Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems

Shen, Xiao-ning; Yao, Xin

DOI:
10.1016/j.ins.2014.11.036

License:
Other (please specify with Rights Statement)

Document Version
Peer reviewed version

Citation for published version (Harvard):
Shen, X & Yao, X 2015, 'Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems', *Information Sciences*, vol. 298, pp. 198-224. https://doi.org/10.1016/j.ins.2014.11.036

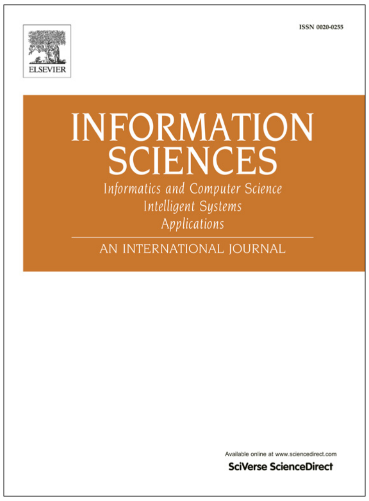
Mathematical modelling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems

Xiao-Ning Shen, Xin Yao

PII: S0020-0255(14)01117-7
DOI: http://dx.doi.org/10.1016/j.ins.2014.11.036
Reference: INS 11272

To appear in: *Information Sciences*

Received Date: 29 May 2014
Revised Date: 12 October 2014
Accepted Date: 22 November 2014


ISSN 0020-0255
ELSEVIER
INFORMATION SCIENCES
Informatics and Computer Science
Intelligent Systems
Applications
AN INTERNATIONAL JOURNAL
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Please cite this article as: X-N. Shen, X. Yao, Mathematical modelling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, *Information Sciences* (2014), doi: http://dx.doi.org/10.1016/j.ins.2014.11.036

Mathematical modelling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems

Xiao-Ning Shen[a,*], Xin Yao[b]

[a] School of Information and Control, Nanjing University of Information Science and Technology, Nanjing 210044, China

[b] CERCIA, School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom

Abstract Dynamic flexible job shop scheduling is of significant importance to the implementation of real-world manufacturing systems. In order to capture the dynamic and multi-objective nature of flexible job shop scheduling, and provide different trade-offs among objectives, this paper develops a multi-objective evolutionary algorithm (MOEA)-based proactive-reactive method. The novelty of our method is that it is able to handle multiple objectives including efficiency and stability simultaneously, adapt to the new environment quickly by incorporating heuristic dynamic optimization strategies, and deal with two scheduling policies of machine assignment and operation sequencing together. Besides, a new mathematical model for the multi-objective dynamic flexible job shop scheduling problem (MODFJSSP) is constructed. With the aim of selecting one solution that fits into the decision maker's preferences from the trade-off solution set found by MOEA, a dynamic decision making procedure is designed. Experimental results in a simulated dynamic flexible job shop show that our method can achieve much better performances than combinations of existing scheduling rules. Three MOEA-based rescheduling methods are compared. The modified ε-MOEA has the best overall performance in dynamic environments, and its computational time is much less than two others (i.e., NSGA-II and SPEA2). Utilities of introducing the stability objective, heuristic initialization strategies and the decision making approach are also validated.

Keywords Metaheuristics; Scheduling; Evolutionary computations; Mathematical modelling; Decision making

1. Introduction

Job shop scheduling problem (JSSP) is well-known as a strongly NP-hard combinational optimization problem [17], which mainly deals with finding out the best sequences for processing jobs on each operable machine to achieve the required objectives subject to precedence and processing time constraints. In JSSP, each operation of a job should be processed on a predefined machine only once in a fixed operation sequence. However, the wide employment of multi-purpose machines in the real-world job shop makes it more general that an operation can be managed by several machines, i.e., there are alternative routings, which is the so-called flexible job shop scheduling problem (FJSSP). FJSSP is a

* Corresponding author. Address: No.219, Ningliu Road, Nanjing, 210044, China. Tel.: +86 13813835843.
Email address: sxnystsyt@gmail.com

generalization of JSSP. It has more complexity than JSSP because the machine assignment problem which selects an alternative machine for each operation should also be addressed, besides the sequencing problem. Hence, FJSSP is also considered to be strongly NP-hard [15].

In real-world manufacturing systems, it is often the case that the working environment changes dynamically by unpredictable real-time events, such as one machine fails to work suddenly, and new jobs arrive in a stochastic way, etc. A previously optimal schedule may get poor system performance or even becomes infeasible in the new environment. Moreover, some information about the job shop is previously unknown. For example, the due date and processing time of the new job are not given until the job arrives. This kind of problems is generally known as dynamic scheduling [14]. As indicated in [29], dynamic scheduling is of great importance to the successful implementation of real-world manufacturing systems.

In the literature reported, there are mainly three categories of dynamic scheduling technologies, which are completely reactive, predictive-reactive, and pro-active scheduling [29]. Among them, predictive-reactive scheduling is the most commonly used. It has a scheduling/rescheduling process where previous schedules are revised to adapt to the new environment caused by dynamic events. Most of the existing research generated a new schedule by minimizing the effect of disruption on shop efficiency like make-span [2,5]. However, it may produce a new schedule totally different from the original one. For example, some remaining operations in the previous schedule which have not begun processing at the time of rescheduling may have their starting time accelerated or delayed. It has a serious impact on other production activities planned based on the original schedule, and brings instability and lack of continuity in the shop system [33]. Thus, both the performances of efficiency and stability should be considered in predictive-reactive scheduling. Above all, FJSSP in the real world has the dynamic and multi-objective nature.

A few literature have rescheduled dynamic job shops based on multiple objectives. Some of them only considered the performances of efficiency [2,5,32], e.g. make-span and tardiness. The others incorporated both efficiency and stability [14,33,62]. All the above studies used a weighted sum approach to convert multiple objectives to a single function. However, in most real-world cases, it would be difficult to identify suitable weights for each objective. On the other hand, multiple objectives such as make-span, tardiness and stability are usually conflicted with each other. It is better to handle multiple objectives with knowledge about their Pareto front. The various trade-offs among different objectives provided by the Pareto front is very useful in making an informed decision in dynamic scheduling. Evolutionary algorithms (EAs) have been recognized to be well suited for multi-objective optimization problems due to their capability to evolve a set of solutions simultaneously in one run. In the past 20 years, MOEA received much attention, and lots of success has been achieved [39].

So far, various EAs have been applied to solve manufacturing optimization problems. To optimize cutting parameters in the multi-pass turning operations, a comparative study of ten population-based optimization algorithms was performed in [47], and an artificial bee colony algorithm [48] and a hybrid Taguchi-differential evolution algorithm [49] were proposed, respectively. To select optimal machining parameters in milling operations, a hybrid differential evolution algorithm and a cuckoo search algorithm were presented in [50] and [51], respectively. As to the structural and shape design optimization problem, different EAs have been investigated, such as the hybrid of immune algorithm and Taguchi method [52], the hybrid differential evolution algorithm [53], the harmony search algorithm [54], the hybrid particle swarm optimization algorithm [55-57], Cuckoo search algorithm [13], genetic algorithm [58], the immune algorithm combined with a hill climbing local search [59,60], and the hybrid of immune and simulated annealing algorithm [61].

To our best knowledge, in the literature reported, MOEA has not yet been adopted to regenerate new schedules in a predictive-reactive way when shop environments change. The primary aim of this paper is to solve MODFJSSP based on an MOEA in a modified predictive-reactive scheduling manner. With the aim of covering the shortage of existing methods, three aspects are studied: (i) the mathematical model for MODFJSSP is constructed. In the model, four objectives including both the performances of efficiency and stability are considered simultaneously. Besides, constraints to the search space change dynamically when real-time events occur, which are also addressed in the model developed; (ii) a new MOEA-based rescheduling method is proposed, which do not regenerate a new schedule from scratch, but incorporate several heuristic methods in creating the initial population, and use problem specific genetic operators for variation; and (iii) in order to select one appropriate solution from the trade-off solution set found by an MOEA, a dynamic decision making procedure is designed. To evaluate the effectiveness of the proposed methods, a realistic dynamic flexible job shop is simulated with three purposes: (1) comparing the job shop performance produced by the MOEA-based rescheduling method to that of the combinations of existing heuristic rules and that of the existing static algorithms; (2) analysing different trade-offs among the four objectives, and comparing the overall performances in dynamic environments produced by three MOEAs (ε-MOEA [9], NSGA-II [10], SPEA2 [64]); and (3) investigating the impact and utility of the stability objective, heuristic initialization strategies and the decision making approach.

The remainder of this paper is organized as follows. Section 2 presents a short overview of the existing related work. Section 3 describes the problem formulation which introduces the rescheduling mode and constructs the mathematical model of MODFJSSP. In Section 4, the new MOEA-based rescheduling method for MODFJSSP and the dynamic decision making approach are described in detail. Experimental results are discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. Related work

Mathematical model is very useful for understanding the problem structure, thus a few literature have focused on mathematical formulations for static FJSSP. A mathematical model was presented in [15] to achieve optimal solution for small size problems. A mixed-integer linear programming model was developed for FJSSP in [30]. In [12], models formulated for FJSSP in literature were reviewed which categorised them as sequence-position variable based model, precedence variable based model, and time indexed model. As to the dynamic flexible job shop scheduling problem (DFJSSP), there have been few studies describing the mathematical model. In [63], a dynamic rescheduling model based on Multi-Agent System was proposed. A mathematical model for DFJSSP which minimized a weighted sum of two objectives (make-span and stability) was developed in [14]. It used binary variables to form constraints, which would introduce a lot of binary parameters. Besides, the definition of make-span at a specific rescheduling point is not given. In this paper, a dynamic multi-objective optimization model for DFJSSP which can capture the dynamic characteristics of both objectives (related to efficiency and stability) and constraints are constructed. DFJSSP is formulated in our model in a more comprehensible way. Stability measures the deviation between revised and original schedules, and there is no universal definition for stability. In [6], stability was defined as the number of times rescheduling occurred. In [1,44], stability was defined as the starting time deviation and operation sequence deviation. In [14,33], stability had two dimensions. One was the starting time deviation, and the other reflected how close to the current time changes were made. In our model, a more sophisticated definition for stability is presented which captures the deviation of operation starting time accelerating, starting time delay and completion time delay between two successive schedules, respectively.

