Int J Adv Manuf Technol (2011) 55:1183–1192
DOI 10.1007/s00170-010-3151-z

ORIGINAL ARTICLE

Scheduling stochastic job shop subject to random breakdown to minimize makespan

Deming Lei

Received: 8 May 2009 / Accepted: 30 December 2010 / Published online: 12 January 2011

Abstract The problem of scheduling stochastic job shop subject to breakdown is seldom considered. This paper proposes an efficient genetic algorithm (GA) for the problem with exponential processing time and *non-resumable* jobs. The objective is to minimize the stochastic makespan itself. In the proposed GA, a novel random key representation is suggested to represent the schedule of the problem and a discrete event-driven decoding method is applied to build the schedule and handle breakdown. Probability stochastic order and the addition operation of exponential random variables are also used to calculate the objective value. The proposed GA is applied to some test problems and compared with a *simulated annealing and a particle swarm optimization*. The computational results show the effectiveness of the GA and its promising advantage on stochastic scheduling.

Keywords Stochastic job shop scheduling · Breakdown · Stochastic order · Random key representation · Discrete event driven

1 Introduction

Machine breakdowns are not uncommon for machines running for long periods without maintenance. Such breakdowns make shop behavior hard to predict and reduce the efficiency of production systems; however, these phenomena attract less attention and most literature on scheduling assumes that machines are always available during the planning time horizon.

Some results have been obtained for open shop, flow shop, and single machine subject to random breakdown. Li and Cao [1] analyzed single machine stochastic scheduling problems subject to random breakdowns. Kasap et al. [2] considered the optimality of LPT rule for the expected makespan problems with an unreliable machine. Flow-shop problems with random breakdowns and the constant processing times are studied by Allahverdi [3, 4] and Allahverdi and Mittenthal [5, 6], and some effective heuristics are proposed. General stochastic flow-shop scheduling problems subject to breakdowns are also studied by Alcaide et al. [7], where a general procedure to convert problems with breakdowns into problems without breakdowns is proposed. The above conversion method is also used by Alcaide et al. [8] to minimize the expected makespan on open shop subject to stochastic processing time and breakdown.

Stochastic job shop scheduling problem (SJSSP) is an important aspect of manufacturing systems and the extended version of job shop scheduling problem (JSSP) by introducing some stochastic processing constraints such as stochastic processing time. For SJSSP, Luh et al. [9] presented an effective approach for JSSP considering uncertain arrival times, processing time, due date, and part priority. A solution methodology based on a combined Lagrangian relaxation and a stochastic dynamic programming is developed to obtain the dual solutions. Singer [10] derived a heuristic for allowing the use of a deterministic algorithm for scheduling a job shop subject to stochastic processing time. Ginzburg and Gonik [11] considered three sets of cost in SJSSP with stochastic processing time in normal, exponential, and uniform distributions and treated the problem as the identification of the earliest start time in

D. Lei (✉)
School of Automation, Wuhan University of Technology,
Wuhan 430070 Hubei Province, People's Republic of China
e-mail: deminglei11@163.com

order to minimize the average cost of storage and tardiness from the delivery time. Tavakkoli-Moghaddam et al. [12] proposed a hybrid method based on neural network and simulated annealing (SA), in which a neural network is applied to generate an initial feasible solution and then a SA is used to improve the quality of the initial solution. Lei and Xiong [13] presented an efficient genetic algorithm (GA) to minimize the expected makespan and the expected total tardiness. Lei and Xiong [14] presented a permutation-based representation method and then designed a GA to minimize makespan. Gu et al. [15, 16] *proposed two effective quantum genetic algorithm.*

The literature on stochastic scheduling and breakdown has the following features. (1) The expected makespan is always used as objective in most of literature and only Cai et al. [17] and Lei and Xiong [14] regarded the stochastic makespan as the objective of the problem. (2) Meta-heuristics are seldom applied to solve the problem with random breakdown. (3) The combination of SJSSP and random breakdown is hardly considered for the high complexity.

In this paper, we propose an efficient GA to deal with SJSSP with random breakdown. The objective is to minimize the stochastic makespan. To calculate the makespan, probability stochastic order and the addition operation of random variables are used. A discrete event-driven decoding procedure is also applied to convert the chromosome of random key representation into the feasible schedule of problem; meanwhile, machine breakdown and repair are also handled in the procedure. Binary tournament selection, discrete crossover (DX) and swap are applied to produce new population. The proposed GA is finally tested on a set of the computational experiments.

The remainder of this paper is organized as follows. The addition operation of exponential random variables is defined in section 2 and the problem is described in section 3. The random key scheduling algorithm is depicted in section 4. The computational experiments and discussions are done in section 5. The final conclusions are drawn in the final section.

2 Exponential processing time

In stochastic production process, the processing conditions and constraints are stochastic and often modeled as random variable with exponential, normal, or uniform distribution. Exponential distribution is often reasonable for processing time if there is a high degree of uncertainty about the processing time in the sense that only information available on it is its mean [17], so exponential distribution is often used to model processing time.

For exponential random variable $X \sim \exp(\lambda)$, its density function is as follows.

$$f_X(z) = \begin{cases} \lambda e^{-\lambda z} & \text{if } z > 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

Where parameter λ represents the rate of X and is reciprocal to the mean of X.

