Int J Adv Manuf Technol (2013) 66:501–514
DOI 10.1007/s00170-012-4344-4

ORIGINAL ARTICLE

Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies

Wei He · Di-hua Sun

Received: 8 March 2012 / Accepted: 22 June 2012 / Published online: 6 July 2012

Abstract In this paper, flexible job shop scheduling problem with machine breakdown is of concern. Considering that there is a limitation in improving robust and stable performance of rescheduling with a single strategy, an approach with multi-strategies is proposed to make the scheduling more robust and stable. First, in prescheduling, a new idle time insertion strategy is put forward. In this new policy, idle time equal to repair time is inserted into an appropriate position of each machine according to the machine's breakdown nature. Second, route changing strategy combined with right-shift policy is proposed to keep the rescheduling as stable and robust as possible. In this policy, whether to right shift or route change is dependent on the cost of archiving robustness and stability. Based on the two strategies, new algorithms dealing with idle time insertion, right-shift scheduling, and route changing scheduling are designed. The computational results show the effectiveness of the new strategies and new algorithms compared with other strategies.

Keywords Flexible job shop scheduling · Machine breakdown · Idle time insertion · Right shift · Route changing · Rescheduling

1 Introduction

Flexible job shop problem (FJSSP) has been investigated for many years [1–4]. In many researches, parameters such as processing time and time of job arrival are assumed to be predetermined, and the resources are always available. But, in practice, there are many interruptions during the manufacturing process such as random processing time, random machine breakdown, random job arrivals, or job cancelations. Thus, the algorithms for determined scheduling cannot be applied in an uncertain environment. Among all the uncertain events, machine breakdown is one of the most popular disruptions in flexible job shop scheduling. In this article, FJSSP with machine breakdown is focused on.

When a machine breakdown takes place, a repair procedure starts, and the operation being processed on the machine is suspended. When the machine repair is finished, the operation suspended has two ways to process. If the operation continues its processing, it is called resumable processing; otherwise, the operation starts its processing from its beginning, which is called nonresumable processing [5]. In this research, all jobs affected by breakdown are considered to be nonresumable.

Due to the complexity of FJSSP, it has been attracted more interest from researchers. The approaches to solve FJSSP with random breakdown are divided into two types of solutions: predictive scheduling and completely reactive scheduling [6]. In predictive scheduling, a prescheduling considering or without considering breakdowns is generated and implemented until machine breakdown occurs, and a rescheduling procedure will be launched to react to the machine breakdown. In completely reactive scheduling, the scheduling will generate a response to breakdown in real time. This completely reactive scheduling in real time will result in great cost and low performance, so the prescheduling–rescheduling approach is the most popular approach adopted to deal with machine interruptions.

In the prescheduling–rescheduling approach, some strategies such as right shift [7], routing changing [8],

W. He (✉) · D.-h. Sun
School of Automation, Chongqing University,
Chongqing 400044, China
e-mail: rabbit_hw@163.com

W. He · D.-h. Sun
Key Laboratory of Dependable Service Computing in Cyber Physical Society of Ministry of Education,
Chongqing 400044, China

or full rescheduling [9] are applied to adjust the initial scheduling when breakdown happens. The optimal object that is mostly adopted is makespan, but in recent research, robust and stable performances are paid more attention [10, 11]. According to the definition of Selcuk Goren and Ihsan Sabuncuoglu [12], a schedule whose performance does not deteriorate in the face of disruptions is called robustness; a schedule whose realization does not deviate from the original schedule in the face of disruptions is called stability.

From the above definitions, the robust and stable performances mean that there are minimum changes to prescheduling. Although a few researches have been focused on the way to maintain robustness and stability simultaneously, it is still a challenge to the problem. In previous researches, single policy is applied to keep robust and stable features, but it is not enough to improve the robust and stable performance of rescheduling. In this paper, multi-strategies and corresponding algorithms are studied to obtain the better robust and stable performances. The contributions of the studies in this paper are as follows: (1) study new idle time insertion policy in prescheduling considering different breakdown probabilities of different machines according to their distributions,(2) study a new strategy combining right shift with route changing when breakdown happens, and (3) study corresponding algorithms of right shift and routing change based on prescheduling.

This paper is organized as follows:Sect. 2 presents the literature review. Section 3 gives the definitions and assumptions of JSSP. Prescheduling strategy is given in Sect. 4. Section 5 presents the policies and approaches of routing changing and right shift based on prescheduling. Experiments are conducted in Sect. 6. Lastly, the conclusion is drawn in Sect. 7.

2 Literature review

FJSSP is an NP-hard problem and heuristic or metaheuristic approaches are employed to optimize an object such as makespan [13, 14]. But in recent years, more attention is paid to FJSSP under uncertainties. Machine breakdown is one of the uncertain factors considered in many literatures.

Scheduling with single machine breakdown is the foundation of studying on job shop with breakdown. S.V. Mehta and R. Uzsoy [15] designed an algorithm in which extra idle times are inserted before operations to be processed. R. O'Donovan, R. Uzsoy and K.N. McKay [16] took the average tardiness of each job as the optimal object and proposed robust scheduling and rescheduling approaches to improve stability.

Based on scheduling with single machine breakdown, many researchers have studied job shop scheduling subject to machine breakdown. Jian Fang and Yugeng Xi [17] present a periodic, event-driven rolling horizon scheduling strategy and a hybrid of genetic algorithms with dispatching rules to solve the problem. Sanjay V. Mehta and Reha M. Uzsoy [18] proposed a predictive schedule into which an idle time is inserted according to breakdown and repair distributions as well as the structure of the predictive schedule. Oliver Holthaus [19] investigated simulation-based analysis of dispatching rules in job shop scheduling taking into account machine breakdown. Results revealed that different levels of breakdown parameters have impact on performance of scheduling rules. E. Kutanoglu and I. Sabuncuoglu [8] studied reactive scheduling policies based on rerouting of jobs and gave the impact of the rerouting strategy to the jobs affected by the machine breakdown. Velusamy Subramaniam and Amritpal Singh Raheja [20] studied the typical job shop disruptions. They decomposed the repair processes into four generic repair steps using the proposed modified affected operation rescheduling heuristic method. N.A. Masruroh and K. L. Poh [21] proposed a method to manage the shop floor uncertainty using the Bayesian Network. The proposed method considered the interaction of the scheduling with other factors in the system and extended it into influence diagram to evaluate the need of rescheduling. Yongmei Hu et al. [22] proposed a rolling schedule based on singular rough sets to solve the problem. In the case of machine failure, the due date of the jobs is to be changed; the urgent job and the reselection and rescheduling of the jobs in the rolling window are executed once more. Haruhiko Suwa and Hiroaki Sandoh [23] proposed a new when-to-schedule policy in reactive scheduling, which considered time of schedule revision based on the concept of a control limit policy. Under the proposed policy, schedule revision is carried out based on a cumulative task delay which can be a measure to determine suitable timing of schedule revision. M. Zandieh and M.A. Adibi [24] proposed a scheduling method based on variable neighborhood search(VNS) to solve a dynamic job shop scheduling problem that considers random machine breakdown. In their method, an event-driven policy is selected. To enhance the efficiency and effectiveness of the scheduling method, an artificial neural network with a back propagation error learning algorithm is used to update parameters of the VNS at any rescheduling point according to the problem condition. H. Kamoun and C. Sriskandarajah [7] proposed right-shift rules to handle machine breakdown. Deming Lei [5] proposed an efficient genetic algorithm for the problem with exponential time and nonresumable jobs. In the proposed genetic algorithm (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 to deal with machine breakdown.