Generally, there are mainly four research directions on multi-objective dynamic job shop scheduling in the existing literature. The first class developed customized rules before the running of a job shop. A co-evolutionary genetic programming method was developed in [26] for simultaneous design of dispatching rules and due-date assignment rules. Gene express programming was adopted in [27] to evolve machine assignment rules and job dispatching rules together in DFJSSP. This kind of methods is suitable for off-line optimization.

The second class belongs to completely reactive scheduling. In [32], at each scheduling point, an existing dispatching rule that performed best was determined by looking up the idiotypic network model constructed in advance. A heuristic was proposed in [28] to implement the reactive scheduling in a dynamic production environment where jobs arrive over time. A multiple attribute decision making technique which used grey numbers to deal with uncertainties was given in [45] to determine which lot was suitable to be processed next when a machine was free. [38] focused on the implementation concept of a discrete event simulation based “online near-real-time” dynamic scheduling system using conjunctive simulated scheduling. Completely reactive scheduling is quick to implement, but it considers only the local information, so the shop performance cannot be guaranteed. It is suitable for online dynamic scheduling.

The third class is called the predictive-reactive scheduling, which uses the global information and searches in a larger solution space in comparison with completely reactive scheduling [29]. In [2], an adaptive variable neighbourhood search was triggered in respond to a random event. A conventional genetic algorithm was used in [5] to regenerate a new schedule whenever a dynamic event occurred. However, it is often inefficient to restart the optimization process with a totally new population [34]. In order to solve the instability problem induced by unrestricted rescheduling, bi-objectives of stability and efficiency were considered simultaneously in [14,33,62], respectively. However, the weighted sum method was adopted in all the above studies to deal with multiple objectives. Predictive-reactive scheduling is also suitable for online dynamic scheduling. In our paper, we solve MODFJSSP based on an MOEA in a modified predictive-reactive scheduling way.

The fourth class can be categorized as the pro-active scheduling, which builds predictive schedules in advance. [21] introduced four different probability distributions to model stochastic processing times, and proposed three uncertainty handling methods to estimate the fitness of a solution. A multi-objective immune algorithm is given in [66] to produce robust scheduling solutions of uncertain scheduling problems described by the workflow simulation scheduling model. A simplified multi-objective genetic algorithm was proposed in [23] for the stochastic JSSP with exponential processing times. A robust and stable predictive schedule for one machine scheduling, JSSP, and FJSSP with random machine breakdowns was generated by a genetic algorithm in [24,20,3], respectively. A robust genetic algorithm was proposed in [4] to minimize the make-span of a parallel machine scheduling problem with fuzzy processing times. In [46], parallel machine scheduling with learning effects and fuzzy processing times was solved by the simulated annealing algorithm and the genetic algorithm. In this class of methods, optimization is performed offline.

Decision making is an important process in evolutionary multi-objective optimization, especially in the dynamic case. However, few attentions have been paid to this aspect. In [8,11], only problems with bi-objectives were considered, and a precise weight value of each objective should be provided by the decision maker (DM). However, the DM usually does not have enough knowledge about objective functions. When expressing preferences, they prefer to employ the qualitative language like "Objective A is more important than objective B" to describe the relative importance between two objectives [38]. For this reason, in our paper, linguistic terms are used to represent the DM's vague thought instead of requiring them to give numerical values so as to reduce his/her cognitive overload.

3. Problem formulation

3.1 Rescheduling mode

In order to infuse more reality in job shops, random new job arrivals and machine breakdowns (repairs) are considered. Among them, urgent job arrivals, machine breakdowns and repairs are regarded as critical events, and regular job arrivals are uncritical events.

A modified predictive-reactive dynamic scheduling is adopted. A production schedule for all the jobs at the initial time is generated at first. In order to reduce the rescheduling frequency, a critical-event-driven mode is employed. Once a critical event occurs, the rescheduling method is triggered. The time at which a new schedule is constructed is called the rescheduling point, and the time span between two successive rescheduling points is named the rescheduling interval. Besides, a special case is considered. Suppose by the time instant t^* after a specific rescheduling point t_l, all the scheduled jobs have finished so that all the available machines are idle, and the next critical event has not occurred. If the number of regular job arrivals between t_l and t^* is larger than an upper bound, then a new schedule is constructed for these new uncritical jobs to make full use of the machine resources. We call it the resource-idle-driven mode. The upper bound is set to be 5 here.

3.2 Mathematical modelling of MODFJSSP

Some notations used for describing the mathematical model are listed in Table 1.

Given the extremely high complexity of MODFJSSP, some common assumptions are made in this paper.

(1) A job can be processed by only one machine each time and each machine can perform at most one operation at a time.

(2) Once an operation has begun on a machine, it must not be interrupted, except for the machine breakdown.

(3) The machine setup time for two consecutive jobs is included in the processing time.

(4) There is no travel time between machines. Jobs are available for processing on a machine immediately after completing processing its previous operation.

(5) Jobs can wait to be processed in an unlimited buffer of a machine.

The mathematical model for MODFJSSP at a specific rescheduling point is formulated. At the rescheduling point t_l ($t_l > t_0$), considering all the current information gathered from the job shop floor, which includes attributes of the available machines, all the unprocessed job operations from the previous schedule, and the new arrival regular or urgent jobs since the

Table 1 Notations used for describing the mathematical model

t_0 : the initial time	t_l : The rescheduling point (l=1,2,…)
$n(t_l)$: Number of jobs that contain unprocessed and available operations at t_l	$J_i(t_l)$: The ith job at t_l , $i=1,2,\cdots,n(t_l)$
$m(t_l)$: Number of available machines at t_l	$M_k(t_l)$: The kth available machine at t_l , $k=1,2,\cdots,m(t_l)$
n_i : Total number of operations in job $J_i(t_l)$. It is predefined and unchanged with t_l	$n_i^{'}(t_l)$: Number of unprocessed and available operations left in job $J_i(t_l)$ at t_l , $1 \le n_i^{'}(t_l) \le n_i$
$C_i(t_l)$: The completion time of all the current available operations of job $J_i(t_l)$ at t_l	$S_i(t_l)$: The starting time of the first unprocessed operator of job $J_i(t_l)$ at t_l
$DD_i(t_l)$: Due date for all the current available operations of job $J_i(t_l)$ at t_l	a_i : The initial arrival time of job $J_i(t_l)$ in the job shop. It is unchanged with t_l
ω_i : The importance weight of job $J_i(t_l)$. It is unchanged with t_l	$I_i(t_l)$: Index of the first unprocessed operation in job $J_i(t_l)$ at t_l , $I_i(t_l) \ge 1$ and $I_i(t_l)+n_i^{'}(t_l)-1 \le n_i$
$c_{i(I_i(t_l)-1)}$: The completion time of the last operation of job $J_i(t_l)$ that has begun processing before t_l	$O_{ij}(t_l)$: The jth operation of job $J_i(t_l)$ which is available to be processed during the rescheduling interval of t_l , $j=I_i(t_l),I_i(t_l)+1,\cdots,I_i(t_l)+n_i^{'}(t_l)-1$
TMA_{ij} : The set of all the machines that can process operation $O_{ij}(t_l)$. It is predefined and unchanged with t_l	$Average\_p_{ij}$: The average processing time of operation $O_{ij}(t_l)$ for the machines in TMA_{ij} . It is unchanged with t_l
$MA_{ij}(t_l)$: The set of available machines that can process operation $O_{ij}(t_l)$ at t_l . $MA_{ij}(t_l) \subseteq TMA_{ij}$	$p_{ij}^{kk}(t_l)$: The processing time of $O_{ij}(t_l)$ on the machine $M_{kk}(t_l) \in MA_{ij}(t_l)$, $kk=1,2,\cdots,\left|MA_{ij}(t_l)\right|$ ($\mid \cdot \mid$ means cardinality of a set)
$A\_p_{ij}(t_l)$: The actual processing time of $O_{ij}(t_l)$ on its assigned machine at t_l	$s_{ij}(t_l)$: The starting time of operation $O_{ij}(t_l)$
$c_{ij}(t_l)$: The completion time of operation $O_{ij}(t_l)$, $c_{ij}(t_l)=s_{ij}(t_l)+A\_p_{ij}(t_l)$	$c^{k\_\text{last}}(t_{l-1})$: The completion time of the last operation processed on $M_k(t_l)$ before t_l
$q^k(t_l)$: Number of operations assigned to the machine $M_k(t_l)$ at t_l , $k=1,2,\cdots,m(t_l)$	$O^{kr}(t_l)$: The rth operation scheduled on the machine $M_k(t_l)$ at t_l , $r=1,2,\cdots,q^k(t_l)$
$p^{O^{kr}}(t_l)$: The processing time of operation $O^{kr}(t_l)$	$c^{O^{kr}}(t_l)$: The completion time of operation $O^{kr}(t_l)$
$R_i(t_l)$: The initial release time of job $J_i(t_l)$ during the rescheduling interval of t_l	$A_k(t_l)$: The initial available time of machine $M_k(t_l)$ during the rescheduling interval of t_l

previous rescheduling point $t_{l\text{-}1}$, a new schedule which represents the operations assigned to each machine and the corresponding sequence is constructed by optimizing the following objectives with respect to both shop efficiency and stability.

$$\min \mathbf{F}=[f_1(t_l), f_2(t_l), f_3(t_l), f_4(t_l)] \tag{1}$$

where $f_1(t_l)$ represents make-span which means the elapsed time required for finishing all the current jobs rescheduled at t_l; $f_2(t_l)$ is the tardiness measure which gives penalties to delays from the due date; $f_3(t_l)$ denotes the maximal machine workload which is to avoid assigning too much work to a single machine; $f_4(t_l)$ indicates the stability which measures the deviation between the new and original schedules.

The formula of each objective is given below.

$$f_1(t_l)=\max_{i=1,2,\cdots,n(t_l)}(C_i(t_l))-\min_{i=1,2,\cdots,n(t_l)}(S_i(t_l)) \tag{2}$$

$$f_2(t_l)=\sum_{i=1,2,\cdots,n(t_l)}\omega_i\cdot\max(0,C_i(t_l)-DD_i(t_l)) \tag{3}$$

$$f_3(t_l)=\max_{k=1,2,\cdots,m(t_l)}(\sum_{r=1}^{q_k(t_l)}p^{O^{kr}}(t_l)) \tag{4}$$

$$f_4(t_l)=\sum_{\substack{O_{ij}\in rush\\ starting}}\gamma\left|s_{ij}(t_l)-s_{ij}(t_{l\text{-}1})\right|+\sum_{\substack{O_{ij}\in delay\\ starting}}\left|s_{ij}(t_l)-s_{ij}(t_{l-1})\right|+\sum_{\substack{O_{ij}\in delay\\ delivery}}\left|c_{ij}(t_l)-c_{ij}(t_{l-1})\right| \tag{5}$$

where $s_{ij}(t_{l\text{-}1})$ and $c_{ij}(t_{l\text{-}1})$ are the starting time and completion time of operation O_{ij} in the previous schedule generated at the rescheduling point $t_{l\text{-}1}$, respectively.