For $X_1 \sim \exp(\lambda_1)$ and $X_2 \sim \exp(\lambda_2)(\lambda_1 \neq \lambda_2)$, the density function of X_1+X_2 is expressed as:

$$f_{X_1+X_2}(z) = \begin{cases} \frac{\lambda_1\lambda_2}{\lambda_2-\lambda_1}\left[e^{-\lambda_1 z} - e^{-\lambda_2 z}\right] & \text{if } z > 0 \\ 0 & \text{else} \end{cases} \tag{2}$$

the cumulative distribution function(cdf) is written as:

$$F_{X_1+X_2}(z) = \begin{cases} 1 - \left(\frac{\lambda_2}{\lambda_2-\lambda_1} e^{-\lambda_1 z} - \frac{\lambda_1}{\lambda_2-\lambda_1} e^{-\lambda_2 z}\right) & \text{if } z > 0 \\ 0 & \text{else} \end{cases} \tag{3}$$

As shown in Eq. 3, X_1+X_2 does not follow the exponential distribution. To obtain the exponential completion time and makespan, it is necessary to find an approximation of X_1+X_2. The variable X with the parameter λ is used to approximate X_1+X_2, where λ is defined as follows.

$$\lambda = \begin{cases} \lambda_1\lambda_2 / (\lambda_2 + 1.5\lambda)_1 & \text{if } \lambda_2 > \lambda_1 \\ \lambda_1\lambda_2 / (\lambda_1 + 1.5\lambda)_2 & \text{else} \end{cases} \tag{4}$$

Suppose $\lambda_2 > \lambda_1$, for $z > 8/\lambda_1, |F_X(z) - F_{X_1+X_2}(z)| <, 1e-6$, for 500 uniformly selected $z_i \in [0, 8/\lambda_1]$, $\max_{i=1,2,\cdots 500} |F_{X_1+X_2}(z_i) - F_X(z_i)| < 0.1$; moreover, if $|1/\lambda_2 - 1/\lambda_1| > 10$, $\max_{i=1,2,\cdots 500} |F_{X_1+X_2}(z_i) - F_X(z_i)| < 0.05$. For other conditions of Eq. 4, the same conclusion is also can be drawn. X can be applied to approximate X_1+X_2.

The ranking of random variables can be done based on stochastic order theory. Many kinds of stochastic order including expectation order and probability order et al. have been defined; some of them can be easily used on computer. In this study, probability stochastic order is considered.

Definition 1 X_1 and X_2 are random variables, X_1 is stochastically greater than or equal to $X_2 (X_1 \geq_{st} X_2)$ if $\forall z \Pr(X_1 > z) \geq \Pr(X_2 > z)$

For $X_1 \sim \exp(\lambda_1)$ and $X_2 \sim \exp(\lambda_2)$, if $\lambda_1 < \lambda_2$, for $\forall z > 0, 1 - e^{-\lambda_1 z} < 1 - e^{-\lambda_2 z}$, then $X_1 \geq_{st} X_2$.

3 Problem description

The problem under consideration is composed of n jobs J_i and m machines M_k, $i=1, 2, \cdots, n$, $k=1, 2, \cdots, m$. Each job consists of several operations. Operation o_{ij} indicates the jth

operation of job J_i and its processing time $\widetilde{p}_{ij}$ can be modeled by the exponential distribution, that is to say, the processing time follows exponential distribution, i.e. $\widetilde{p}_{ij}$ has a density function of the form $\lambda_{ij}e^{-\lambda_{ij}x}$, or equivalently, a cdf of $1-e^{-\lambda_{ij}x}$.

When machines are subject to random breakdown, $u_{k\upsilon} \sim \exp(\lambda_{uk\upsilon})$ denotes the length of time between the $(\upsilon-1)$th and υth breakdown on machine M_k, $d_{k\upsilon}\sim\exp(\lambda_{dk\upsilon})$is the υth repair time. A breakdown can interrupt the processing of at most one operation. Suppose that the processing a job is interrupted by breakdown of a machine, on which the job is processed, if the job has to restart processing from beginning when machine becomes available, the job is non-resumable; if the job may continue its processing when machine become available, the job is resumable. In this study, all jobs are non-resumable.

There are several constraints on jobs and machines, 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,
- Operations of a given job have to be processed in a given order,
- Setup times and remove times are included in the processing times.

In this study, the following objective is discussed.

$$\text{Minimize } C_{\max} = \max_{i=1,2,\cdots n} C_i \tag{5}$$

With respect to $C_{\max}\sim\exp(\lambda_{\max})$, first let $\lambda_{\max}=\alpha$, then for $i=1, 2,\cdots, n$, if the parameter λ_i of C_i is smaller than $\lambda_{\max}$, then $\lambda_{\max}=\lambda_i$. Where α is a big enough positive number.

4 Genetic algorithm for SJSSP subject to breakdown

GA has been extensively applied to deal with *production scheduling problem with* the processing constraints such as flexible process plan, preventive maintenance and multi-objective, etc. [18–20]; however, it is hardly used to solve the problem with breakdown. In this study, all things related to machine breakdown are done in the decoding procedure. These things are as follows: the condition of the occurrence of breakdown, the repair of machine and the treatment of the interrupted operation.

4.1 Coding and decoding

In this study, we present a new random key representation, which encodes a schedule of $n\times m$ SJSSP as a real string $(p_1, p_2,\cdots, p_n,\cdots, p_{mn})$ with $n\times m$ random numbers in the same interval $[a, b)$.

To obtain a feasible schedule, the following decoding procedure is adopted.

1 Divide the interval $[a, b)$ into a group of sub-intervals $[a_1, a_2)$, $[a_2, a_3),\cdots, [a_l, a_{l+1})$, where $a=a_1<a_2<\cdots a_l<a_{l+1}=b$. Let np_k indicates the number of genes in sub-interval $[a_k, a_{k+1})$, determine np_k for each sub-interval, $k=1, 2, \cdots, l$, let $t=1$, $\upsilon_h=1$and $op_i=0$, $i=1, 2,\cdots, n$, $h=1, 2,\cdots,m$, $\upsilon_k=1$, $k=1, 2,\cdots, m$.