Recently, robust and stable scheduling in flexible job shop with machine breakdown has been intriguing many researchers. Mikkel T. Jensen [25] defined robustness measure and investigated its properties. Then, GA is applied to find robust and flexible solutions with low makespan. Selcuk Goren and Ihsan Sabuncuoglu [12] defined two robustness measures and three stable measures. Then, a dominance rule and two lower bounds for one of the robustness measures are developed and used in a branch-and-bound algorithm to solve the problem exactly. A beam search heuristic is also proposed to solve large problems for all five measures. Nasr Al-Hinai and T.Y. El Mekkawy [4] defined a number of bi-objective measures combining the robustness and stability of the predicted scheduling. Consequently, a two-stage Hybrid Genetic Algorithm is proposed to generate the predictive scheduling. S.M. Kamrul Hasan, Ruhul Sarker, and Daryl Essam [26] proposed an improved local search technique, shifted gap reduction (SGR), which improves the performance of GAs when solving relatively difficult test problems. The new algorithm for JSSPs with machine unavailability or breakdown is modified considering two scenarios of machine unavailability.

From the literatures reviewed above, the approaches can be classified into two types: static scheduling and dynamic scheduling. In static scheduling, the machine which will break down and the time when the machine will break down are assumed to be known in advance. But, it does not fit practice because no one can exactly know the information of the breakdown. Dynamic scheduling means when breakdown takes place, adjustments or rescheduling will be executed in terms of some strategies based on prescheduling. The strategies which are mostly applied in the adjustments or rescheduling include: right-shift scheduling (RSS), routing changing scheduling, and full rescheduling. Although there many researches on scheduling with machine breakdown, researches on flexible job shop scheduling subject to machine breakdown are still relatively limited. In addition, there are some shortage in the existing studies on the scheduling of FJSSP with machine breakdown: (1) Preschedule either takes no account of the disruption of the machine or inserts idle times before all of the operations processed on machines to accommodate machine failure. Both of the policies can degrade the performance of robustness or stability. (2) Most researches take a single strategy to deal with breakdown, and the performance of robustness or stability cannot be easily guaranteed. (3) It takes a long time to obtain the robust and stable features in full rescheduling, which is always not accepted in practice.

In this paper, a new approach and a new policy will be put up to obstacle the aforementioned shortages. In the prescheduling stage, a new idle insertion policy is presented to assure the robustness of the rescheduling performance if machine breakdown takes place. In the rescheduling stage, routing change combined with right-shift strategy is put up to assure stability. This will be illustrated in detail in the following sections.

3 Problem definitions and assumptions

FJSSP with machine breakdown is more complex than that in determination conditions. The critical technologies to solve the problem are: (1) assignment of operation to alternative machines, (2) sequencing operations in each machine, and (3) rescheduling policy when a disruption occurs. In this article, discussion about FJSSP with machine breakdown in detail is based on the following definitions and assumptions:

- There are m machines indexed by M_k (k=1, 2,……, m).
- There are n jobs to be processed indexed by i (i=1, 2, ……, n).
- Each job has m operations is presented by $O_{ij}(i = 1, 2, ..., n, j = 1, 2, ..., m)$.
- The processing time for operation j of job i on machine k is denoted as t_{ijk}.
- A subset of machines for operation O_{ij} is presented by $M_{kij} \subseteq \{M_1, M_2, ..., M_m\}$.
- p_{ijk} denotes the process time of the jth process of job i on machine k.
- s_{ijk} denotes the start time of O_{ij} on machine M_k.
- e_{ijk} denotes the end time of O_{ij} on machine M_k.
- Assume each machine failure is subject to exponential distribution and the probability of the ith machine failure at time t is denoted as $P_{M_i}(t)$.
- t_{BD}^{k} denotes the breakdown time of machine M_k.
- T_{AV}^{k} is defined as the average normal work time of machine k, which means average intervals of the successive failure time of machine i.
- Rt_j^k is defined as the jth repair time interval of the kth machine when breakdown happens. It is reasonable that the repair time for the ith machine is determined.
- $rt_j^k$is defined as the$j$th time at which the repair process is finished on machine$k$, and bt_j^k is defined as the time at witch the$j$th breakdown occurred on machine M_k.
- TH_k denotes the probability threshold of machine M_k.

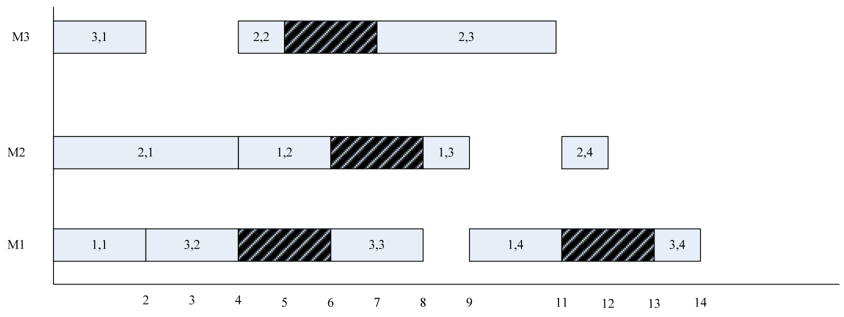

Fig. 1 A feasible scheduling with insertion of idle interval according to breakdown distribution

There are several assumptions and constraints in FJSSP, such as:

- Each machine can process at most one operation at any time.
- Each operation can be processed only at one machine at a time.
- Operations of all the jobs must be processed in a given order.
- Setup time and transport time of each job are neglected.