The make-span measure $f_1(t_l)$ in Eq. (2) is calculated as the difference between the maximum completion time and the minimal starting time of all the current jobs at t_l. Here, $C_i(t_l)$ represents the completion time of all the current available operations of job $J_i(t_l)$. This is because by t_l, some operations might have become unavailable due to the occurrences of dynamic events. For example, the operation O_{47} cannot be processed because all of its alternative machines broke down before or at t_l, and have not gotten repaired by t_l. The succeeding operations O_{48} and O_{49} become unavailable either, because no pre-emption is allowed. Thus at t_l, $C_4(t_l)$ is the completion time of the last available operation O_{46}.

The tardiness measure $f_2(t_l)$ in Eq. (3) is defined as the weighted sum of differences between the completion time and due date of each job in which its completion time is larger than the due date. Similar to $C_i(t_l)$, $DD_i(t_l)$ is the due date for all the current available operations of job $J_i(t_l)$. According to this characteristic, it is generated by a modification to the commonly used total work content (TWK) rule [2]:

$$DD_i(t_l) = a_i + K_i * \sum_{\substack{j=1,2,\cdots,\\ I_i(t_l)+n_i'(t_l)-1}} Average\_p_{ij} \tag{6}$$

where $DD_i(t_l)$ is equal to the sum of the job arrival time and a multiple of the total average processing time from the first operation to the last available operation. The multiple K_i is called the tightness factor and is related to job characteristics. In our research, K_i follows the normal distribution with the mean of 1.5 and variance of 0.5.

The maximal machine workload $f_3(t_l)$ in Eq. (4) is calculated as the maximum working time spent at any available machine.

The stability measure $f_4(t_l)$ in Eq. (5) has three terms. The first term is the starting time deviation of operations which are rescheduled to start processing at an earlier time multiplied by a weight γ, where $\gamma > 1$. It gives more penalties to bring forward the starting time of an operation, because rush order cost is incurred if the delivery of materials is required earlier than planned based on the original schedule. Here, we set $\gamma=1.5$. The second term is the starting time deviation of operations which are rescheduled to begin processing at a later time. This case may lead to carrying costs because materials are delivered earlier than required. The third term is the completion time deviation of operations which have their ending time delayed, and it causes the deterioration of delivery performance.

It should be mentioned that at the initial time t_0, only three objectives which are make-span, tardiness and the maximal machine workload (without *stability*) are to be optimized.

In the dynamic flexible job shop, constraints to the search space change dynamically with occurrences of random events. They are listed as follows.

1) Machine set constraints

$$MA_{ij}(t_l) \subseteq \left\{M_1(t_l), M_2(t_l), \cdots, M_{m(t_l)}(t_l)\right\}, \text{ for } i=1,2,\cdots,n(t_l),\ j = I_i(t_l), I_i(t_l)+1, \cdots, I_i(t_l)+n_i'(t_l)-1 \tag{7}$$

The machine available set $MA_{ij}(t_l)$ contains all available machines that can process the operation $O_{ij}(t_l)$. It may change at different rescheduling points due to machine breakdowns or repairs.

2) Processing time constraints

For each machine $M_{kk}(t_l) \in MA_{ij}(t_l)$ ($kk = 1,2,\cdots,\left|MA_{ij}(t_l)\right|$), a processing time $p_{ij}^{kk}(t_l)$ is associated with operation $O_{ij}(t_l)$. If $O_{ij}(t_l)$ is assigned to $M_{kk}(t_l)$ in the schedule at t_l, then its actual processing time $A\_p_{ij}(t_l)=p_{ij}^{kk}$ ($i=1,2,\cdots,n(t_l)$, $j = I_i(t_l), I_i(t_l)+1, \cdots, I_i(t_l)+n_i'(t_l)-1$).

3) Precedence constraints

Operations of each job should be processed through the machines in a particular order. At t_l, job $J_i(t_l)$ consists of a sequence of $n_i^{'}(t_l)$ operations, and each operation $O_{ij}(t_l)$ can be processed on any machine out of its machine available set $MA_{ij}(t_l)$ ($i=1,2,\cdots,n(t_l)$, $j=I_i(t_l),I_i(t_l)+1,\cdots,I_i(t_l)+n_i^{'}(t_l)-1$).

4) Initial state constraints

$$R_i(t_l)=\max(t_l,c_{i(I_i(t_l)-1)}) \quad \text{for } i=1,2,\cdots,n(t_l) \tag{8}$$

$$A_k(t_l)=\max(t_l,c^{k-\text{last}}(t_{l-1})) \quad \text{for } k=1,2,\cdots,m(t_l) \tag{9}$$

Eq. (8) gives the initial release time of each job. It guarantees that all the operations that have begun processing before t_l not be considered in the rescheduling model. If $I_i(t_l)=1$, then $c_{i(I_i(t_l)-1)}=0$. Eq. (9) gives the initial idle time of each machine. It indicates that one machine is available until it has finished all the operations that have begun before t_l. If there is no operation processed on the machine $M_k(t_l)$ before t_l, then $c^{k-\text{last}}(t_{l-1})=0$.

5) No preemption constraints

5.1) No preemption in a single job

An operation of a job cannot be processed until its preceding operations are completed.

If $O_{ij}(t_l)$ is the first unprocessed operation of job $J_i(t_l)$ at t_l, i.e. $j=I_i(t_l)$, then it should start processing after the initial release time $R_i(t_l)$:

$$R_i(t_l)\le s_{ij}(t_l)\text{, for } i=1,2,\cdots,n(t_l),\ j=I_i(t_l) \tag{10}$$

Otherwise,

$$c_{i(j-1)}(t_l)\le s_{ij}(t_l)\text{, for } i=1,2,\cdots,n(t_l),\ j=I_i(t_l)+1,\cdots,I_i(t_l)+n_i^{'}(t_l)-1 \tag{11}$$

5.2) No preemption in a single machine

An operation can be processed on its assigned machine until the machine has finished its previous scheduled operations. Suppose at t_l, the operation $O_{ij}(t_l)$ is assigned to the machine $M_k(t_l)$, and it is scheduled as the r^{th} operation on $M_k(t_l)$, i.e., $O_{ij}(t_l)=O^{kr}(t_l)$.

If $r=1$, then $O_{ij}(t_l)$ should start processing after the initial machine available time $A_k(t_l)$:

$$A_k(t_l)\le s_{ij}(t_l)\text{, for } r=1,\ k\in\{1,2,\cdots,m(t_l)\} \tag{12}$$

If $r\ge 2$, suppose the completion time of the $(r\text{-}1)^{\text{th}}$ operation scheduled on $M_k(t_l)$ is $c^{O^{k(r-1)}}(t_l)$, then

$$c^{O^{k(r-1)}}(t_l)\le s_{ij}(t_l)\text{, for } r=2,\cdots,q^k(t_l),\ k\in\{1,2,\cdots,m(t_l)\} \tag{13}$$

5.3) Starting time of an operation

From Eq. (10) - Eq. (13), it can be concluded that the starting time of operation $O_{ij}(t_l)$ (suppose it is scheduled as the r^{th} operation on $M_k(t_l)$ at t_l) is:

$$s_{ij}(t_l)=\begin{cases}\max(R_i(t_l),A_k(t_l)) & \text{for } j=I_i(t_l),\ r=1\\ \max(R_i(t_l),c^{O^{k(r-1)}}(t_l)) & \text{for } j=I_i(t_l),\ r=2,\cdots,q^k(t_l)\\ \max(c_{i(j\text{-}1)}(t_l),A_k(t_l)) & \text{for } j=I_i(t_l)+1,\cdots,I_i(t_l)+n_i^{'}(t_l)-1,\ r=1\\ \max(c_{i(j\text{-}1)}(t_l),c^{O^{k(r-1)}}(t_l)) & \text{for } j=I_i(t_l)+1,\cdots,I_i(t_l)+n_i^{'}(t_l)-1,\ r=2,\cdots,q^k(t_l)\end{cases} \tag{14}$$

6) Interruption mode constraints

At t_l, if one machine breaks down, and an operation is being processed on it, then the work has to stop and wait until the machine has been repaired. On the other hand, if a broken machine gets repaired at t_l, then the previously interrupted operation (if any) will resume to be processed on it with the interrupt-resume mode, or be processed from scratch with the interrupt-repeat mode [1]. In this paper, the interrupt-resume mode is used.

Compared to the existing model in [14], superiorities of our model can be summarized as follows: (1) Multi-objective handling method. A dynamic multi-objective optimization model for DFJSSP is constructed, where three efficiency objectives and one stability objective are optimized simultaneously based on Pareto dominance, instead of being converted into a single one in [14]; (2) Objective definitions. In our model, considering the dynamic feature of DFJSSP, objective definitions are formulated specifically for each rescheduling point t_l. Since some unprocessed operations might become unavailable temporarily due to occurrences of random events (e.g. machine breakdowns), the four objectives (make-span, tardiness, maximal machine workload and stability) at t_l are defined only for current available operations. In contrast, only two objectives of make-span and stability were considered in [14], and the definition of make-span at a specific rescheduling point was not given; (3) Consideration of dynamic constraints. In the dynamic flexible job shop, in addition to objectives, constraints to the search space also change dynamically with occurrences of random events. We classify these dynamic constraints into six categories, and give straightforward and comprehensible definitions to them which can capture dynamic features of constraints. In contrast, [14] used binary variables to form constraints, which would introduce a lot of extra binary parameters. Besides, the initial state constraints, the earliest starting time of an operation, and interruption mode constraints were not considered in [14]; and (4) Definition of the stability objective. A more sophisticated definition for stability is presented which captures the deviation of operation starting time accelerating, starting time delay and completion time delay between two successive schedules, respectively, since they have different impact on the production plan.

