
job 4	[7,14]4	[8,16]3	[6,11]7	[5,10]5	[9,17]8	[10,18]9	[4,9]6	[8,13]10	[3,7]1	[6,10]2
job 5	[5,8]5	[7,10]7	[8,16]2	[3,5]3	[10,18]8	[7,13]1	[7,14]9	[6,9]6	[7,10]4	[2,6]10
job 6	[8,16]7	[6,12]1	[7,12]5	[5,11]4	[4,9]8	[3,8]9	[9,12]2	[5,8]6	[12,16]3	[13,19]10
job 7	[14,17]4	[9,15]10	[11,19]7	[7,14]6	[9,18]1	[6,11]9	[7,12]5	[12,21]3	[8,13]8	[10,14]2
job 8	[7,14]5	[3,8]2	[6,12]9	[4,9]1	[10,18]8	[5,11]7	[6,9]6	[4,10]4	[4,8]10	[9,17]3
job 9	[13,22]10	[11,17]2	[9,16]5	[7,12]4	[6,13]9	[7,14]3	[5,10]7	[12,16]1	[17,21]8	[1,3]6
job 10	[8,15]4	[7,13]3	[9,16]7	[10,15]10	[8,13]8	[7,11]1	[6,10]5	[7,10]6	[6,10]2	[5,8]9
job 11	[7,14]2	[9,17]5	[6,12]1	[8,14]3	[10,17]10	[8,15]7	[5,11]8	[8,16]9	[6,12]6	[4,9]4
job 12	[4,9]2	[7,10]4	[6,11]1	[5,12]3	[4,10]10	[9,14]8	[7,12]9	[10,17]5	[7,11]7	[5,9]6
job 13	[3,8]6	[5,9]4	[6,10]7	[5,11]2	[4,9]1	[3,6]8	[8,13]9	[4,8]10	[6,10]3	[8,13]5
job 14	[2,5]2	[4,8]1	[6,10]8	[5,9]5	[3,9]4	[7,11]6	[6,12]10	[4,9]9	[5,8]7	[7,9]3
job 15	[5,11]5	[7,11]9	[6,9]3	[5,8]4	[4,7]2	[8,16]7	[5,12]8	[13,17]10	[4,7]6	[3,9]1

Fig. 9 Processing data of instance 6

job1	[78,88]8	[55,61]6	[72,83]9	[40,47]3	[92,101]5	[84,93]7	[3,7]4	[58,63]2	[92,101]10	[7,12]1
job2	[18,23]7	[82,91]2	[13,18]5	[35,43]6	[76,85]3	[82,89]4	[27,36]8	[51,61]9	[70,79]10	[74,82]1
job3	[35,44]1	[67,76]6	[30,39]9	[80,86]10	[65,75]4	[19,24]7	[8,13]5	[77,84]8	[44,53]3	[45,55]2
job4	[72,80]6	[13,20]3	[6,10]4	[69,77]7	[9,14]5	[58,66]8	[45,50]9	[33,38]10	[87,96]2	[53,59]1
job5	[18,23]10	[11,14]5	[67,73]7	[64,70]8	[60,67]1	[89,100]3	[12,19]9	[46,53]6	[71,80]4	[88,93]2
job6	[89,97]7	[90,98]6	[53,61]2	[67,75]8	[74,81]9	[55,62]5	[50,55]1	[27,32]3	[5,10]10	[63,71]4
job7	[53,58]8	[90,98]1	[43,53]9	[24,29]5	[79,86]3	[60,67]2	[33,40]10	[24,30]4	[82,91]7	[5,7]6
job8	[72,80]4	[13,18]6	[73,82]10	[5,11]2	[38,44]9	[33,40]3	[28,33]5	[82,87]7	[33,38]1	[74,80]8
job9	[70,79]1	[10,17]8	[78,85]3	[26,31]9	[51,56]5	[80,86]7	[84,92]6	[75,82]2	[37,42]10	[11,16]4
job10	[5,8]3	[23,28]2	[29,36]8	[60,67]7	[51,58]5	[49,56]1	[80,85]6	[4,8]4	[83,92]10	[50,59]9
job11	[58,67]9	[63,71]3	[29,34]6	[60,65]1	[65,74]8	[58,65]4	[32,39]2	[70,75]5	[4,6]10	[90,97]7
job12	[76,81]3	[86,95]10	[81,90]1	[69,75]2	[61,68]9	[60,68]7	[9,12]4	[80,85]8	[81,92]6	[5,9]5
job13	[24,33]5	[9,13]10	[47,54]8	[85,93]7	[40,49]1	[29,34]6	[25,30]3	[63,70]2	[45,53]9	[31,40]4
job14	[12,17]3	[35,43]6	[53,61]7	[60,65]5	[93,101]4	[63,71]10	[65,74]8	[6,10]2	[43,51]9	[73,81]1
job15	[13,23]2	[89,99]9	[50,63]8	[41,51]7	[63,73]4	[52,61]10	[37,41]3	[49,58]6	[79,82]5	[60,64]1