2 For $p_t \in [a_j, a_{j+1})$, $1\le j\le l$, determine $f_t=\left|\left\{p_q \in \left[a_j, a_{j+1}\right)\middle|\right\}\right.$ $p_t>p_q$ *or* $\left\{p_q=p_t \text{ and } q\le t\right\}|$, $kt=f_t+\sum_{i=1}^{j-1} np_i$, if $(z-1)$ $m<kt\le zm$, then $op_z=op_z+1$. The operation $o_{zop_z}$of gene p_t is assigned the best available time on the required machine M_k.

3 $b=\{\upsilon_k\}$, If $\tau_{zop_z} \ge_{st} u_{k\upsilon_k}$, repeat $\upsilon_k=\upsilon_k+1$, $b=b\cup\{\upsilon_k\}$ until $\tau_{zop_z}\le_{st} u_{k\upsilon_k}$.

For each $\omega \in b\backslash\{\upsilon_k, \upsilon_k-1\}$, breakdown does not happen and maintain the machine;

For $\omega=\upsilon_k-1$, breakdown may happen in the processing of o_{zop_z}, repair the machine;

Determine the beginning time σ_{zop_z} again and make o_{zop_z} to be processed.

4 $t=t+1$, if $t<mn$, go to (2); else stop the decoding and the final schedule is formed.

Where op_i denotes the serial number of operation of job J_i.σ_{ij} and τ_{ij} indicate respectively the beginning time and completion time of o_{ij}.

In (2), to decide σ_{zop_z} for operation $o_{zop_z}(op_z>1)$ processed on machine M_k, $\tau_{z(op_z-1)}$is compared with η_k, if $\tau_{z(op_z-1)}\ge_{st}\eta_k$, then $\sigma_{zop_z}=\tau_{z(op_z-1)}$; else $\sigma_{zop_z}=\eta_k$. For operation o_{z1} processed on machine M_k, $\sigma_{z1}=\eta_k$ for $\lambda_k\infty$. If $\tau_{zop_z}\le_{st}\sigma_{lr}$, o_{zop_z}can be inserted before o_{lr}.

In (3), it is possible more than one breakdown is not handled before the processing of o_{zop_z}, these breakdowns must be dealt with ahead of o_{zop_z}. Two cases may happen: breakdown occurs in the processing of o_{zop_z} and its processing is interrupted, only one operation is possible to be interrupted once; or no processing is interrupted, the maintenance of a machine not only avoids the occurrence of breakdown but guarantees the enough duration of the normal run for the machine.

Suppose a chromosome of the 3×3 SJSSP in Table 1 is ((1,1),(2,1),(3,1),(3,2),(2,2),(3,3),(1,2),(1,3)(2,3)). The corresponding operation list is (o_{11}, o_{21}, o_{31}, o_{32}, o_{22}, o_{33}, o_{12}, o_{13}, o_{23}). η_k indicates the available time of machine M_k, initially, $\eta_k\sim\exp(\infty)$ and $\sigma_{i1}\sim\exp(\infty)$. λ_{bij}, λ_{cij} and λ_k denote the parameter of σ_{ij}, τ_{ij} and η_k.

For operation o_{11}, $\sigma_{11}\sim\exp(\infty)$, $\tau_{11}\sim\exp(0.25)$, $\eta_1\sim\exp(0.25)$.

For operation o_{21}, $\eta_1\ge_{st}\sigma_{21}$ for $0.25<\infty$, so $\lambda_{b21}=0.25$, $\lambda_{c21}=0.0909091$,

Table 1 An illustration of the problem under study

J_i	Operations						M_k	λ_{ukv}	λ_{dkv}
	1	2	3	1	2	3			
	Processing time (λ)			Processing sequence					
1	1/4	1/6	1/5	M_1	M_3	M_2	M_1	0.05	0.3
2	1/5	1/7	1/2	M_1	M_2	M_3	M_2	0.04	0.4
3	1/8	1/6	1/3	M_1	M_3	M_2	M_3	0.04	0.2

For operation o_{31}, λ_{b31}=0.0909091 for $\lambda_{b31}=\lambda_1$, $\lambda_{c31}=\lambda_{b31}p_{31}/(1.5\lambda_{b31}+p_{31})$=0.0434783. $\tau_{31}>_{st}u_{11}$ for λ_{31}<0.05, breakdown occurs on M_1 and o_{31} is interrupted.

When repair is finished, $\eta_1 = u_{11} + d_{11} \sim \exp(0.04)$ and M_1 is available, λ_{b31}=0.04, λ_{b31}=0.027.

For operation o_{32}, λ_{b32}=0.027 for $\lambda_{b32}<\infty$ and λ_{c32}=0.0217. $\tau_{32}\geq_{st}u_{31}$ for 0.0217<0.04, no operation is interrupted, unavailable period is determined, $\eta_3\sim\exp(0.03)$.

For operation o_{22}, λ_{b22}=0.0909091, λ_{c22}=0.0465156.

For operation o_{33}, λ_{b33}=0.0217 and λ_{c33}=0.0197.

For operation o_{12}, λ_{b12}=0.25, λ_{c12}=0.083333>0.04, o_{12} can be processed before the unavailable period. For o_{13}, if $\lambda_{b13}=\lambda_{c22}$, then λ_{c13}=0.0345, for $\tau_{13}\geq_{st}u_{21}$, the processing of o_{13} is interrupted, $\eta_3\sim\exp$ (0.0307). If λ_{b13}=0.0307, then λ_{c13}=0.025 and o_{13} is allocated before o_{33} for 0.025> 0.0217.