4 Prescheduling strategy

The approach adopted usually for FJSSP with machine breakdown is prescheduling and rescheduling. In most literatures, two methods are accepted in prescheduling. One is swarm intelligence algorithm (i.e., GA) which is applied to generate prescheduling without considering any disruptions [27]. Another one is generating prescheduling with the insertion of a certain length of idle time into prescheduling to accommodate the time of machine failure [16]. The first one does not consider the potential breakdown in the future, and its robustness and stability are completely depending on rescheduling. The second one will deteriorate the performance of prescheduling, and it is unnecessary to insert idle time for each operation. In this paper, a new strategy compromising between the two mentioned above is proposed.

4.1 Machine breakdown distribution

Although the time of machine failure is not known in advance, its probability distribution can be known according to historical data. Assume the probability of machine breakdown is subjected to exponential distribution, and it can be illustrated as follows:

$$P_{M_i}(t) = \begin{cases} 0 & t < 0 \text{ or } t = rt_j^k \\ 1 - e^{\frac{t - rt_j^k - \sum_{h=1}^{H_j} I_h}{T_{\mathrm{AV}}^k}} & t \geq 0 \text{ and } rt_j^k < t < \mathrm{bt}_{j+1}^k \end{cases} \qquad j = 1, 2, 3 \ldots\ldots \tag{1}$$

where I_h is the hth idle time in $\left[rt_j^k, t\right]$. H_j is total number of idle times in $\left[rt_j^k, t\right]$. During the period of job processing, the machine may be break down several times; when $P_{M_i}(t)$ is greater than a threshold determined by an expert, machine breakdown may happen and a repair procedure is started. After repair procedure is finished, $P_{M_i}(t)$ of the machine comes back to 0 at $t = rt_{j+1}^k$. It is noted that when $P_{M_i}(t)$ is greater than a threshold, the machine has the probability of breaking down, but not always in failure.

4.2 Handling potential breakdown

Prescheduling is the first stage to handle machine breakdowns. In prescheduling, it is assumed that all the machines are available and their probability of breakdown is zero. Then, prescheduling is generated by GA algorithm and implemented in the shop floor.

Based on definition in Sect. 4.1, machine breakdown occurs subjected to exponential distribution; the time at which breakdown happens is when the probability of machine breakdown surpasses the threshold. Figure 1 shows the job shop scheduling and idle time insertion to handle breakdown; the shadow block on each machine stands for idle time interval (time length is equal to repair time). If the average normal work time T_{AV}^k is 4 and the threshold is set to 0.6 on machine 1, the probability of breakdown can be computed as follows: in time 2, $P_{M1} = 1 - e^{-\frac{2}{4}} = 0.3935$, in time 4, $P_{M1} = 1 - e^{-\frac{4}{4}} = 0.6321$, which is greater than the threshold of machine 1 (0.6); then, an idle time equal to fix time (=2) is inserted. In time 6, the probability of machine 1 is taken as 0 for granted. In time interval [6, 11], the actual working time of machine 1 is 4, and then, another idle time is inserted according to Formula (1).

4.3 Prescheduling generation

In this paper, GA which is taken from [4] with objective makespan is applied to produce active prescheduling. The chromosomes in population are divided into two parts: machine part and operation part. Refer to [4] for the coding procedure in detail.

The decoding procedure is a little different from that in [4] in that machine failures have to be take account of. The decoding algorithm which is modified to produce active scheduling taking consideration of machine failure is as follows:

Algorithm 1 Decode (decoding a chromosome to an active schedule)

```
1. Initialize Gantt chart and parameters such as $P_{M_k}$, $t_k^j$, $TH_k$ ($t_k^j$ is the total workload in $[rt_j^k, t]$ of machine $k$).
2. For each gene of each chromosome extracted from left to right do
        Get the current operation $o_{i,j}$
        Get the machine $M_k$ which processes $o_{i,j}$ from the first gene's digit and obtain its processing time $p_{i,j,k}$
        IF $o_{i,j}$ is the first operation Then
            Set $t0$ to 0
        ELSE
            Set $t0$ to be stop time of the predecessor operation $o_{i,j-1}$
        End IF
        IF $M_k$ did not process any operation Then
            Set $t1$ to 0
        ELSE IF there exists idle time in immediate predecessor of $o_{i,j}$
            Set $t1$ to be the stop time of idle time on $M_k$
        ELSE
            Set $t1$ to be the stop time of the last operation on $M_k$
        End IF
        IF $t1 <= t0$ Then
            Add to $M_k$ starting at $t0$
        ELSE IF there exists time interval between $t0$ and $t1$ (time interval except idle time between two consecutive operations on $M_k$) greater than $p_{i,j,k}$ Then
            Add $o_{i,j}$ to $M_k$ starting at the end of the finished processing time of the operation to the left
        ELSE
            Add $o_{i,j}$ to $M_k$ starting at $t1$
        End IF
        $t_k^j = t_k^j + p_{ijk}$
        Compute $P_{M_k}$ for each machine.
        IF $P_{M_i}(t)$ is greater than $TH_k$ Then
            Insert idle time after the last operation of machine $M_k$ .and set $P_{M_i}(t)$ to zero..
        End IF
    End For
```

Selection, crossing, and mutation are the basic operators of GA. Tournament selection is applied to form the mating pool and mating sub-pool. The chromosomes in the mating pool and mating sub-pool are selected to produce child chromosomes by crossing over operation. In this paper, Precedence Preserving Order-Based Crossover (POX) is introduced by Kacem et al. [28] to build feasible child chromosomes. To preserve the diversity of the population, the mutation operator is applied with position-based mutation and machine-based mutation. Details of these operators can be referred to in [4, 29].

5 Rescheduling strategies and algorithms

5.1 Robust and stable measures of rescheduling

Robustness and stability are important features of the scheduling implemented in the job floor. They are referred to as the ability to keep good performance under dynamic or uncertain conditions. It is important to develop a robust and stable scheduling to reduce influence of random disruptions. In this paper, the robust measure is adopted from [4] which is defined as follows:

$$\mathrm{RM} = \frac{\mathrm{MS\_R} - \mathrm{MS\_P}}{\mathrm{MS\_P}} \times 100\% \qquad (2)$$

where MS_R is real makespan of scheduling and MS_P is makespan of prescheduling. RM is also called relative robustness.