Fig. 10 Processing data of instance 7

1 [25,30] [33,41] [23,30] [45,50] [21,28] [35,36] [22,31] [32,38] [28,36] [30,39]

2 [103,111] [91,111] [78,89] [73,93] [82,97] [103,118] [103,112] [91,113] [60,77] [94,112]

3 [94,104] [111,126] [79,94] [70,89] [73,81] [63,79] [82,87] [100,114] [70,89] [78,99]

4 [78,95] [66,80] [82,93] [72,86] [79,93] [82,96] [88,99] [108,123] [57,75] [67,83]

5 [107,142] [114,144] [126,153] [138,164] [131,157] [140,172] [113,137] [100,128] [93,116] [82,114] [91,112] [67,104] [85,110] [77,107] [81,110]

6 [114,142] [87,127] [83,120] [125,175] [117,152] [136,172] [176,215] [110,162] [167,201] [138,169] [134,191] [121,161] [98,135] [93,126] [114,149]

7 [987,1049] [897,971] [799,878] [756,816] [894,947] [974,1027] [838,894] [766,820] [887,937] [737,795] [899,948] [1033,1080] [676,750] [855,918] [906,984]

Fig. 11 Due date of seven instances

In Figs. 4, 5, 6, 7, 8, 9, 10, each job has ten operations and the data on each operation is composed of two parts: the first is interval processing time and the second is the number of machine on which the operation is processed.

In Fig. 11, for instance 1–4, ten intervals from left to right are due dates of job J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}.

For instance 5–7, 15 intervals from left to right are due dates of job J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}, J_{11}, J_{12}, J_{13}, J_{14}, J_{15}.

References

1. Sakawa M, Mori T (1999) An efficient genetic algorithm for job shop scheduling problems with fuzzy processing time and fuzzy due date. Comput Ind Eng 36:325–341
2. Sakawa M, Kubota R (2000) Fuzzy programming for multi-objective job shop scheduling with fuzzy processing time and fuzzy due date through genetic algorithm. Eur J Oper Res 120:393–407
3. Li FM, Zhu YL, Yin CW, Song XY (2005) Fuzzy programming for multi-objective fuzzy job shop scheduling with alternative machines through genetic algorithm. In: Wang L, Chen K, Ong YS (eds) Advance in Natural Computation. Springer, New York, pp 992–1004
4. Song XY., Zhu YL., Yin CW., Li FM. (2006) Study on the combination of genetic algorithms and ant colony algorithms for solving fuzzy job shop scheduling problems. In Proceedings of IMACS multi-conferences on computational engineering in systems applications, 1904–1909
5. Niu Q, Jiao B, Gu XS (2008) Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. Appl Math Comput 205:148–158
6. Lei DM (2008) Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems. Int J Adv Manuf Technol 37:157–165
7. Petrovic S, Fayad C, Petrovic D, Burke E, Kendall G (2008) Fuzzy job shop scheduling with lot-sizing. Annals of Operational Research 159:275–292
8. Lei DM (2010) A genetic algorithm for flexible job shop scheduling with fuzzy processing time. Int J Prod Res 48:2995–3013
9. Lei DM (2010) Solving fuzzy job shop scheduling using random key genetic algorithm. Int J Adv Manuf Technol 49:253–262
10. Tavakkoli-Moghaddam R, Jolai F, Vaziri F, Ahmed PK, Azaron A (2005) A hybrid method for solving stochastic job shop scheduling problem. Appl Math Comput 170:185–206
11. Hu YM, Yin MH, Li XT (2011) A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. Int J Adv Manuf Technol, in press
12. Lei DM (2011) Scheduling stochastic job shop scheduling subject to random breakdown to minimize makespan. Int J Adv Manuf Technol 55:1183–1192
13. Gu JW, Gu XS, Gu M (2009) A novel parallel quantum genetic algorithm for stochastic job shop scheduling. J Math Anal Appl 355:63–81
14. Gu JW, Gu M, Gu XS (2010) A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. Comput Oper Res 37:927–937
15. Moore RE (1979) Methods and applications of interval analysis. Prentice-Hall, London
16. Hao PY (2009) Interval regression analysis using support vector networks. Fuzzy Sets and Systems 160:2466–2485
17. Lhommeau M, Hardouin L, Cottenceau B, Jaulin L (2004) Interval analysis and dioid: application to robust controller design for timed event graphs. Automatica 40:1923–1930
18. Jiang C, Han X, Liu GR, Liu GP (2008) A nonlinear interval number programming method for uncertain optimization problems. Eur J Oper Res 188:1–13
19. Sengupta A, Pal TK (2000) On comparing interval numbers. Eur J Oper Res 127:28–43
20. Ishibuchi H, Tanaka M (1990) Multiobjective programming in optimization of the interval objective function. Eur J Oper Res 48:219–225
21. Cheng RW, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms: I. Representation. Comput Ind Eng 30:983–997
22. Goldberg DE, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins G (ed) Foundations of genetic algorithms. Morgan Kaufmann, New York
23. Bierwirth C (1995) A generalized permutation approach for job shop scheduling with genetic algorithms. OR Spectrum, Special issue: Applied Local Search 17(1):87–92