For operation o_{23}, λ_{b23}=0.0217, λ_{c23}=0.0203.

Figure 1 shows the decoding procedure, which has the different features from Gantt chart of JSSP. The numbers do not represent the actual completion time and are reciprocal to the parameter of completion time. $1/\lambda_{cij} \neq 1/\lambda_{bij} + 1/\lambda_{ij}$.

With respect to λ_{cij}, i λ_{ij} *is always greater than* λ_{bij} in most cases, especially for some operations, the condition $\left|1/\lambda_{bij} - 1/\lambda_{ij}\right| > 10$ is met, so the approximate value of λ_{cij} is very close to the real λ_{cij}. The addition operation defined in section 2 is reasonable.

In Fig. 1, breakdown is regarded as discrete event, when the condition ($\tau_{ij}\geq_{st}u_{kv}$) of event is met, machine is repaired or maintained. When the obtained schedule is used, the processing on each machine is done in terms of the generated operation sequence. Take M_3 as an example, after o_{12} is processed, breakdown does not happen most likely for $\tau_{12}\leq_{st}u_{31}$, machine is directly maintained in its idle time, and then the processing of o_{32} and o_{23} is sequentially done after the machine is available.

As shown above, addition and ranking of exponential random variables are required to build the schedule of the problem. If resumable jobs are considered, to calculate the remaining processing time of the interrupted operation, subtraction of exponential random variables is also needed to be defined and approximated to make the subtraction follow exponential distribution, as a result, there are two approximate operations of exponential random variables and the obtained schedule results may deviate notably from the real results. New methods are necessary for SJSSP with resumable jobs, so only non-resumable jobs are considered.

4.2 Genetic operators

In this paper, we make fitness function of an individual be equal to its objective function.

The classical elite strategy is used, in which the optimal solution produced by GA is stored as an elite individual, moreover, the elite individual is always added into population and participates in reproduction in each generation.

Roulette wheel reproduction and breeding pool reproduction cannot be applied for the random objective, so tournament selection [21] *is used and performed in the following way*: randomly select two individuals from the population, choose one of them if its fitness is smaller than or equal to that of another individual and put them back into the population, repeat the procedure until population scale is met. Fitness is a stochastic variable, so *probability stochastic order* in section 2 is used to compare them.

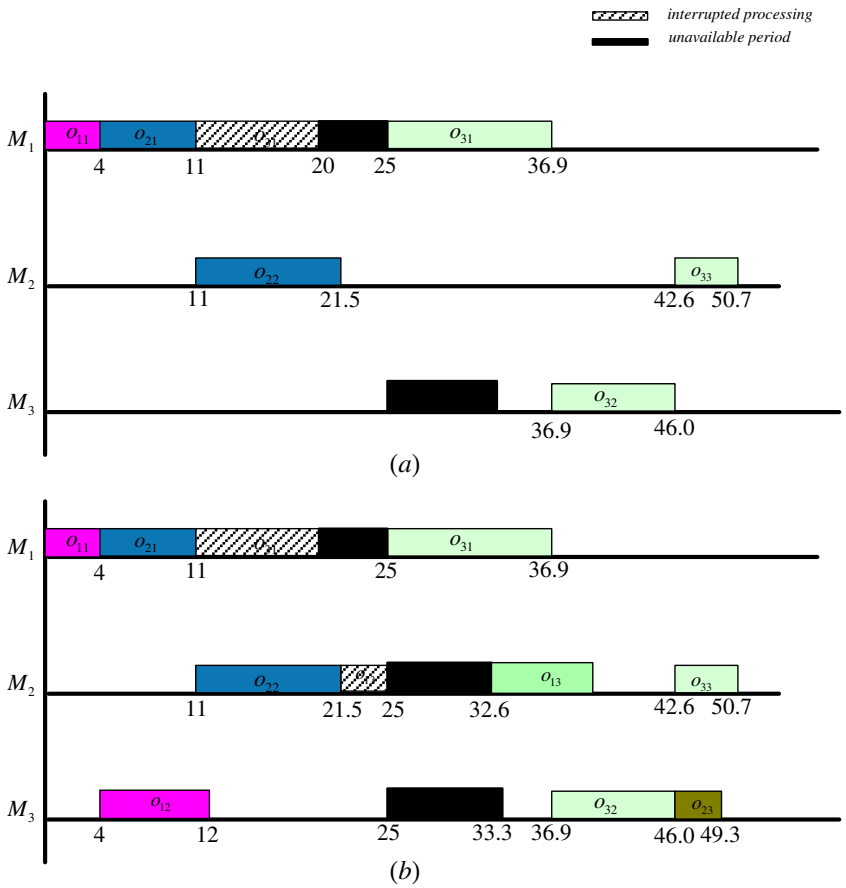
DX is frequently used in the real-coded GA. DX is done in the following way: produce the random number s following the uniform distribution on [0, 1], if s<0.5, select the gene of one parent; otherwise, select the gene of another parent; repeat the above step until an offspring is obtained.

Mutation is just used to produce small perturbations on chromosomes in order to maintain the diversity of population. The swap mutation is adopted and described as follows: randomly choose two genes and then exchange them.

4.3 Algorithm description

The proposed GA has the following features: the chromosome is real string and can be converted into the ordered operation list, which never violates the precedence constraints, and no special genetic operators are required, machine breakdown and repair are dealt with only in the

Fig. 1 The decoding procedure



decoding procedure and genetic operators are not related to the handling of breakdown. These features make the optimization of the problem be simple.