The stable measures of scheduling are taken from [4], but in this paper, not all of the operations are computed because when breakdown happens, parts of operations have been completed, and their real end time is equal to the predicted time. So, only the remaining operations including unfinished and being-processed operations are computed. Based on the above point of view, the stable measures can be defined as follows:

$$\mathrm{SM} = \min \frac{\sum_{i=1}^{n'} \sum_{j=1}^{q'} \left| \mathrm{CO}_{ijp} - \mathrm{CO}_{ijR} \right|}{\sum_{i=1}^{n} O_i} \qquad (3)$$

where n' is the number of unfinished and being-processed jobs. n is the total number of all operations. q' is the number of unfinished and being-processed operations of job i. CO_{ijp} is the predicted completion time of operation j of job i, in prescheduling. CO_{ijR} is realized completion time of operation j of job i. The completion time generated by rescheduling can be taken as realized completion time. When the next breakdown happens, the current rescheduling can be looked as prescheduling, and new rescheduling is generated, by parity of reasoning.

To address the algorithm in a better way, two comprehensive measures called average relative robust measure (ARRM) and average stability measure (ASM) are proposed which are defined as follows:

$$\mathrm{ARRM} = \frac{\sum_{i=1}^{\mathrm{NBM}} \sum_{j=1}^{\mathrm{NBK}} \frac{\mathrm{MS\_P}_i^j - \mathrm{MS\_R}_i^j}{\mathrm{MS\_P}_i^j} \times 100\%}{\mathrm{NBM} \times \mathrm{NBK}} \qquad (4)$$

$$\mathrm{ASM} = \frac{\sum_{i=1}^{\mathrm{NBM}} \sum_{j=1}^{\mathrm{NBK}} \mathrm{SM}_j^i}{\mathrm{NBM} \times \mathrm{NBK}} \qquad (5)$$

where NBK is the number of breakdowns taking place in one machine and NBM is the number of machines for breakdown.

5.2 Right-shift scheduling

When prescheduling is implemented in the job floor, the operations are processed as prescheduling generated. If there are not any breakdowns that occurred, the prescheduling is the realized scheduling; otherwise, something must be done to deal with the breakdowns. Right shift is one of the most popular policies to handle breakdown. The first step is to obtain the operation that is directly affected by machine breakdown, for example, an operation being processed or to be processed when the machine fails is called the operation that is directly affected by machine breakdown. Then, if breakdown happens, the following inequation is checked for $o_{i,j}$ on machine k:

$$t_{\mathrm{BD}}^k + \mathrm{Rt}_k^j + p_{ijk} \le \min\left(s_{i(j+1)k'}, s_{\mathrm{suc},k}\right) \qquad (6)$$

In formula (6), $s_{\mathrm{suc},k}$ is the start time of the succeeding operation to $o_{i,j}$ on machine M_k. If formula (6) is held, the operation $o_{i,j}$ being affected can shift to the right without affecting other operations and start at time $t_{\mathrm{BD}}^k + \mathrm{Rt}_k^j$; otherwise, other operations will also be right shifted.

In previous researches, many right-shift policies have been proposed. For example, A. Yahyaoui et al. [30] proposed a new shifting method for job shop with machine unavailability. In their research, six rules are put forward to determine when and how to right shift by determining the different starting times for each operation in order to minimize the makespan. Shifting one operation to the right will affect not only the succeeded operations of the same job but also the operations on the same machine (see Fig. 2). In this paper, new a right-shift rescheduling algorithm based on binary tree is proposed.

In Fig. 2, the left child of the binary tree stores the intermediate next operation of the same job affected by machine failure, and the right child of the binary tree stores the next operation arranged on the same machine. When a breakdown

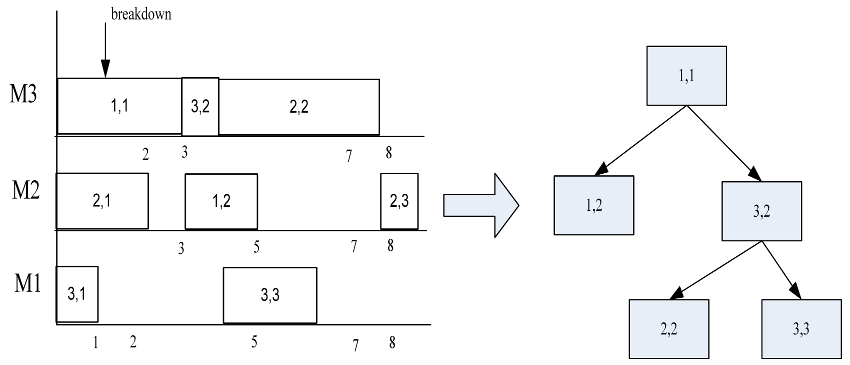

Fig. 2 Binary tree of operations to be shifted

happens, the operation affected is moved to the right for a period of time until there exists an idle time to absorb the breakdown. The binary tree can be built in recursion way. The right-shift algorithm can be illustrated as follows:

Algorithm 2 Right_shift_schedule (t_breakdown, machine)

1. Initialize the binary tree: B_tree=∅.
2. Determine the position *a_pos* of machine in which the breakdown occurs.
3. Determine the operations affected by breakdown.
4. Root=1.
5. Right_shift_tree (B_tree, root, machine, a_pos, t_breakdown).

Algorithm 3 right_shift_tree (B_tree, root, machine, a_pos, t)

```
IF B_Tree (root) is empty then
    Record the affected operation in a_pos of the machine to the node of the trees;
    Move the operation right to $t + Rt_j^k$ time period;
    Update and Record start time and end time of affected operation;
    Update $t$ with the end time of the affected operation;
END
//Left Tree
IF the left child is empty then
  IF there exists idle time immediately after the operation in a_pos Then
      IF the end of the operation is smaller than the start time of the next operation on the same
      machine then
          Return;
      ELSE
          Right_Shift_Tree (B_Tree, lChild, machine, a_pos+1, t);
      END
  ELSE IF the operation is the last one on machine then
        Return;
      ELSE
        Right_Shift_Tree (B_Tree, 2*root, machine, a_pos+1, t);
      END
  END
// Right Tree
IF right child is empty
    Obtain the start time denoted as tl of the next operation in sequence to the operation in a_pos
    of machine;
    IF the end of the operation is smaller than tl then
        Return;
    ELSE
        Update t with the end time of the affected operation;
        Right_Shift_Tree (B_Tree, rChild, another machine, pos, t);
    END
END
```

To illustrate algorithm 2 more clearly, the scheduling in Fig. 2 is taken as example. In Fig. 2, the breakdown takes place at time 1 on machine M3; the operation being affected can be built as a binary tree which is shown in the right of Fig. 2. In algorithm 2, the parameter t_breakdown is set to the breakdown time, and the machine is set to the breakdown machine. In Fig. 2, they are referred to as 1 and 3, respectively. If the repair time is 1, the result of algorithm 2 on Fig. 2 is shown in Fig. 3.