4. A predictive-reactive scheduling method to solve MODFJSSP

4.1 Framework of the predictive-reactive scheduling method

The flowchart of our predictive-reactive scheduling in MODFJSSP is summarized in Fig. 1. At the initial time of the shop floor, a predictive schedule is generated by an MOEA considering three objectives which are make-span, tardiness and the maximal machine workload. Then during the implementation of the schedule, at each rescheduling point, an MOEA-based rescheduling method is triggered to construct a new schedule by considering four objectives which are make-span, tardiness, the maximal machine workload and stability simultaneously. The newly generated schedule is implemented in the job shop until the next rescheduling point comes, at which time the rescheduling method is triggered again. In short, MODFJSSP is a dynamic process formed by a sequence of multi-objective FJSSPs with different sets of job operations and machines to be scheduled. This process continues until all the jobs appearing in the shop floor have finished. At each scheduling point, a set of non-dominated solutions are obtained by an MOEA. Thus one solution that fits into the DM’s preferences is selected by a decision making procedure and implemented in the shop floor.

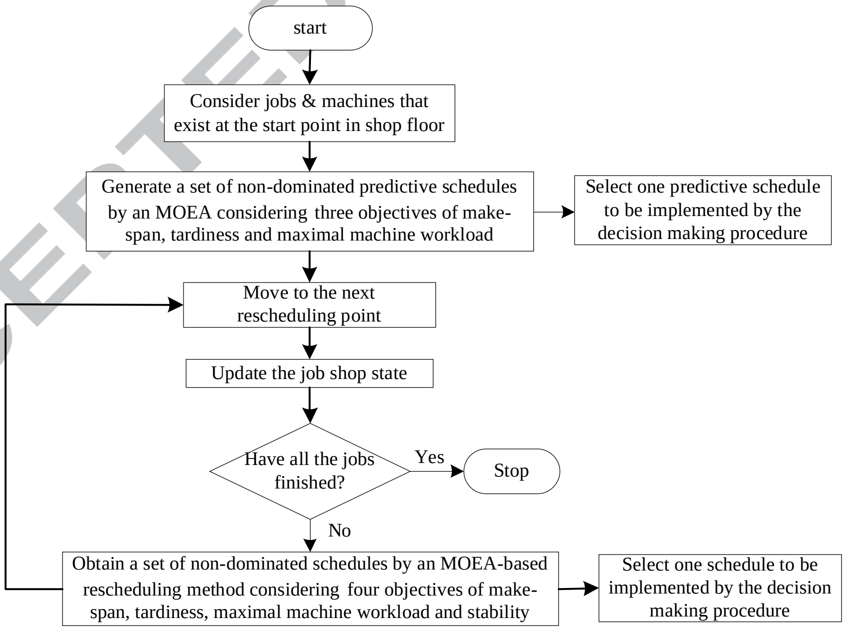

start
Consider jobs & machines that exist at the start point in shop floor
Generate a set of non-dominated predictive schedules by an MOEA considering three objectives of make-span, tardiness and maximal machine workload
Select one predictive schedule to be implemented by the decision making procedure
Move to the next rescheduling point
Update the job shop state
Have all the jobs finished?
Yes
Stop
No
Obtain a set of non-dominated schedules by an MOEA-based rescheduling method considering four objectives of make-span, tardiness, maximal machine workload and stability
Select one schedule to be implemented by the decision making procedure

Fig. 1. Flowchart of the proposed predictive-reactive scheduling in the multi-objective flexible job shop.

4.2 The ε-MOEA-based rescheduling method for MODFJSSP

As indicated in Section 4.1, MODFJSSP can be seen as a dynamic process formed by a sequence of multi-objective FJSSPs. However, we should not just treat MODFJSSP as a sequence of independent static problems and use the existing static MOEAs to solve it. There are mainly four reasons for that: (1) these problems are not independent and they are related to each other. At each rescheduling point, a new FJSSP with an updated set of job operations and machines is formed and to be scheduled. Most operations in the current problem are composed of unprocessed operations left from the previous schedule, and most of the machines are the same as those of the prior problem; (2) in a real-world job shop system, stability and continuity which means there should be a small difference between the new generated schedule and the original one are very important. So when rescheduling, arrangements in the previous schedule should be taken into account; (3) MODFJSSP is a dynamic problem thus some dynamic optimization strategies should be introduced to make the algorithm adapt to the changing environments quickly. Here, the features of different dynamic events can be utilized to guide the searching direction; and (4) as indicated in [34], it is often inefficient to restart the dynamic optimization process with a totally new population. Thus, we should invent a new dynamic algorithm for solving MODFJSSP, which can capture the correlations between the sequence of problems, and avoid producing a new schedule totally different from the original one.

ε-MOEA is an ε-domination based steady-state MOEA. It employed efficient parent and archive update strategies, and was validated that it is a balanced algorithm which can produce good convergence and diversity with a very small computational effort, especially when dealing with many objectives (3 or more) [9]. MODFJSSP studied in this paper is a dynamic problem with four objectives. In order to solve it in an efficient way, an ε-MOEA-based rescheduling method is proposed. Meanwhile, to keep the system stability and continuity in mind, and to exploit the information left from the original schedule and the characteristics of different dynamic events, our ε-MOEA-based rescheduling method is featured with three points: (1) some heuristic strategies are incorporated in constructing the initial population of ε-MOEA at each rescheduling point; (2) new individual representations and two kinds of problem specific variation operators are designed so that the proposed method can handle operation sequencing and machine assignment simultaneously; and (3) the stability objective is considered together with the shop efficiency (make-span, tardiness, the maximal machine workload) for multi-objective optimization in our approach. The procedure of ε-MOEA-based rescheduling method at the rescheduling point t_l ($t_l > t_0$) is presented below.

Step 1: Initialization: Construct the initial population $P(t_l)$ by some heuristic strategies according to the updated job shop state at t_l. Then multi-objective evaluations are performed, and all the non-dominated solutions are determined to form the initial archive population $A(t_l)$. Set the counter of objective evaluation numbers $ct = population\_size$.

Step 2: Population selection: One individual *sp* is chosen from the population $P(t_l)$. Here, the tournament selection method is used. Two individuals are picked up uniformly at random from the population, and check the domination of each other. If one dominates the other, the former will be chosen. Otherwise, one of them is selected at random.

Step 3: Archive selection: One solution *e* is chosen uniformly at random from the archive $A(t_l)$.

Step 4: Variation: Two offspring sc_1 and sc_2 are generated from sp and *e* by two kinds of problem specific variation operators.

Step 5: Decoding and objective evaluation: Evaluate the offspring individuals sc_1 and sc_2.

Step 6: Update of the population: Offspring individuals sc_1 and sc_2 are included in $P(t_l)$ using a *pop_acceptance* procedure.

Step 7: Update of the archive: Individuals sc_1 and sc_2 are included in $A(t_l)$ using an *archive_acceptance* procedure.

Step 8: Termination: If the termination criterion is not satisfied, set $ct = ct + 2$ and go to Step 2, else output $A(t_l)$, and select one solution from $A(t_l)$ as the implementation schedule based on a decision making procedure.

In the above Steps 6 and 7, the *pop_acceptance* and *archive_acceptance* procedures are the same as those in [9]. The termination criterion is the counter *ct* achieves a predefined maximum number of objective evaluations. It should be mentioned that at the initial time t_0 of the job shop, the *ε*-MOEA used to generate a set of predictive schedules is also based on the procedure introduced above. The differences are that the random population initialization is used in Step 1 instead of the heuristic population initialization, and when evaluating an individual, only three objectives (without *stability*) are considered.

Details of our implementation for the *ε*-MOEA-based rescheduling method and the decision making procedure will be discussed below.

4.2.1 Representations

In MODFJSSP, both the operation sequence vector and the machine assignment vector are used to represent a complete scheduling individual. Fig. 2 gives an example of such a representation.

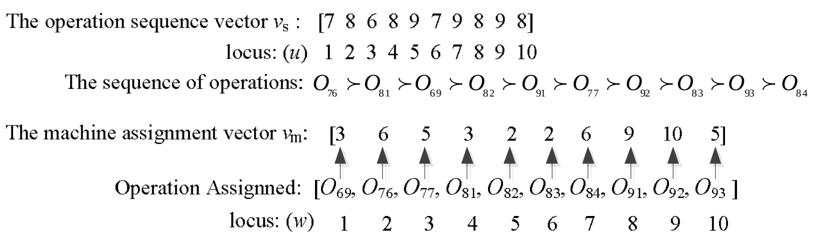

The operation sequence vector v_s : [7 8 6 8 9 7 9 8 9 8]
locus: (u) 1 2 3 4 5 6 7 8 9 10
The sequence of operations: $O_{76} \succ O_{81} \succ O_{69} \succ O_{82} \succ O_{91} \succ O_{77} \succ O_{92} \succ O_{83} \succ O_{93} \succ O_{84}$
The machine assignment vector v_m: [3 6 5 3 2 2 6 9 10 5]
Operation Assigned: [O_{69}, O_{76}, O_{77}, O_{81}, O_{82}, O_{83}, O_{84}, O_{91}, O_{92}, O_{93}]
locus: (w) 1 2 3 4 5 6 7 8 9 10

Fig. 2. An example of the representation of a chromosome.

For the operation sequence vector, a job-based representation [18] is used. All the operations from the same job are denoted by the job number. Take Fig. 2 as an example. Suppose at a specific rescheduling point, the operations O_{69}, O_{76} and O_{77} from jobs 6 and 7 are left unprocessed from the previous schedule, and there are two new jobs 8 and 9. Thus, the operation sequence vector contains the job numbers 6, 7, 8, and 9. Each operation is interpreted according to its order of appearance in the sequence vector. For example, the first appearance of number 8 represents O_{81}, the second appearance of 8 means O_{82}, and so on. So the operation sequence vector in Fig. 2 can be interpreted as:

$$O_{76} \succ O_{81} \succ O_{69} \succ O_{82} \succ O_{91} \succ O_{77} \succ O_{92} \succ O_{83} \succ O_{93} \succ O_{84}$$

where $a \succ b$ means operation a joins the waiting queue of its assigned machine first, then operation b is scheduled.