The overall structure of our GA can be described as follows.

1. generate an initial population P with N individuals, calculate the objective value for each individual and reserve the elite individual.
2. Perform binary tournament selection on *P*.
3. Perform DX and swap mutation on population *P*.
4. Calculate the objective value of each individual and update the elite individual if possible
5. If the predetermined number of generations is met, stop the search; otherwise, go to step 2.

5 Computational results

In this section, our GA is tested on some problems and compared with SA [22] and a PSO. 22 benchmark problems are used. These problems are the extension of ORB1–10, ABZ5–6, and LA16–25 et al. The processing times of these deterministic problems are the reciprocal of λ_{ij}. λ_{1010} of ORB7 is defined to be 1. To simplify, these extended problems are still called ORB1–10, ABZ5–6, and LA16–25. Data related to machine breakdown and repair time are shown in appendix. All experiments are implemented by using Microsoft Visual C++ 7.0 and run on 512M RAM 2.0G CPU Pentium PC.

With respect to SA, Tavakkoli-Moghaddam et al. [12] proposed a SA-based method for SJSSP and an initial solution is generated by a neural network approach. In this study, we randomly produce a chromosome of random key representation and translate it into a schedule as the initial solution for simplicity, then the initial solution is improved by exchanging a pair of randomly chosen operations on a machine and the acceptance probability is $\min\{1, \exp\left(-\left(1/\lambda_{\max} - 1/\lambda^{c}_{\max}\right)/T_k\right)\}$, where $1/\lambda^{c}_{\max}$ and $1/\lambda_{\max}$ are the mean value of makespan of the current solution and the new solution, T_k is the temperature at the *k*th iteration.

For the continuous nature of particle swarm optimization (PSO) [23], some strategies have been used to apply PSO to JSSP. For example, Lei [24] transforms JSSP into the continuous one and solve the latter by using PSO. In this study, the real chromosome (p_1, p_2,⋯, p_n, ⋯, p_{mn}) is regarded as vector of position and can be directly optimized by PSO itself.

We adopt mutation probability of 0.1, crossover probability of 0.7 and population scale of 150 for all problems. For all problems except LA21–25, the maximum generation

of 500. For LA21–25, the maximum generation is 700. These parameters are determined experimentally. We adopt the following parameter settings for SA: the initial temperature $T_0 = -(1/\lambda_{\text{worst}} - 1/\lambda_{\text{best}})/\ \ln p_r$, $1/\lambda_{\text{worst}}$ and $1/\lambda_{\text{best}}$ are the biggest mean value and smallest mean value of solutions. p_r=0.02, $T_k=\lambda T_{k-1}$, λ=0.95, the number of the movements in each temperature is 300 for all problems except LA21–25 and 420 for LA21–25. The parameters of PSO are as follows: population scale of 150 and the maximum generation of 500 for all problems except LA21–25 and 700 for LA21–25, inertia weight diminishes linearly from 1.2 to 0.4, learning factors $c_1=c_2=2.0$. All algorithms randomly run 20 times on each instance. Table 2 shows the computational results and Table 3 gives the average computational times of three algorithms. $\lambda^i_{\max}$ denotes the parameter of makespan produced in the ith runs. Min is the smallest mean value of makespan obtained in all runs. To decide min, first let min=α, then for any $\lambda^i_{\max}$, if $\min \le 1/\lambda^i_{\max}$, then $\min = 1/\lambda^i_{\max}$. Max indicates the biggest mean value of makespan produced in 20 runs, first define max=0, then for any $\lambda^i_{\max}$, if $\max \ge 1/\lambda^i_{\max}$, $\max = 1/\lambda^i_{\max}$, where α is a big enough positive number. $\text{avg} = \frac{1}{20}\sum_{i=1}^{20} 1/\lambda^i_{\max}$. Figure 2 shows a schedule of ORB2.

Table 3 Computational times of three algorithms

Problem	GA	SA (s)	PSO
ABZ5–6	15.6 s	13.9	14.2 s
ORB1–10	14.7	13.2	13.6 s
LA16–20	15.1 s	12.6	13.9 s
LA21–25	39.7	36.5	37.6

As shown in Tables 2 and 3, our GA produces the less value of min than SA and PSO on 21 of 22 problems and three algorithms converge to the same solution with identical min on LA16; moreover, the difference of min between GA and other algorithm is at least 30 on 15 problems. With respect to avg, our GA produces smaller value of avg than PSO and SA by at least 30 on 13 problems and generates the similar results with other algorithms on LA16 and ORB7. For max, our GA has the close results to SA and PSO on ABZ5, ORB4, ORB7, ORB9, and LA16, the better results on 17 problems than SA and PSO. Our GA performs better than SA and PSO in most cases in a slightly long times.

The schedule of SJSSP is only decided by the relative sequence of random keys and the different chromosome