In the binary tree of Fig. 2, the numbers in a pair of brackets are start time and end time of operations being affected. For example, in the root of the tree, 1, 1 (2, 5) means the first operation of job 1 is affected and the start time is 2 as well as the end time is 5, respectively, after right-shift rescheduling.

5.3 Route changing scheduling integrated with right shift

Applying only right-shift scheduling sometimes cannot assure robust and stable performances better. Another strategy called route changing is adopted to maintain robust and stable performance. Different from right-shift strategy, route changing means that operations affected directly or indirectly by machine breakdown can shift to another available machine to finish their work. If the condition (7) is satisfied, the operation can move to the destination machine without affecting other operations; otherwise, right shift will also be needed.

$$\max\left(t^{k}_{\mathrm{BD}}, E_{\mathrm{pre},k'}\right) + p_{ijk'} \le \min\left(s_{i(j+1)k''}, s_{\mathrm{suc},k'}\right) \qquad (7)$$

In forumula (7), $E_{\mathrm{pre},k'}$ is the end time of the operation immediately precedent to the operation $o_{i,j}$ processed on machinek'. k'' is the machine that operation $o_{i,j+1}$ is processed on. The rest of the variables are the same as formula (6). The purpose of this strategy is to give enough time to repair the failure machine and keep the robust and stable features.

In flexible job shop scheduling, an operation has alternative machines to process. Once a breakdown takes place, the operations affected have two choices, right shift or route changing according to minimum impact on robustness and stability. E. Kutanoglu and I. Sabuncuoglu [8] proposed four route changing rules and drew the conclusion that the most robust reactive policies that did not show much deterioration are arrival rerouting (AR) and all rerouting (AAR). AAR policy was the best among the tested ones almost in all conditions. But, they did not consider the robustness and stability altogether. In this paper, modification based on above the work of E. Kutanoglu and I. Sabuncuoglu is proposed: route changing combined with right shift. The cost of route changing is adopted from Nasr Al-Hinai and T.Y. El Mekkawy [6] which is expressed as follows:

$$z = \min(\gamma \times \mathrm{RM}) + (1-\gamma)\mathrm{SM} \qquad (8)$$

where γ is a weightiness that belongs to [0, 1]. The alternative that the operation moves to should be selected according to minimum cost depicted by formula (8). If the selection is the breakdown machine, it is equal to right shift; otherwise it is routing changing. The route changing algorithm is as follows:

Algorithm 4 route_change_rescheduling

```
1. Find the time interval which the time breakdown is in.
2. Determine the set of operations affected by breakdown denoted by $S$.
3. For each operation $O_{ij}$ in $S$ from last to first, do:
        Get the alternative machines $M_{kij}$ of $O_{ij}$;
        For each machine $M_i$ in $M_{kij}$ do
            IF $M_i$ is breakdown machine THEN right shifting is executed
            ELSE IF formula (5) is satisfied THEN
                    $O_{ij}$ can be inserted into $M_i$ without right shifting
            ELSE $O_{ij}$ can be inserted into $M_i$ and relative operation execute right shifting.
        End For
    End For
4. Compute the costs of each $O_{ij}$ move to each $M_i$ according to formula (8).
5. Select $M_i$ to be moved to in terms of the minimum cost. If there are more than one alternative that has the same minimum cost, then randomly select one among them.
```

Figure 4 shows the routing changing strategy. (a) Illustrate that formula (7) is not satisfied and relative operations ($o_{1,2}$,$o_{2,3}$) have to right shift;(b) illustrate that formula (7) is satisfied and no operation is right shifted. (c) illustrate that M3 has the minimum cost and only right shift is required.

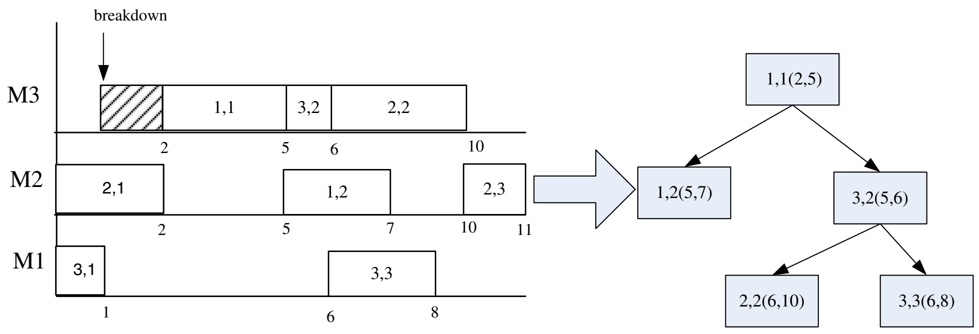

Fig. 3 Binary tree of operations after right shift

6 Results of experiments

In this section, the new strategies and algorithms will be tested. The data for the tests are taken from other literatures due to no benchmarks for FJSSP. Hence, the data of different sizes are taken from following literatures:

1. Ex 1 of 10×7, Ex 2 of 15×10, and Ex 3 of 10×10 are taken from Kacem et al. [31] and Mesghouni et al. [32].
2. Ex 4 of 5×3 is taken from Lee and DiCesare [33], and Ex 5 of 8×8 is taken from Kacem et al. [28]. Examples MK01–MK08, MK11, MK13, and MK15 of different sizes involving 10×6, 15×8, 15×4, 10×15, 20×5, 20×10, 30×5, 30×10, and 30×15 are taken from Brandimarte [34].

For each datum, preschedulings with and without idle time insertion policies are generated by hybridized GA proposed by Al-Hinai Nasr et al. [4]. The size of the population in GA is set to be 100, and the time of iterations is 100. The capacity of the mating pool and sub-mating pool are set to be four, respectively. The probability of crossover operation p_c is 0.7, and the probability of mutation p_m is 0.01.