The machine assignment vector represents the assigned machine of each operation. The order is from the first remaining operation of the oldest job (with the minimum job index) to the last remaining operation of the newest job (with the maximum job index). For example, in Fig. 2, suppose the current available machines are 2, 3, 5, 6, 9, and 10. In the machine assignment vector, the first element of 3 means the first remaining operation O_{69} of the oldest job 6 will be assigned to machine 3, the second element of 6 represents the first remaining operation O_{76} of the second oldest job 7 will be assigned to machine 6, and so on. Thus, the machine assignment vector can be interpreted as:

$$O_{69} \to \text{machine } 3, O_{76} \to \text{machine } 6, O_{77} \to \text{machine } 5, O_{81} \to \text{machine } 3, O_{82} \to \text{machine } 2,$$

$$O_{83} \to \text{machine } 2, O_{84} \to \text{machine } 6, O_{91} \to \text{machine } 9, O_{92} \to \text{machine } 10, O_{93} \to \text{machine } 5$$

where $\to$ means the operation is assigned to the corresponding machine.

4.2.2 Decoding

For the convenience of objective evaluation, a scheduling solution should be decoded into the form of Gantt chart. Fig. 3 gives the Gantt chart of the chromosome represented in Fig. 2. Assume the rescheduling point in Fig. 2 is $t_l = 10$, and at $t_l = 10$, two operations O_{68} and O_{75} in the previous schedule are being processed on machine 6 and 5, respectively. The predefined processing time of each operation on the assigned machine is listed in Table 2. From Fig. 2, we can see that the sequence of operations is: $O_{76}(6) \succ O_{81}(3) \succ O_{69}(3) \succ O_{82}(2) \succ O_{91}(9) \succ O_{77}(5) \succ O_{92}(10) \succ O_{83}(2) \succ O_{93}(5) \succ O_{84}(6)$, where the number in the parentheses is the machine assignment of each operation. The process of constructing the Gantt chart in Fig. 3 is described as follows. First, take the first operation O_{76} in the operation sequence into account. Since O_{76} is also assigned to machine 6, it can be processed until both O_{75} (its previous operation in the same job 7) and O_{68} (its previous operation in the same machine 6) have finished due to the no preemption constraints. Then, since the second operation O_{81} is the first operation of job 8 and its assigned machine 3 is idle, it is scheduled at t_l=10 for a processing time of 0.5 time units. Next, the third operation O_{69} is to be processed on machine 3 at the maximum value of the completion time of O_{68} and O_{81}. The following operations in the operation sequence are scheduled to the assigned machines following the same method.

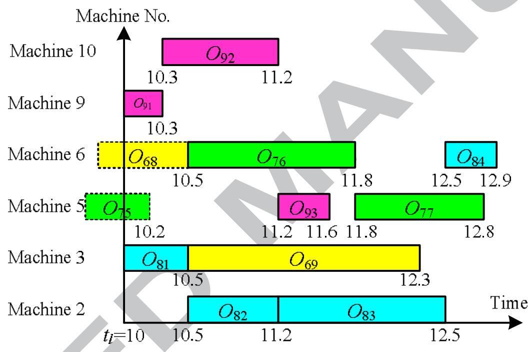


Fig. 3. Decoded schedule of the chromosome in Fig. 2.

Table 2 Processing time of each operation on the assigned machine in Fig. 2

operation	O_{76}	O_{81}	O_{69}	O_{82}	O_{91}	O_{77}	O_{92}	O_{83}	O_{93}	O_{84}
assigned machine	6	3	3	2	9	5	10	2	5	6
processing time	1.3	0.5	1.8	0.7	0.3	1	0.9	1.3	0.4	0.4

It should be noted that the idle time insertion method which inserts an operation into the first available idle time interval of its assigned machine is used to make full use of the machine resources. For example, O_{93} goes after O_{77} on machine 5 according to the operation sequence in Fig. 2. But in Fig. 3, there is an interval of idle time on machine 5 between the completion time of O_{92} (11.2) and the starting time of O_{77} (11.8). Meanwhile, the processing time of O_{93} (0.4) is smaller than the length of the time interval (11.8-11.2=0.6). Thus O_{93} is inserted into this interval and begins at the completion time of O_{92} (11.2). Let $L(t_l)$ be the total number of operations to be processed at t_l, and $W(i, j)$ be the locus of $O_{ij}(t_l)$ in the machine assignment vector. The pseudo code of the decoding procedure is shown in Fig. 4.

```
Procedure: Decoding Procedure
Input: $JJ_{1\times n(t_l)}$: the array of available jobs at $t_l$, chromosome $v_s(u)$, $v_m(w)$
// examples of $v_s(u)$, $v_m(w)$ is given in Fig. 2, $n(t_l)$ is the total number of available jobs at $t_l$
Output: a schedule
for $i = 1 \rightarrow n(t_l)$
    $j_{JJ(i)} \leftarrow 0$;      // $j_{JJ(i)}$ means the number of operations already assigned for job JJ(i)
end for
for $u = 1 \rightarrow L(t_l)$   // $L(t_l)$ is the total number of operations to be processed at $t_l$
    $i \leftarrow v_s(u)$, $M \leftarrow v_m(W(i, I_i(t_l) + j_i))$;  // $I_i(t_l)$ is the index of the first unprocessed operation in job $J_i(t_l)$
    search an available idle time interval on machine $M$ from left to right for operation $O_{i(I_i(t_l)+j_i)}$;
    if such a time interval is found,
    then the operation is inserted there;
    else the operation is scheduled at the end of machine $M$;
    end if
    $j_i \leftarrow j_i + 1$
end for
return the schedule;
```

Fig. 4. Pseudo code of the decoding procedure.

4.2.3 Update of the job shop state

Once the rescheduling procedure is triggered, the shop state should be updated at first.

(i) At t_l, information left from the previous schedule should be collected, which includes the remaining unprocessed operations, and the operations that are being processed on each machine at t_l. Meanwhile, information about new arrival jobs since the previous rescheduling point t_{l-1} and the current available machines must also be gathered.

(ii) Update the machine available set $MA_{ij}(t_l)$ for all the current operations. By t_l, if some machines have broken down and thus become unavailable, they should be removed from the machine set of each operation. For a specific operation, if there is no machine available to process it, then it will not enter the rescheduling model at t_l. On the other hand, if a broken machine has been repaired by t_l, it must rejoin the machine set of corresponding operations. Meanwhile, all the operations that cannot be processed temporarily due to the previous breakdown of the repaired machine, must be added to the rescheduling model at t_l.

(iii) Update the machine available time $A_k(t_l)$ and job release time $R_i(t_l)$ according to the initial state constraints (Eq. (8) and (9)) for all the current machines and jobs.

4.2.4 Construction of the initial population in rescheduling

At each rescheduling point, a new multi-objective FJSSP with an updated set of job operations and machines is formed and to be scheduled. In order to guide the search of the *ε*-MOEA-based rescheduling method and accelerate the convergence speed so that the method can adapt to the new environment quickly, some heuristic methods are incorporated in creating the initial population of ε-MOEA, which makes the proposed rescheduling methods different from those completely rescheduling approaches that regenerate a new schedule from scratch [14,33,62].

(i) Make use of the characteristics of different dynamic events. As indicated in [29], schedule repair refers to local adjustments of the original schedule, and it can preserve the system stability well. Hence, three schedule repairs are employed here to exploit the dynamic event characteristics. Firstly, for machine breakdowns, a modification to the partial schedule repair [1] is designed. All the unaffected operations remain unchanged both for their machines and starting times. The directly affected operations (previously scheduled on the broken machine and unprocessed) are moved to another alternative machine if possible, and the indirectly affected operations are assigned to the same machines as before. Only the sequences of the affected operations are rescheduled. Secondly, for machine repairs, some operations are shifted to the repaired machine so as to balance the machine workload. These operations should satisfy that they can be processed by the repaired machine, and the shift will not affect the starting time of other operations. Thirdly, for new job arrivals, a new job is scheduled as soon as one of its alternative machines becomes idle. The result of schedule repair is called the schedule repair solution.

(ii) Make use of the history information. At each rescheduling point, information left from the previous schedule is regarded as the history information which can be exploited. The sequence and machine assignment vector of all the remaining unprocessed operations in the old schedule is called the history solution.

(iii) Make use of the heuristic machine assignment rules. Two machine assignment rules following the approach of localization [22] are adopted. The first rule searches for the global minimum in the processing time table [14]. Then it fixes that assignment, and updates the machine workload on every other operation. The second rule randomly permutes jobs and machines in the processing time table at first. Then for each operation, it finds the machine with the minimum processing time, fixes that assignment, and updates the machine workload. The first rule determines the machine assignment for each operation uniquely, while the second rule finds different assignments in different runs of the rules.

(iv) Incorporation of random individuals. In order to introduce diversity, some random individuals are created in the initial population. Sequence vectors are generated by permuting all the current operations at random. For half of these random individuals, machine assignments are determined according to the two rules described above. Each operation in another half is assigned to a randomly chosen machine from its machine available set.

In this paper, 20% of the initial population are formed with the history solution and its variants by mutation (as introduced in Section 4.2.5), 30% with the schedule repair solution and its variants, and 50% with the random individuals.

4.2.5 Problem specific genetic operators

I) Sequence based variation operators

In order to preserve the feasibility of the generated offspring, a specialized crossover operator is designed for the operation sequence vectors in the individuals. It works as follows.

Step i: All the current available jobs at the rescheduling point t_l are divided uniformly at random into two groups: G^1 and G^2.

Step ii: The operations from the first job group G^1 are picked from $Parent^1$, and recorded in a new array R^1 as their original positions in $Parent^1$. The operations from G^2 are picked from $Parent^2$, and recorded in a new array R^2 as their original positions in $Parent^2$.

Step iii: All of the recorded operations in R^1 and R^2 are merged according to their original sequences to generate an offspring.

Another offspring is generated using the same method described above, except that the operations from G^1 are picked from $Parent^2$, and the operations from G^2 are picked from $Parent^1$. This procedure is illustrated in Fig. 5. When merging, if two operations have the same positions in the parents, such as jobs 6 and 7 in the third order of each parent in Fig. 5, their sequence in the offspring is generated uniformly at random.

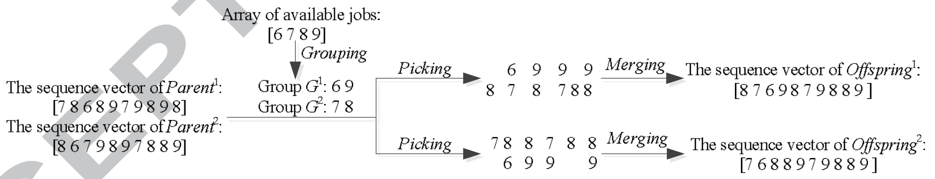


Fig. 5. An example of the crossover for the operation sequence vectors in the scheduling individuals.