Table 2 Computational results of three algorithms

problem	Min			Max			Avg		
	GA	SA	PSO	GA	SA	PSO	GA	SA	PSO
ABZ5	1,895.14	1,949	1,931.57	2,042.08	2,042.08	2,042.08	1,985.6	2,031.67	2,005.59
ABZ6	1,440.83	1,458.3	1,455.79	1,481.33	1,508.33	1,530.43	1,468.45	1,493.3	1,481.77
ORB1	1,675.51	1,739.5	1,731.01	1,795.51	1,847.5	1,834.95	1,742.26	1,798.29	1,784.53
ORB2	1,398.79	1,478.75	1,467.79	1,478.75	1,512.79	1,512.79	1,443.3	1,497.76	1,476.42
ORB3	1,625.25	1,717.48	1,670.25	1,786.17	1,836.25	1,810.5	1,711.8	1,779.73	1,749.22
ORB4	1,597.23	1,640.05	1,631.22	1,719.72	1,719.72	1,719.72	1,634.35	1,690.33	1,694.33
ORB5	1,385.83	1,418.5	1,413.31	1,517.01	1,549.64	1,549.64	1,435.87	1,469.17	1,451.4
ORB6	1,623.57	1,670.07	1,678.32	1,727.07	1,752.57	1,752.57	1,682.95	1,720.72	1,713.88
ORB7	634.581	635.116	635.116	659.75	676.581	659.75	643.104	654.636	645.645
ORB8	1,390.17	1,454.75	1,445.04	1,504.17	1,551.54	1,543.94	1,455.98	1,504.1	1,492.52
ORB9	1,280.71	1,339.21	1,328.71	1,367.95	1,370.56	1,370.56	1,333.91	1,367.2	1,363.97
ORB10	1,450.02	1,496.21	1,488.5	1,535.52	1,576.43	1,552.02	1,492.06	1,545.12	1,529.08
LA16	1,486.39	1,486.39	1,486.39	1,533.56	1,533.56	1,533.56	1,502.71	1,511.58	1,510
LA17	1,183	1,193.54	1,193.54	1,204	1,228.04	1,228.04	1,199.76	1,221.55	1,212.89
LA18	1,287.54	1,326.21	1,304	1,385.74	1,401.54	1,385.74	1,324.09	1,357.43	1,349.85
LA19	1,305.96	1,355.58	1,350.05	1,371.55	1,388.39	1,395.6	1,344.81	1,372.75	1,371.61
LA20	1,390.51	1,420.63	1,390.51	1,446.13	1,499.63	1,479.01	1,405.16	1,451.55	1,448.95
LA21	1,670.29	1,753.19	1,717.26	1,768.26	1,817.29	1,790.76	1,723.16	1,780.58	1,767.59
LA22	1,471.6	1,526.05	1,517.27	1,579.36	1,607.23	1,611.34	1,499.14	1,593.47	1,570.8
LA23	1,517.27	1,611.57	1,593.89	1,614.89	1,679.07	1,679.07	1,599.64	1,660.78	1,649.73
LA24	1,486.89	1,589.27	1,563.51	1,569.39	1,648.2	1,662.8	1,518.76	1,616.51	1,589.02
LA25	1,558.21	1,606.21	1,593.24	1,596.6	1,674.38	1,620.24	1,567.02	1,624.33	1,594.79