In order to better test the performance of the algorithm, machine for failure and the time for breaking down are

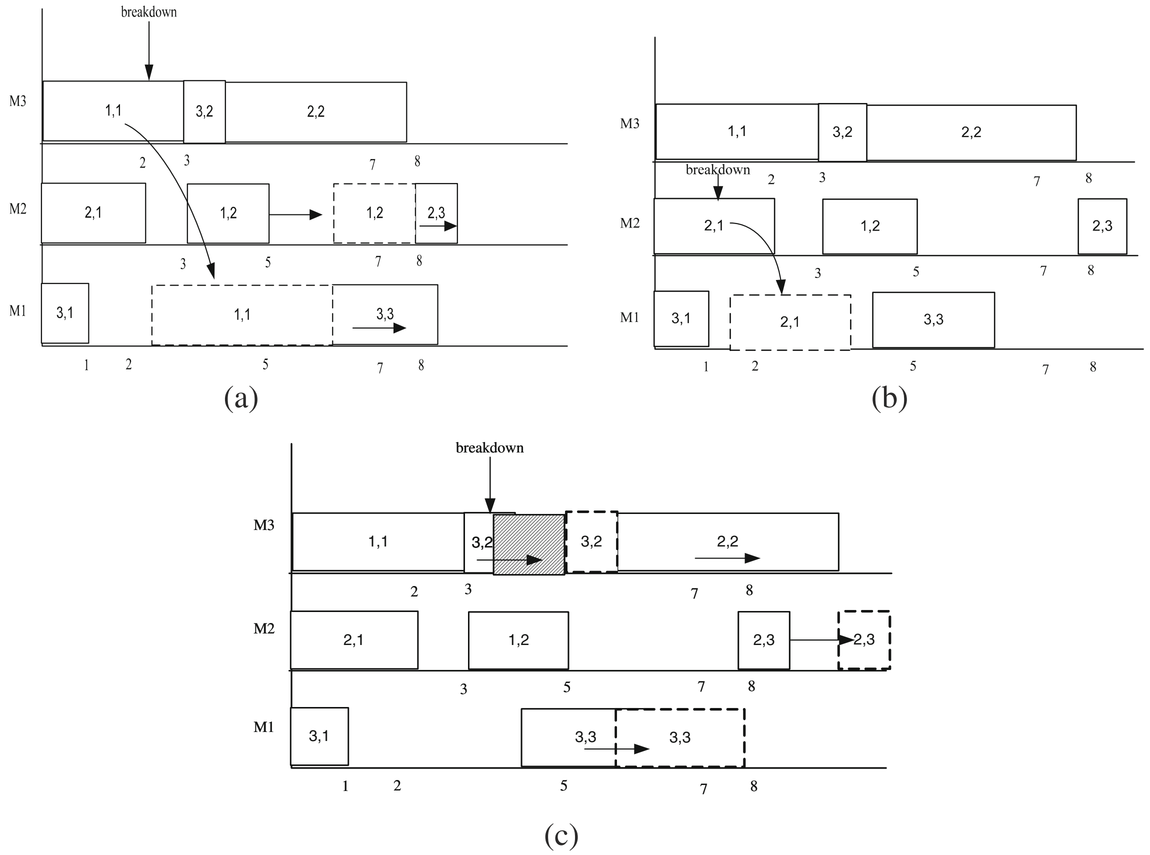

Fig. 4 Routing changing scheduling

Table 1 Computational results of ARRM

Problem	Size	RSS	AAR	SGR	RC+RSS
EX1	10×7	0.064	0.058	0.052	0.049
EX2	15×10	0.237	0.226	0.203	0.194
EX3	10×10	0.186	0.175	0.133	0.129
EX4	5×3	0.118	0.103	0.1000	0.009
EX5	8×8	0.144	0.137	0.142	0.118
MK01	10×6	0.009	0.009	0.009	0.009
MK02	10×6	0.164	0.161	0.157	0.155
MK03	15×8	0.102	0.007	0.112	0.007
MK04	15×8	0.211	0.203	0.198	0.198
MK05	15×4	0.187	0.176	0.167	0.166
MK06	10×15	0.223	0.185	0.189	0.183
MK07	20×5	0.275	0.232	0.218	0.218
MK08	20×10	0.112	0.146	0.117	0.107
MK11	30×5	0.234	0.247	0.225	0.226
MK13	30×10	0.187	0.182	0.171	0.182
MK15	30×20	0.274	0.253	0.246	0.238
Average		0.1704	0.1563	0.1530	0.1359

randomly generated for the above two preschedulings, respectively. The average processing time for each machine is randomly generated in interval [7, 10]. The threshold of breakdown for each machine can be set variably from 0.6 to 0.8. The length of repair time for each machine is assumed to be constant. In this paper, it is set to be 5 time unit. For each problem, 10 machines for breakdown are generated, and for each failure machine, there are 20 breakdown time points covering from early breakdown to late breakdown. That is to say, the parameters NBK and NBM are set to be 20 and 10, respectively. Thus, there are a total of $16 \times 10 \times 20 = 3,200$ test cases.

The strategies proposed in this paper are compared with right-shift scheduling, all rerouting (AAR) and GA with shifted gap reduction (SGR) proposed by S.M. Kamrul Hasan et al. [26]. The RSS, AAR, and SGR are based on prescheduling without idle time insertion while RC+RSS is based on prescheduling with idle time insertion. All experiments are implemented by Matlab R2011(b) and run on Intel (R) Core(TM)2 Duo CPU E7200 at 2.53 GHZ, 2 G RAM computer with windows XP.

6.1 Results of robustness improvement

The experiments test the robust improvement of the four policies. Table 1 shows the results of robustness of different strategies. The data in Table 1 are average relative robust measure ARRM which is defined in formula (4).

Table 2 shows the compared results in terms of average improvement percentage of ARRM for RC+RSS comparing with other three policies. The negative number denotes the improvement of the ARRM whereas the positive number indicates degradation of robustness. Zeros mean that there is neither improvement nor degradation in robust performance. In Table 1 (the same as Table 3 and Table 5), the lower the data, the better.

It can be seen from Tables 1 and 2 that the RC+RSS strategy has better robust performance than RSS and AR in most of the cases. But, it does not always hold in comparison with GA-SGR. RC+RSS results in degradation of

Table 2 Percentage of ARRM to be improved

Problem	Size	RC+RSS vs RSS	RC+RSS vs AAR	RC+RSS vs SGR
EX1	10×7	−0.23438	−0.15517	−0.05769
EX2	15×10	−0.18143	−0.14159	−0.04433
EX3	10×10	−0.30645	−0.26286	−0.03008
EX4	5×3	−0.92373	−0.91262	−0.91
EX5	8×8	−0.18056	−0.13869	−0.16901
MK01	10×6	0	0	0
MK02	10×6	−0.05488	−0.03727	−0.01274
MK03	15×8	−0.93137	0	−0.9375
MK04	15×8	−0.06161	−0.02463	0
MK05	15×4	−0.1123	−0.05682	−0.00599
MK06	10×15	−0.17937	−0.01081	−0.03175
MK07	20×5	−0.20727	−0.06034	0
MK08	20×10	−0.04464	−0.26712	−0.08547
MK11	30×5	−0.03419	−0.08502	0.004444*
MK13	30×10	−0.02674	0	0.064327*
MK15	30×20	−0.13139	−0.05929	−0.03252
Average		−0.20246	−0.13052	−0.11176