The sequence based mutation operator is the commonly used swap and insert operators. The swap operator selects two operations in the operation sequence vector at random, and exchanges the positions of them. The insert operator inserts one randomly selected operation before another one. When performing a mutation on an individual, either the swap or the insert is chosen with the possibility of 0.5.

It should be noted that when performing the sequence based variation operators, the machine assignment vector is kept unchanged. Since our representation of the machine assignment vector given in Section 4.2.1 is not related to the operation sequence, the variation of the sequence vector will not affect it. This will not cause the potential infeasibility problem when performing the swap or insert operator, which would otherwise be faced if we had used the representation where each machine corresponds to each operation in the operation sequence vector as in [14].

II) Machine based variation operators

In the representation of the machine assignment vector given in Section 4.2.1, the machines in the same positions of two parents correspond to the same operation. So the traditional single point crossover can be used. The mutation operator is performed as follows. An allele is chosen randomly, and the machine on which the operation is to be processed is replaced with one of the alternative machines. Similarly, when performing the machine based variation operators, the operation sequence vector is kept unchanged in the offspring.

4.2.6 Parameters

We also apply NSGA-II and SPEA2 to explore the Pareto front of non-dominated schedules at each rescheduling point in order to understand the impact of different algorithms on the performance of MODFJSSP. The population initialization and variation operators introduced above are also used in NSGA-II and SPEA2.

Parameters used by the three MOEA-based rescheduling methods are given in Table 3. SPEA2 had a tournament size of 2 for mating selection and the archive size of 100. Each algorithm stopped after 20000 objective evaluations had been performed.

Table 3 Parameter settings of MOEA-based rescheduling methods

Population size	100
Sequence based crossover possibility	0.9×0.5=0.45
Machine based crossover possibility	0.9×0.5=0.45
Sequence based mutation possibility	0.2×0.5=0.1
Machine based mutation possibility	0.2×0.5=0.1
maximum number of objective evaluations	20000

4.3 Decision making in DFJSSP

At each rescheduling point, once a set of trade-off solutions are found by the MOEA method, one solution that fits into the DM's preferences should be selected, and implemented in the shop floor. Here, a decision making method inspired by the Analytic Hierarchy Process (AHP) [35,43] and the Multi-attribute Utility Theory (MAUT) [16] is proposed, and the procedure is given as follows.

Step i: Construction of the pairwise comparison matrix.

Suppose there are *N_O* objectives to be optimized. As in AHP, the pairwise comparison questions of "How important is the objective f_i relative to f_j?" ($i, j = 1, 2, \cdots, N\_o$, $j > i$) are answered by the DM a priori. So there are totally $N\_o \cdot (N\_o - 1)/2$ comparisons. The answers use the following nine-point scale which describes the degree of the preference for one objective versus another [16],

$$c_{ij}=\begin{cases}1, \text{ Equal importance or preference}\\ 3, \text{ Moderate importance or preference of one over another}\\ 5, \text{ Strong or essential importance or preference}\\ 7, \text{ Very strong or demonstrated importance or preference}\\ 9, \text{ Extreme importance or preference}\\ 2\text{n}, \ n=1,2,3,4, \text{ between the intensity of importance of } 2n-1 \text{ and } 2n+1\end{cases}$$

where c_{ij} denotes the value derived by comparing the objective f_i versus f_j. And we have $c_{ji}=1/c_{ij}$ and $c_{ii}=1$ ($i, j=1,2,\cdots,N\_o$) to maintain the consistency of judgements about any pair of objectives. Then, the pairwise comparison matrix $C_1=(c_{ij})_{N\_o\times N\_o}$ can be constructed.

Step ii: Estimation of the weight vector for multiple objectives.

A weight vector $w=(w_i)_{1\times N\_o}$ should be estimated so that the entries $W_{ij}=w_i/w_j$ in the matrix $W=(W_{ij})_{N\_o\times N\_o}$ will provide the best consistency with the judgement c_{ij} in the pairwise comparison matrix C_1 ($i, j=1,2,\cdots,N\_o$). Here, the logarithmic least squares method [36] is adopted. First, the geometric mean of each row in the matrix C_1 is calculated. Then normalize each geometric mean by dividing it by the sum of them.

Step iii: Normalization of the objectives of trade-off solutions.

For the objective f_i ($i=1,2,\cdots,N\_o$), find out the maximum $f_i^{\max}$ and minimum $f_i^{\min}$ among all the trade-off solutions obtained at current rescheduling point. Then for each solution *x*, the normalized objective $n\_f_i(x)$ is calculated as:

$$n\_f_i(x)=(f_i^{\max}-f_i(x))/(f_i^{\max}-f_i^{\min}),\ i=1,2,\cdots,N\_o \tag{15}$$

Since all the four objectives in the model of MODFJSSP are to be minimized, while the utility function in the MAUT has to be maximized, in Eq. (15), $f_i(x)$ is not only normalized to locate in [0, 1], but also converted so that the bigger the value of $n\_f_i(x)$, the better it is.

Step iv: Calculation of the utility value for each trade-off solution.

In [25], it was pointed out that the optimal method to find the utility value for each trade-off solution *x*, is a weighted geometric means of its multiple objective values:

$$U(x)=\prod_{i=1}^{N\_o} n\_f_i(x)^{w_i/\Sigma_i^{N\_o}w_i} \tag{16}$$

Step v: Choose the solution with the maximum utility value as the final schedule to be implemented.

It should be mentioned that the execution of the above Steps i and ii can be either before the running of the dynamic job shop, or performed dynamically. In the first way, the pairwise comparison matrix and the weight vector for multiple objectives are determined beforehand and remain unchanged during the dynamic process. Only Steps iii, iv, and v are performed at each rescheduling point. In the second way, the preferences of DM are allowed to be changed dynamically, which can be realized through a graphical user interface, and DM can interact with the job shop running process. To simplify the problem, the first way is employed in this paper. Suppose the pairwise comparison matrix for the four objectives given in Section 3.2 is $C_1 = (c_{ij})_{4\times 4} = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1/2 & 1 & 2 & 1/2 \\ 1/3 & 1/2 & 1 & 1/3 \\ 1 & 2 & 3 & 1 \end{bmatrix}$. Then the weight vector can be calculated as $w = (w_i)_{1\times 4} = [0.3512,\ 0.1887,\ 0.1089,\ 0.3512]$.

5 Experimental studies

5.1 Dynamic job shop simulation model

In order to validate the effectiveness and efficiency of our proposed method, a realistic job shop has been simulated. The characteristics of the simulation were synthesized from literature ([2,5,33]). All the experiments were performed in the software of MATLAB 2010a on a personal computer with Intel core i5, 3.2 GHz CPU and 4 GB RAM.

It was point out that a job shop with more than six machines presents the complexity involved in large dynamic job shop scheduling [2]. In this paper, a job shop consisting of ten machines (m=10) is simulated to evaluate the performance of methods.

The number of operations per job and the number of machines that can process each operation both vary uniformly in the interval of [1, m]. The set of alternative machines that can process each operation are randomly selected from $\{1, 2, \cdots, m\}$. The processing time of each operation follows the exponential distribution with the mean of 1.

For each machine, the time interval between fails (TBF) and the time to repair (TTR) are also assumed to follow an exponential distribution. To infuse realism into the simulation, each machine was assigned different mean time between failure (MTBF) and mean time to repair (MTTR). MTBF and MTTR vary uniformly in [100, 300] and [20, 120], respectively. These values were chosen such that on average, a machine is available for 130 time units, then breaks down and gets repaired for 70 time units. Hence, a machine's availability is 65%.

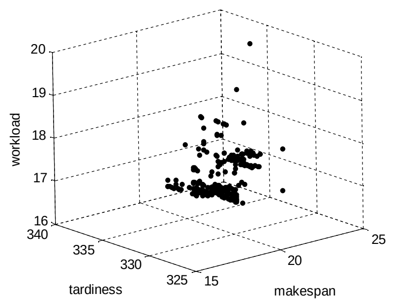
Simulation starts with a 10×10 static flexible job shop problem, where the initial numbers of jobs and machines are both 10. Then new jobs arrive following the Poisson distribution [19], so the time between new job arrivals is distributed exponentially. Suppose the mean time between job arrivals (*MTBJA*) is 0.625. New jobs are assigned random weights such that the weights of 20% of jobs are 1, 60% are 2, and 20% are 4 [26]. Here, jobs with weight 1 and weight 2 are of less importance and average importance, respectively, both of which are considered as regular jobs. Jobs with weight 4 are urgent jobs which are of high importance.

Simulation continues until the number of jobs which have arrived at the shop floor reaches 1240. As in [33], to eliminate transient effects, all dynamic events occurring during the interval of last 1000 new job arrivals are considered to evaluate the statistical performance measures in the dynamic environment.

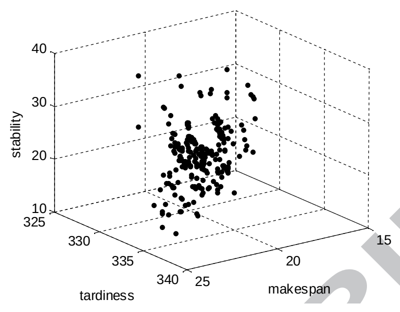
One dynamic flexible job shop instance was generated using the parameters introduced above, and it was used as the problem instance in all the following experiments. The data of the instance is provided in Appendix A.

5.2 Pareto fronts of the evolved schedules at rescheduling points

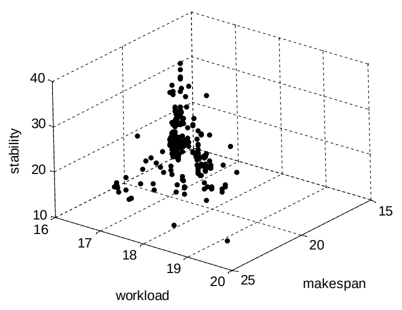
At each rescheduling point, a set of non-dominated solutions was evolved by an MOEA-based rescheduling method. In order to demonstrate the trade-offs among these solutions, one rescheduling point was selected at random and taken as an example. At $t_l=344.2344$, machine 7 broke down, and only machines 1, 4, 5, 8, 10 were available. By this time instant, 544 jobs had already arrived in the shop floor, and there were totally 26 jobs with 116 operations left to be processed. 30 independent runs of each of the three MOEA-based (ε-MOEA, NSGA-II, and SPEA2) rescheduling methods were performed. The evolved Pareto fronts in each run of each algorithm were combined, and the non-dominated solutions were determined from them. Since there are totally four objectives to be optimized, the obtained aggregated Pareto front cannot be plotted graphically. We pick three objectives from the four in turn, and give the 3-D plot of them respectively in Fig. 6. Firstly, it is obvious from Fig. 6(b)(c)(d) that the efficiency objectives are seriously conflicted with the stability measure. When tracing along the Pareto front to find solutions that have high efficiency (small make-span, tardiness and workload), it can be observed that the stability measure becomes worse. Secondly, it can be seen from Fig. 6(a) that given a similar value of workload, there is also a slight conflict between the objectives of make-span and tardiness. A smaller make-span normally leads to a larger tardiness. Thirdly, Fig. 6(c) suggests that given a similar value of stability, there is little conflict between the make-span and the workload, which means that a small make-span requires a small maximal workload. This is reasonable because a small make-span implies full use of the resources to finish all the current operations as soon as possible, thus the workload balance among different machines can be achieved. All the above observations suggest that solutions that provide better performance of shop efficiency will result in the deterioration in the system stability. There is no solution that can simultaneously optimize all the considered objectives.