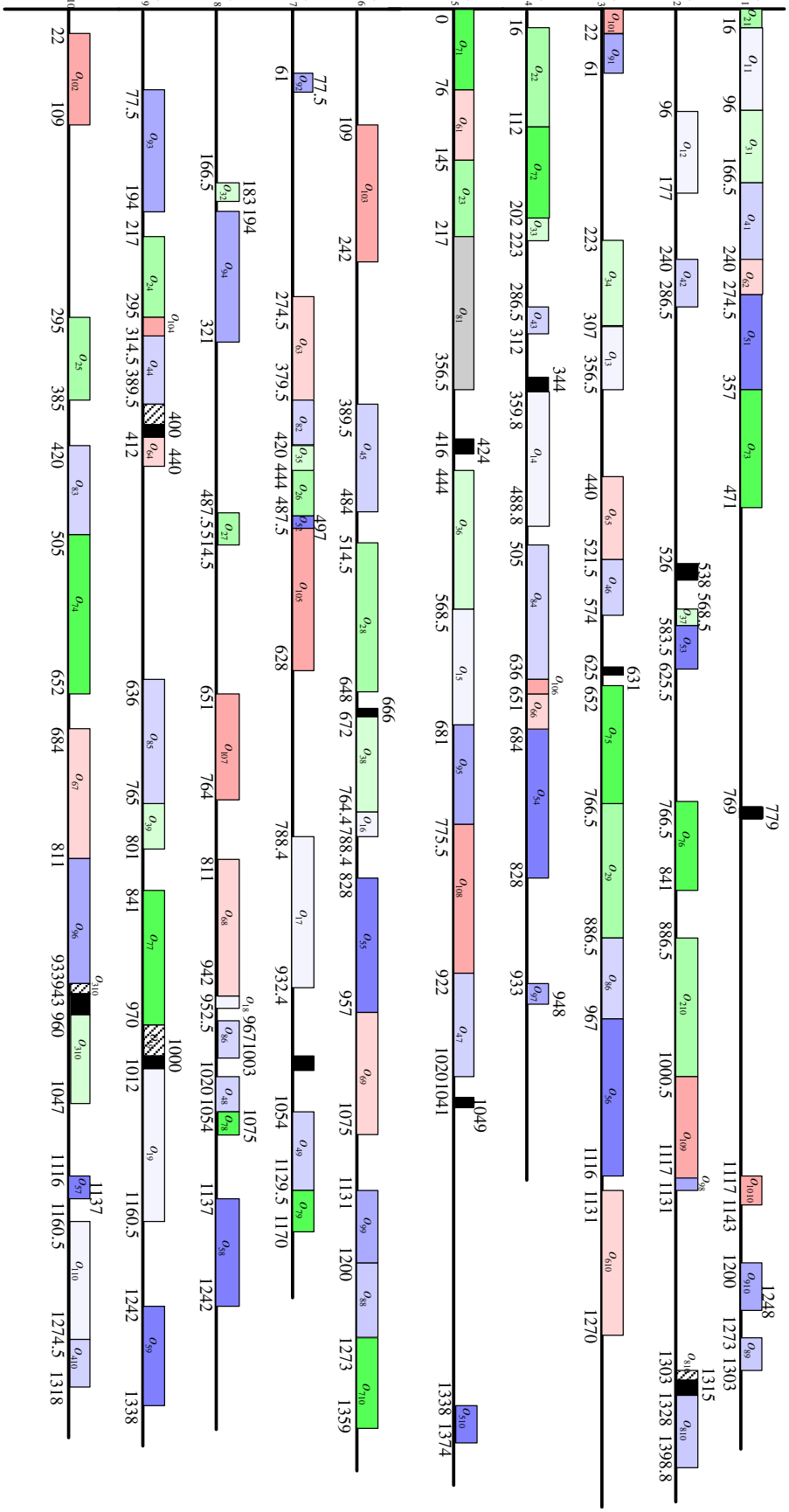


Fig. 2 A schedule of ORB2

may correspond to the same schedule. In GA, no new genes are produced and genes are just moved from one individual to other. PSO also can directly act on chromosome and obtains new solutions by continuously generating new values for all genes. The exploration ability of PSO notably diminishes with the increasing of generation and PSO loses the good balance between exploration and exploitation. The low performance of SA results from its limited optimization ability, in SA only one movement is used to produce neighborhood solutions.

6 Conclusions

Some stochastic processing conditions and constraints make SJSSP be difficult. In this paper, SJSSP with exponential processing time, *non-resumable* jobs and random breakdown is considered and a random key scheduling algorithm is proposed. The objective is to minimize the stochastic makespan. A novel random key representation is used to represent the schedule of the problem and a discrete event-driven decoding method is applied to build the schedule. Probability stochastic order and the addition operation of exponential random variables are also used to calculate the objective value. Our GA is tested on 22 benchmark problems and is compared with SA and PSO. The comparative results show the promising advantage of our GA on stochastic scheduling.

A number of meta-heuristics have been successfully applied to JSSP and only few of them are used to solve SJSSP. SJSSP still attracts less attention and the great efforts must be made to deal with SJSSP. In the near future, we will focus on the application of some meta-heuristics including evolutionary algorithm and PSO to SJSSP and the consideration of some actual processing conditions such as flexible process plan, batch, and breakdown with consideration of *resumable* jobs.

Acknowledgment This paper is supported by the China National Science Foundation (70901064).

Appendix

Table 4 Machine breakdown and repair data (the first is λ_{uk1} and the second is λ_{dkv} in each grid)

problem	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
ABZ5	0.001	0.0016	0.00208	0.00165	0.0018	0.00125	0.0022	0.00119	0.00209	0.0013
	0.21	0.20	0.25	0.19	0.10	0.24	0.20	0.16	0.19	0.12
ABZ6	0.0012	0.0016	0.00108	0.0017	0.0018	0.00205	0.0022	0.0019	0.00129	0.0013
	0.10	0.24	0.20	0.21	0.20	0.25	0.19	0.12	0.19	0.16
ORB1	0.0025	0.0017	0.0015	0.0018	0.0021	0.0031	0.0024	0.0018	0.0020	0.00095
	0.10	0.14	0.20	0.18	0.10	0.15	0.19	0.12	0.19	0.16
ORB2	0.0013	0.0019	0.0016	0.0029	0.0024	0.0015	0.0010	0.0008	0.0025	0.00106
	0.15	0.12	0.25	0.10	0.20	0.24	0.12	0.08	0.125	0.09
ORB3	0.0015	0.0021	0.00116	0.0019	0.0023	0.0017	0.00125	0.0018	0.0015	0.00206
	0.18	0.22	0.20	0.16	0.23	0.14	0.25	0.28	0.15	0.16
ORB4	0.002	0.0016	0.00205	0.0017	0.0020	0.00123	0.0030	0.00118	0.0015	0.0013
	0.28	0.32	0.25	0.11	0.33	0.24	0.20	0.18	0.17	0.20
ORB5	0.0022	0.0017	0.0025	0.0017	0.0021	0.0011	0.00206	0.0020	0.00109	0.0023
	0.20	027	0.25	0.21	0.13	0.14	0.25	0.18	0.09	0.20
ORB6	0.00107	0.0013	0.0016	0.0013	0.0014	0.0016	0.0015	0.0013	0.00119	0.00175
	0.16	0.21	0.25	0.11	0.13	0.115	0.28	0.18	0.19	0.10
ORB7	0.0025	0.0026	0.0032	0.00365	0.00271	0.0035	0.0027	0.0023	0.0027	0.0031
	0.19	0.21	0.115	0.205	0.11	0.16	0.28	0.18	0.109	0.10
ORB8	0.00125	0.0017	0.00135	0.00115	0.00105	0.0023	0.00185	0.00127	0.00167	0.0018
	0.19	0.16	0.21	0.215	0.28	0.18	0.205	0.21	0.189	0.14
ORB9	0.00195	0.0013	0.00205	0.00135	0.0014	0.0013	0.00145	0.00107	0.00147	0.0015
	0.19	0.26	0.21	0.205	0.21	0.215	0.28	0.18	0.189	0.24
ORB10	0.00115	0.0013	0.00105	0.00135	0.0014	0.0013	0.00145	0.00157	0.00147	0.0015