Table 3 Computational results of ASM

Problem	Size	RSS	AAR	SGR	RC+RSS
EX1	10×7	0.654	0.651	0.417	0.239
EX2	15×10	0.1982	0.1931	0.1883	0.1785
EX3	10×10	0.1857	0.1862	0.1752	0.15
EX4	5×3	0.5442	0.5511	0.5271	0.5084
EX5	8×8	0.5245	0.5319	0.5258	0.6245
MK01	10×6	0.4343	0.4288	0.4018	0.1457
MK02	10×6	0.44	0.3961	0.4083	0.1275
MK03	15×8	0.223	0.2177	0.1963	0.1555
MK04	15×8	0.9967	0.9773	0.9248	0.925
MK05	15×4	0.4548	0.443	0.3887	0.3238
MK06	10×15	0.1294	0.1186	0.1123	0.2134
MK07	20×5	0.2513	0.2446	0.233	0.2213
MK08	20×10	0.3425	0.2976	0.2661	0.2588
MK11	30×5	0.3032	0.1581	0.1992	0.2237
MK13	30×10	0.2925	0.2279	0.2088	0.2374
MK15	30×20	0.2322	0.2317	0.2191	0.2218
Average		0.387906	0.365919	0.336988	0.297144

robustness by 0.4 and 6 % in mk11 and mk13 problems, respectively. It does not have much improvement compared with GA-SGR. This means RC+RSS may have the same effectiveness with GA-SGR in improving robustness.

6.2 Results of stability improvement

Stability is another important feature to be evaluated in revision of prescheduling. Similar to robustness, average stability measure ASM is computed in terms of formula (5).

Table 3 shows the stability of each problem with four policies, respectively. Table 4 shows the compared results in terms of average improvement percentage of ASM for RC+RSS compared with other three policies.

Tables 3 and 4 indicate that in most problems, stability has got an improvement with RC+RSS policy compared with RSS and AR. But for SGR, improvement is relatively less than the other two compared with RC+RSS. It is noted that the stability obtained in RC+RSS policy is degraded in EX5, MK06, MK11, and MK13 compared with the other three policies.

Table 4 Percentage of ASM to be improved

Problem	Size	RC+RSS vs RSS	RC+RSS vs AAR	RC+RSS vs SGR
EX1	10×7	−0.63456	−0.63287	−0.42686
EX2	15×10	−0.09939	−0.07561	−0.05204
EX3	10×10	−0.19225	−0.19441	−0.14384
EX4	5×3	−0.06578	−0.07748	−0.03548
EX5*	8×8	0.190658	0.174093	0.187714
MK01	10×6	−0.66452	−0.66021	−0.63738
MK02	10×6	−0.71023	−0.67811	−0.68773
MK03	15×8	−0.30269	−0.28571	−0.20785
MK04	15×8	−0.07194	−0.05351	0.000216
MK05	15×4	−0.28804	−0.26907	−0.16697
MK06*	10×15	0.64915	0.799325	0.900267
MK07	20×5	−0.11938	−0.09526	−0.05021
MK08	20×10	−0.24438	−0.13038	−0.02743
MK11*	30×5	−0.2622	0.414927	0.122992
MK13*	30×10	−0.18838	0.041685	0.136973
MK15	30×20	−0.04479	−0.04273	0.012323
Average		−0.23398	−0.18795	−0.11824

*The case that the robustness is not improved

Table 5 Computational results of compound effectiveness of robustness and stability

Problem	Size	RSS	AAR	SGR	RC+RSS
EX1	10×7	0.3	0.2952	0.198	0.125
EX2	15×10	0.22148	0.21284	0.19712	0.1878
EX3	10×10	0.18588	0.17948	0.14988	0.1374
EX4	5×3	0.28848	0.28224	0.27084	0.20876
EX5	8×8	0.2962	0.29496	0.29552	0.3206
MK01	10×6	0.17912	0.17692	0.16612	0.06368
MK02	10×6	0.2744	0.25504	0.25752	0.144
MK03	15×8	0.1504	0.09128	0.14572	0.0664
MK04	15×8	0.52528	0.51272	0.48872	0.4888
MK05	15×4	0.29412	0.2828	0.25568	0.22912
MK06	10×15	0.18556	0.15844	0.15832	0.19516
MK07	20×5	0.26552	0.23704	0.224	0.21932
MK08	20×10	0.2042	0.20664	0.17664	0.16772
MK11	30×5	0.26168	0.21144	0.21468	0.22508
MK13	30×10	0.2292	0.20036	0.18612	0.20416
MK15	30×20	0.25728	0.24448	0.23524	0.23152
Average		0.257425	0.240118	0.226258	0.200908

The reason is that in some cases, the robustness and stability cannot be satisfied at the same time. If robustness is emphasized, the stability may be worse, and vise versa. Hence, the compound effect of robustness and stability is to be of concern.

6.3 Results of compound effectiveness of robustness and stability improvement

The compound effectiveness of robustness and stability reflects the comprehensive performance of rescheduling because sometimes the robustness and stability cannot be improved simultaneously. The average compound effectiveness of robustness and stability can be computed as follows:

$$\mathrm{ARSM} = \sum_{i=1}^{10}\sum_{j=1}^{20}\left(\gamma\ \mathrm{ARM}_i^j + (1-\gamma)\mathrm{ASM}_i^j\right) \qquad (9)$$

In this experiment, γ is set to be 0.6. Tables 5 and 6 show the ARSM and the percentage of improvement of the ARSM, respectively.

Table 6 Percentage of compound effectiveness of robustness and stability to be improved

Problem	Size	RC+RSS vs RSS	RC+RSS vs AAR	RC+RSS vs SGR
EX1	10×7	−0.58333	−0.57656	−0.36869
EX2	15×10	−0.15207	−0.11765	−0.04728
EX3	10×10	−0.26081	−0.23446	−0.08327
EX4	5×3	−0.27634	−0.26035	−0.22921
EX5	8×8	0.082377	0.086927	0.084867
MK01	10×6	−0.64448	−0.64006	−0.61666
MK02	10×6	−0.47522	−0.43538	−0.44082
MK03	15×8	−0.55851	−0.27257	−0.54433
MK04	15×8	−0.06945	−0.04665	0.000164
MK05	15×4	−0.221	−0.18982	−0.10388
MK06	10×15	0.051735	0.23176	0.232693
MK07	20×5	−0.174	−0.07476	−0.02089
MK08	20×10	−0.17865	−0.18835	−0.0505
MK11	30×5	−0.13987	0.06451	0.048444
MK13	30×10	−0.10925	0.018966	0.096927
MK15	30×20	−0.10012	−0.05301	−0.01581
Average		−0.23806	−0.16796	−0.12864

It can be seen from Tables 1–6 that robustness, stability, and comprehensive effectiveness of the two are increased in most of the problems. Compared with RSS and AR policies, the performance has obviously improved with the policy proposed by this paper. But, compared with SGR, the performances of about 25 % of the problems are less than those of SGR.