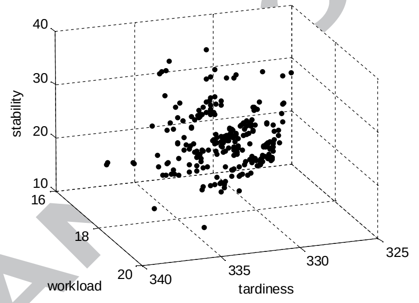
(a) Pareto front plotted on make-span, tardiness, and workload



(b) Pareto front plotted on make-span, tardiness, and stability



(c) Pareto front plotted on make-span, workload and stability



(d) Pareto front plotted on tardiness, workload and stability

Fig. 6. Pareto front of the non-dominated solutions plotted on three of the four objectives at the rescheduling point t_l =344.2344 .

Table 4 gives several examples of the objective vectors selected from the aggregated Pareto front. A solution may perform well for one objective, but give bad results for others, such as Solution$_1$ - Solution$_4$. And one solution may have a 'not bad' value on each objective, which means a good compromise among all the objectives, such as Solution$_5$ – Solution$_7$. Pareto front produced by the MOEA-based method can provide the DM with a full knowledge about the various trade-offs among multiple objectives. It is helpful for him/her to make an informed decision about the 'best compromise' with regards to his/her preference.

Table 4 Several examples of objective vectors selected from the aggregated Pareto front at the rescheduling point t_l =344.2344

	[f_1, f_2, f_3, f_4]
Solution$_1$	[18.55,334.36,16.94,29.53]
Solution$_2$	[24.24,335.56,19.63,14.35]
Solution$_3$	[18.78,331.72,16.91,37.06]
Solution$_4$	[21.75,327.77,17.99,35.21]
Solution$_5$	[19.67,330.84,16.97,18.85]
Solution$_6$	[18.88,334.08,17.09,23.63]
Solution$_7$	[20.08,329.16,17.70,20.91]

5.3 Comparisons of the MOEA-based rescheduling methods

5.3.1 Performance measures

Four popular metrics are employed to evaluate the performances of MOEA-based methods. The first one is the *hypervolume ratio* (*HVR*) [41]. The hypervolume metric *HV* measures the size of the objective space dominated by the obtained non-dominated front *PF_{known}* [65], and *HVR* is the ratio of *HV* and the hypervolume of the reference Pareto front *PF_{ref}* . The second one is the *Generational Distance* (*GD*), which measures how far *PF_{known}* from *PF_{ref}* is [42]. The third one is *Spacing* which measures the distance variance of neighbouring vectors in *PF_{known}* [37]. The smaller *Spacing* is, the better the distribution uniformity of *PF_{known}* is. The fourth one is *Spread*, which measures the extent of spread achieved among the obtained solutions and the non-uniformity in the distribution of *PF_{known}* . The definition of *Spread* in [10] was used for bi-objective problems. As to problems with three or more objectives, a modified *Spread* is given in Eq. (17):

$$Spread = \frac{\sum_{j=1}^{N\_o} df_j + \sum_{i=1}^{n_{PF}} \left| d_i^{'} - \overline{d^{'}} \right|}{\sum_{j=1}^{N\_o} df_j + n_{PF} \cdot \overline{d^{'}}} \tag{17}$$

where df_j is the Euclidean distance between the best solution on the j^{th} objective and its nearest solution in *PF_{known}* (i.e., the boundary solution), n_{PF} is the number of vectors in *PF_{known}*, $d_i^{'}$ is the Euclidean distance from the i^{th} vector of *PF_{known}* to its nearest neighbour in *PF_{known}*, and $\overline{d^{'}}$ is the mean of all $d_i^{'}$. A wide and uniform spread of solutions in *PF_{known}* will result in a small value of *Spread*.

Because the true Pareto front is unknown in MODFJSSP, *PF_{ref}* is obtained by merging the solutions found during all the independent runs of three MOEA-based methods, and determine the non-dominated solutions from them. The reference point in *HVR* is formed with the worst objective values observed in all the optimization runs.

5.3.2 Performance comparisons of MOEA-based rescheduling methods

Performances of the Pareto fronts produced by the three MOEA-based methods were compared. First, results in the initial static FJSSP are given. Then the overall performances of the algorithms across the rescheduling points in the dynamic job shop are presented.

I) Performance comparisons in the initial 10×10 static FJSSP

At the initial time $t_0 = 0$, there were totally 50 operations from 10 jobs to be processed on 10 machines (data is provided in Appendix A). Three objectives of make-span, tardiness and maximal machine workload were to be optimized. 30 independent runs of each MOEA were performed, and the generated Pareto fronts of each run were recorded for statistical tests.

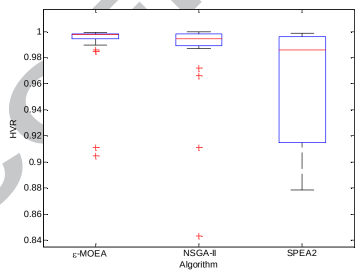
The performance indicators of three MOEA-based methods are shown in Fig. 7. All the objective values of Pareto fronts obtained by each algorithm were normalized by the maximum and minimum value found on the corresponding objective. To significantly compare all the algorithms, Wilcoxon signed-rank tests with significance level of 0.05 were employed and the results were given in Table 5. When the statistical test indicated that there was significant difference between *A* and *B*, the effect size which quantifies how large the difference is between two data sets was checked. The effect size of Cohen's d [7] was used, and it was calculated using the pooled standard deviation [40]:

$$d = \frac{(mean_A - mean_B)}{\sqrt{\frac{std_A^2 + std_B^2}{2}}} \tag{18}$$

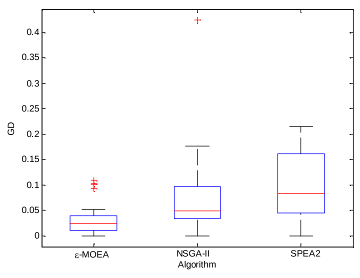
where $mean_{ii}$ denotes the average value on the considered metric obtained by algorithm *ii* in the 30 independent runs, and std_{ii} denotes the corresponding standard deviation, $ii \in \{A, B\}$.

Average CPU time consumed by each of the three methods, and the corresponding standard deviation in the 30 runs in the initial 10×10 static FJSSP are listed in Table 6.

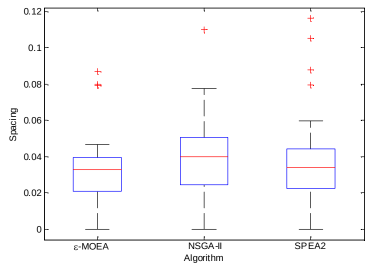
It can be seen that *HVRs* produced by ε-MOEA and NSGA-II are significantly better than that of SPEA2. The values of *GD* and *Spread* obtained by ε-MOEA are significantly better than those of NSGA-II and SPEA2. In terms of *Spacing*, there is no significant difference among the three methods, but the standard deviation of *Spacing* obtained by ε-MOEA is slightly smaller than those obtained by NSGA-II and SPEA2. In addition, the computational cost of ε-MOEA is much smaller than that of NSGA-II and SPEA2. The experimental results show that ε-MOEA can produce a set of scheduling solutions with better convergence and distribution within a shorter period of time. Thus, it is the most competitive algorithm for the initial 10×10 static FJSSP studied in this paper.



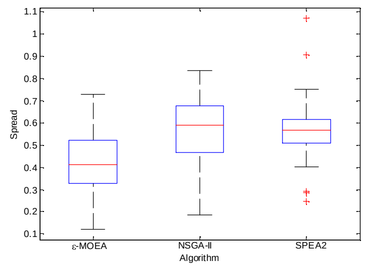
(a) *HVR*



(b) *GD*



(c) *Spacing*



(d) *Spread*

Fig. 7. Performances in the initial static FJSSP. (*HVR is* to be maximized, and *GD*, *Spacing* and *Spread* are to be minimized)

Table 5 Effect size and statistical tests of performances obtained by three MOEA-based methods in the initial static FJSSP. The sign of '+/−/≡' in A vs. B indicates that according to the metric considered, algorithm A is significantly better than B, significantly worse than B, or there is no significant difference between A and B based on the Wilcoxon signed-rank test with the significance level of 0.05. The *p*-values are included in the parentheses. If there is no significant difference between two algorithms, the effect size is not given and '− −' is used.

		HVR	*GD*	*Spacing*	*Spread*
ε-MOEA vs. NSGA-II	effect size	− −	0.7109	− −	0.9428
	(*p*-value) sign	(0.2210) ≡	(3.3173E-004) +	(0.2059) ≡	(0.0020) +
ε-MOEA vs. SPEA2	effect size	0.8033	1.2426	− −	0.9306
	(*p*-value) sign	(0.0039) +	(8.9187E-005) +	(0.5999) ≡	(0.0068) +
SPEA2 vs. NSGA-II	effect size	0.5634	− −	− −	− −
	(*p*-value) sign	(0.0026) −	(0.1020) ≡	(0.3493) ≡	(0.7813) ≡

Table 6 CPU time comparisons of three MOEA-based methods in the initial 10×10 static FJSSP.