Table 4 (continued)

problem	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	0.19	0.26	0.21	0.205	0.21	0.215	0.28	0.18	0.189	0.24
LA16	0.002	0.0016	0.00208	0.0017	0.0018	0.00125	0.0022	0.0019	0.00129	0.00133
	0.10	0.24	0.20	0.21	0.20	0.25	0.19	0.12	0.19	0.16
LA17	0.0025	0.00108	0.0025	0.00141	0.00203	0.0018	0.0020	0.0010	0.00124	0.0022
	0.14	0.16	0.13	0.17	0.13	0.25	0.19	0.19	0.22	0.20
LA18	0.00125	0.00118	0.00135	0.0019	0.00203	0.0018	0.0025	0.0019	0.0020	0.0024
	0.14	0.13	0.25	0.19	0.19	0.16	0.13	0.17	0.28	0.25
LA19	0.0025	0.00137	0.0018	0.0020	0.0022	0.0030	0.0023	0.0019	0.0021	0.0023
	0.24	0.13	0.25	0.19	0.13	0.14	0.21	0.16	0.28	0.25
LA20	0.0017	0.0017	0.00098	0.0020	0.0011	0.0018	0.0013	0.0021	0.0012	0.0015
	0.24	0.15	0.16	0.25	0.13	0.14	0.19	0.21	0.29	0.22
LA21	0.00119	0.0017	0.00203	0.0016	0.00146	0.0023	0.00187	0.0017	0.0022	0.00125
	0.14	0.18	0.16	0.25	0.11	0.14	0.19	0.29	0.21	0.22
LA22	0.0018	0.0017	0.0016	0.00202	0.0020	0.00119	0.00146	0.0022	0.00127	0.0015
	0.14	0.25	0.13	0.14	0.19	0.27	0.21	0.12	0.18	0.26
LA23	0.00102	0.0015	0.00126	0.0014	0.0020	0.0013	0.00146	0.0018	0.00202	0.0015
	0.14	0.20	0.13	0.24	0.19	0.21	0.21	0.12	0.15	0.26
LA24	0.0017	0.00117	0.0021	0.0014	0.0020	0.0023	0.00176	0.0018	0.0017	0.00205
	0.14	0.20	0.13	0.25	0.14	0.21	0.22	0.12	0.18	0.26
LA25	0.0020	0.00123	0.0017	0.0018	0.00114	0.0018	0.0016	0.0025	0.00119	0.0012
	0.19	0.20	0.25	0.17	0.14	0.11	0.22	0.14	0.18	0.26

For all problems, $\lambda_{uk\upsilon} = 0.4^{\upsilon-1}\lambda_{uk1}, \upsilon \geq 1$.

References

1. Li W, Cao J (1995) Stochastic scheduling on a single machine subject to multiple breakdown according to different probabilities. Oper Res Lett 18:81–91
2. Kasap N, Aytug H, Paul A (2006) Minimizing makespan on a single machine subject to random breakdowns. Oper Res Lett 34:29–36
3. Allahverdi A (1995) Two-stage production scheduling with separate setup times and stochastic breakdowns. J Oper Res Soc 46:896–904
4. Allahverdi A (1996) Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness. Comput Oper Res 23:909–916
5. Allahverdi A, Mittenthal J (1994) Two-machine ordered flowshop scheduling under breakdown. Math Comput Modell 20:9–17
6. Allahverid A, Mittenthal J (1998) Dual criteria scheduling on a two-machine flowshop subject to random breakdowns. Int Trans Oper Res 5:317–324
7. Alcaide D, Rodriguez-Gonzalez A, Sicilia J (2002) An approach to solve the minimum expected makespan flow-shop problem subject to breakdowns. Eur J Oper Res 140:384–398
8. Alcaide D, Rodriguez-Gonzalez A, Sicilia J (2006) A heuristic approach to minimize expected makespan in open shops subject to stochastic processing times and failures. Int J Flex Manuf Syst 17:201–226
9. Luh PB, Cheng D, Thakur LS (1999) An effective approach for job shop scheduling with uncertain processing requirements. IEEE Trans Robot Autom 15:328–339
10. Singer M (2000) Forecasting policies for scheduling a stochastic due date job shop. Int J Prod Res 38(15):3623–3637
11. Ginzburg DG, Gonik A (2002) Optimal job-shop scheduling with random operations and cost objectives. Int J Prod Econ 76:147–157
12. 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
13. Lei DM, Xiong HJ (2007) An efficient evolutionary algorithm for multi-objective stochastic job shop scheduling. In: Proceedings of International Conference on Machine Learning and Cybernetics. pp. 867–872
14. Lei DM, Xiong HJ (2008) Job shop scheduling with stochastic processing time through genetic algorithm. In: Proceedings of International Conference on Machine Learning and Cybernetics, Kunming, China. pp. 941–946
15. Gu JW, Gu XS, Gu MZ (2009) A novel parallel quantum genetic algorithm for stochastic job shop scheduling. J Math Anal Appl 355:63–81
16. Gu JW, Gu MZ, Cao CW, Gu XS (2010) A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. Comput Oper Res 37:927–937
17. Cai XQ, Wang LM, Zhou X (2007) Single-machine scheduling to stochastically minimize maximum lateness. J Sched 10:293–301
18. Lei DM (2010) A genetic algorithm for flexible job shop scheduling with fuzzy processing time. Int J Prod Res 48:2995–3013
19. Gao J, Gen M, Sun LY (2006) Scheduling jobs and maintenance in flexible job shop with a hybrid genetic algorithm. J Intell Manuf 17:493–507

20. Lei DM, Wu ZM (2006) Crowding-measure-based multi-objective evolutionary algorithm for Job shop scheduling. Int J Adv Manuf Technol 30(1–2):112–117
21. 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, San Mateo
22. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
23. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, WA, Australia. pp 1942–1948
24. Lei DM (2008) A Pareto archive particle swarm optimization for multi-objective job shop scheduling. Comput Ind Eng 54:960–971