7 Conclusions

FJSSP with machine breakdown has attracted more interest these years, and numerous researches have been done on the problem; it is still a challenge for researchers to solve the problem. The difficulty lies in how to fill in the gap between theory and reality. In this paper, new idle time insertion strategy and routing changing combined with right-shift strategy are proposed to obtain robust and stable performance. In addition, corresponding algorithms based on the above strategies are also put forward. Experiments show that applying integration of multiple strategies result in better performance than applying a single strategy. Because the performance of the approach is near to GA-SGR, future research will focus on how integrate SGR policy into routing changing and right-shift policies to make rescheduling more robust and stable.

Acknowledgments The authors would like to thank the editor and anonymous referees for their valuable comments. This research is reported by the National High-Tech R&D Program, China, under grant number 2006AA04A124 and the 3rd Stage Phase of “211 Project,” China, under grant number S-09108.

References

1. Rahmati Seyed Habib A, Zandieh M (2012) A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. Int J Adv Manuf Technol 58(9–12):1115–1129
2. Amir R, Saeed M (2011) Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach. Int J Adv Manuf Technol 54(9–12):1091–1107
3. Moradi E, Fatemi Ghomi SMT, Zandieh M (2010) An efficient architecture for scheduling flexible job-shop with machine availability constraints. Int J Adv Manuf Technol 51(1–4):325–339
4. Al-Hinai N, ElMekkawy TY (2011) An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem. Flex Serv Manuf J 23(1):64–85
5. Lei D (2011) Scheduling stochastic job shop subject to random breakdown to minimize makespan. Int J Adv Manuf Technol 55 (9–12):1183–1192
6. Al-Hinai N, ElMekkawy TY (2011) Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. Int J Prod Econ 132(2):279–291
7. Kamoun H, Sriskandarajah C (1993) The complexity of scheduling jobs in repetitive manufacturing systems. Eur J Oper Res 70 (3):350–364
8. Kutanoglu E, Sabuncuoglu I (2001) Routing-based reactive scheduling policies for machine failures in dynamic job shops. Int J Prod Res 39(14):3141–3158
9. Cowling PI, Johansson M (2002) Using real time information for effective dynamic scheduling. Eur J Oper Res 139(2):230–244
10. Shafaei R, Brunn P (1999) Workshop scheduling using practical (inaccurate) data—part 2: an investigation of the robustness of scheduling rules in a dynamic and stochastic environment. Int J Prod Res 37(18):4105–4117
11. Wu SD, Storer RH, Chang PC (1993) One-machine rescheduling heuristics with efficiency and stability as criteria. Comput Oper Res 20(1):1–14
12. Goren S, Sabuncuoglu I (2009) Optimization of schedule robustness and stability under random machine breakdowns and processing time variability. IIE Trans 42(3):203–220
13. Alvarez-Valdes R, Fuertes A, Tamarit JM (2005) A heuristic to schedule flexible job-shop in a glass factory. Eur J Oper Res 165 (2):525–534
14. Wang S, Yu J (2010) An effective heuristic for flexible job-shop scheduling problem with maintenance activities. Comput Ind Eng 59(3):436–447
15. Mehta SV, Uzsoy R (1999) Predictable scheduling of a single machine subject to breakdowns. Int J Comput Int Manuf 12 (1):15–38
16. O'Donovan R, Uzsoy R, McKay KN (1999) Predictable scheduling of a single machine with breakdowns and sensitive jobs. Int J Prod Res 37(18):4217–4233
17. Fang J, Xi Y (1997) A rolling horizon job shop rescheduling strategy in the dynamic environment. Int J Adv Manuf Technol 13(3):227–232
18. Mehta SV, Uzsoy RM (1998) Predictable scheduling of a job shop subject to breakdowns. IEEE Trans Robot Aotom 14(3):365–378
19. Holthaus O (1999) Scheduling in job shops with machine breakdowns and experimental study. Comput Ind Eng 36(1):137–162
20. Subramaniam V, Raheja AS (2003) mAOR: a heuristic-based reactive repair mechanism for job shop schedules. Int J Adv Manuf Technol 22(9–10):669–680
21. NA Masruroh, KL Poh (2007) A Bayesian network approach to job-shop rescheduling. International Conference on Industrial Engineering and Engineering Management IEEM : 1098–1102
22. Yongmei Hu, Fu Yanan, Jia Lei (2007) An algorithm of job shop rolling scheduling based on singular rough sets. Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Vol 2, Proceedings: 221–225
23. Suwa HS (2007) Capability of cumulative delay based reactive scheduling for job shops with machine breakdowns. Comput Ind Eng 53(1):63–78
24. Zandieh M, Adibi MA (2010) Dynamic job shop scheduling using variable neighborhood search. Int J Prod Res 48(8):2449–2458
25. Mikkel T. Jensen (2001) Robust and flexible scheduling with evolutionary computation. PhD Dissertation, Department of Computer Science, University of Aarhus, Denmark
26. Kamrul Hasan SM, Sarker R, Essam D (2011) Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns. Int J Prod Res 49(16):4999–5015
27. Szelke E, Kerr R (1994) Knowledge-based reactive scheduling. Prod Plan Contr 5(2):124–145
28. Kacem I, Hammadi S, Borne P (2002) Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. IEEE Trans Syst Man Cybern 32(1):1–13
29. Mattfeld DC (1996) Evolutionary search and the job shop: investigations on genetic algorithms for production scheduling. Physica-Verlag, Heidelberg
30. A Yahyaoui, N Fnaiech, F Fnaiech (2009) New shifting method for job shop scheduling subject to invariant constraints of resources

availability. 35th Annual Conference of IEEE Industrial Electronics: 3211–3216
31. Kacem I, Hammadi S, Brone P (2002) Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Math Comput Simul 60(3–5):245–276
32. Mesghouni K, Hammadi S, Borne P (1997) Evolution programs for job-shop scheduling. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Orlando, Florida, vol. 1: 720–725
33. Lee DY, DiCesare F (1994) FMS scheduling flexible manufacturing systems using Petri nets and heuristic search. IEEE Trans Robot Autom 10(2):123–132
34. Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41(3):157–183