	Running Time (sec)	
	mean	std.
ε-MOEA	28.3966	0.5884
NSGA-II	65.9488	3.2107
SPEA2	161.1220	15.0371

II) Performance comparisons in the dynamic job shop

This section gives the overall performance comparisons of the three MOEA-based rescheduling methods during the dynamic process of the job shop. In this experiment, one complete run of the dynamic job shop instance (data is provided in Appendix A) was performed, and there were totally 220 rescheduling points. At each rescheduling point, 30 independent runs of each MOEA were performed. In this way, it can be ensured that three MOEA-based methods were compared in the same job shop scenarios. All the solution sets obtained in 30 runs of three methods were merged, and the non-dominated solutions were determined from them to form the reference Pareto front at the specific rescheduling point. Then the performance values (*HVR*, *GD*, *Spacing*, *Spread*) of each MOEA in each run at each rescheduling point could be obtained, and the overall performances across the rescheduling points in MODFJSSP were compared.

Firstly, the average performance values across rescheduling points were calculated. For each run of each MOEA, the performance values were averaged over all the rescheduling points during the interval of last 1000 new job arrivals. Thus, 30 mean values could be obtained for each MOEA on each performance metric (because the number of runs was 30 here). Then the 30 mean values were averaged, and the corresponding mean standard deviation across rescheduling points was calculated by means of the pooled standard deviation as follows [40]:

$$std_{pooled} = \sqrt{\frac{std_1^2 + \cdots + std_{jj}^2 + \cdots + std_{n_1}^2}{n_1}} \tag{19}$$

where n_1 is the number of runs (30 in our case), std_{jj} is the standard deviation across rescheduling points in the jjth run. The obtained average performances across rescheduling points are shown in Table 7(a). It can be seen that ε-MOEA achieves the best mean value on the metrics *HVR*, *GD* and *Spacing*, and SPEA2 has the best *Spread* value. The mean standard deviation obtained by each method is rather close to each other.

Secondly, with the aim of investigating to what extent the overall performances of different methods differ from each other, they were compared by the Wilcoxon signed-rank test with the significance level of 0.05, and the effect size of Cohen's *d* was checked when there was a significant difference between them. The results are presented in Table 7(b). It can be found that *HVR* and *GD* obtained by ε-MOEA are significantly better than those of NSGA-II and SPEA2. Especially, the effect size on *GD* is larger than that on *HVR*, which shows that ε-MOEA is very good at obtaining better *GD* performance. These results indicate that ε-MOEA has the best convergence performance among the three MOEA methods. In terms of the distribution metrics, *Spacing* produced by ε-MOEA is significantly better than that of NSGA-II, and there is no significant difference between ε-MOEA and SPEA2. However, the effect size of ε-MOEA versus NSGA-II is slightly larger than that of SPEA2 versus NSGA-II. For *Spread*, SPEA2 is significantly better than ε-MOEA and NSGA-II, which is consistent with the result of average performances. The reason for the relatively poor performance of ε-MOEA on *Spread* is that this metric quantifies the extent of spread achieved among the obtained solutions and it is biased towards the boundary solutions. Since ε-MOEA employs ε-dominance in the archive update procedure, it will not be usually possible to obtain the extreme corners of the Pareto-optimal front [9].

The mean CPU time of three MOEA-based methods at each rescheduling point is plotted in Fig. 8, which shows that the computational time cost by ε-MOEA is much smaller than that of NSGA-II and SPEA2 throughout all the rescheduling points.

From experimental results in both initial static FJSSP and the dynamic flexible job shop, it can be concluded that the ε-MOEA-based rescheduling method is most competitive among the three algorithms for evolving efficient non-dominated scheduling solutions in the problem instance studied in this paper, since it can achieve the best convergence and find a good distribution of solutions in a much less computational time than the other two MOEA-based methods (i.e., NSGA-II and SPEA2). Reasons for a better compromise of ε-MOEA between convergence, diversity and computational efficiency are [9]: (1) it adopts careful strategies in choosing mating partners from two co-evolving populations (an EA population and an archive population) and in accepting the generated offspring to each population; (2) The ε-dominance criterion used can help reduce the cardinality of Pareto-optimal region and maintain a good distribution of the solutions, which is useful when solving the many-objective problems; and (3) its parent population is updated in a steady-state manner, thus it has higher chances of producing good offspring solutions, and high computational speed can be achieved. In contrast, NSGA-II and SPEA2 are two generational algorithms. The nearest neighbour density estimation approach makes SPEA2 find a well distributed set of solutions but at the expense of very large computational time, and it has been validated that the crowding operator in NSGA-II is not adequate in keeping a good distribution of solutions in many-objective problems [9].

Table 7 Comparisons of the overall performances of MOEA methods across rescheduling points in the MODFJSSP instance

(a) Average performances across rescheduling points (*HVR* is to be maximized, and *GD*, *Spread* and *Spacing* are to be minimized. Here, 'std.' is short for the pooled standard deviation. The best of mean value and std. on each metric is in red/yellow (dark / light grey)).

Average Performances across Rescheduling Points		*HVR*	*GD*	*Spacing*	*Spread*
ε- MOEA	mean	0.9206	0.0648	0.0621	0.6294
	std.	0.1568	0.1117	0.0873	0.3029
NSGA-II	mean	0.9068	0.1004	0.0683	0.6233
	std.	0.1638	0.1285	0.0955	0.3006
SPEA2	mean	0.9149	0.0786	0.0641	0.5813
	std.	0.1597	0.1097	0.0868	0.3109

(b) Effect size and statistical tests of MOEA methods across rescheduling points

Effect Size and Statistical Tests across Rescheduling Points		*HVR*	*GD*	*Spacing*	*Spread*
ε-MOEA vs. NSGA-II	effect size	0.0866	0.2961	0.0678	– –
	(*p*-value) sign	(0.0020) +	(0.0020) +	(0.0059) +	(0.5566) ≡
ε-MOEA vs. SPEA2	effect size	0.0362	0.1250	– –	0.1568
	(*p*-value) sign	(0.0273) +	(0.0020) +	(0.4316) ≡	(0.0020) −
SPEA2 vs. NSGA-II	effect size	0.0504	0.1826	0.0464	0.1375
	(*p*-value) sign	(0.0020) +	(0.0020) +	(0.0273) +	(0.0020) +

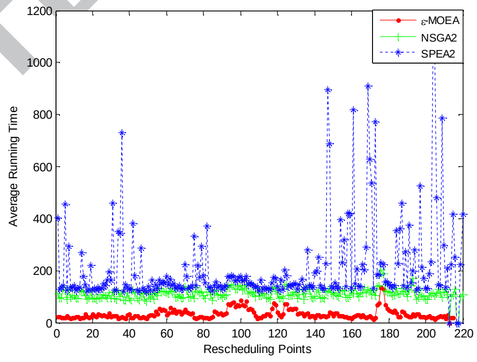


Fig. 8. CPU time comparisons of three MOEA-based methods at each rescheduling point in the MODFJSSP instance.

5.4 Comparisons to existing dynamic scheduling methods

With the aim of further validating the effectiveness of the proposed MOEA-based rescheduling methods in MODFJSSP, we compared them with the commonly used completely reactive scheduling methods, which assigns operations to different machines according to a specific machine assignment rule, and once a machine becomes idle and there are operations in its waiting queue, it chooses the operation with the highest priority to process based on a heuristic priority dispatching rule.

Four popular priority dispatching rules (PDRs) which are Shortest Processing Time (SPT) [31], First-In-First-Out (FIFO), Last-In-First-Out (LIFO) and the random dispatching rule were employed. At each rescheduling point, there may be some operations which need to be assigned to an available machine. For example, the operations of the new arrival jobs, the operations that are waiting in the queue of the broken machine, and some previously unavailable operations become available again due to the machine repairs. We considered three machine assignment rules (MARs) for such operations. The first one finds the available machine with the minimum processing time for each operation, then fixes that assignment, and updates the machine workload. The second one assigns each operation to its alternative machine which has the minimum workload currently. The third one chooses an alternative machine at random for each operation. We call them MAR1, MAR2, and MAR3 in short.

Combinations of four PDRs and three MARs were evaluated. Since the principle of the completely reactive scheduling (make local decisions once a machine becomes free) is different from that of the proposed MOEA-based methods (make global decisions and revise the schedule at each rescheduling point), the objectives in Eq. (2) - (5) which are defined for each rescheduling point are not applicable to the completely reactive scheduling methods. Thus, we compared three other performances which are finishing time of all the jobs, weighted average job tardiness, and average machine workload during the whole running process of the dynamic flexible job shop. 30 independent runs of the dynamic job shop instance (data is provided in Appendix A) were replicated for each combination of the PDRs and MARs, and also for the three MOEAs. Average results are listed in Table 8.

It can be found that compared to the completely reactive scheduling, the proposed MOEA-based rescheduling methods make the job shop finish processing all the jobs in a much smaller time. Meanwhile, they have a much less average delay to the due dates of the jobs. In terms of the average machine workload, the combinations containing MAR1 achieve better performance than MOEA-based methods. The reason is that MAR1 always assigns an operation to its alternative machine with the minimum processing time, which tends to reduce the total workload of all the machines. But on the other hand, it may lead to long waiting queues in specific machines, and idleness of the others, which results in a long finishing time of all the jobs and a large tardiness. Considering three MOEA-based methods, the performance vector obtained by ε-MOEA dominates that generated by NSGA-II or SPEA2. Overall, compared to the traditional completely reactive scheduling, the proposed MOEA-based rescheduling methods can improve the dynamic job shop efficiency to a large extent.

Table 8 Comparisons of the MOEA-based rescheduling methods against the existing dynamic scheduling methods

Scheduling Methods / Performance	Finishing Time of All the Jobs	Weighted Average Job Tardiness	Average Machine Workload
MAR1 + SPT	1436.55 ± 2.40E-13	38.34 ± 0.77	372.88 ± 1.44
MAR2 + SPT	1440.15 ± 0.00033	66.98 ± 0.68	406.11 ± 1.90
MAR3 + SPT	1437.30 ± 1.04	83.29 ± 2.46	602.61 ± 3.91
MAR1 + FIFO	1436.56 ± 2.40E-13	46.39 ± 1.06	385.60 ± 1.91
MAR2 + FIFO	1440.21 ± 0.054	113.38 ± 1.49	436.42 ± 1.83
MAR3 + FIFO	1428.00 ± 14.74	183.44 ± 10.53	642.49 ± 6.29
MAR1 + LIFO	1436.56 ± 2.40E-13	43.30 ± 1.09	386.18 ± 1.91
MAR2 + LIFO	1440.15 ± 0	92.11 ± 2.62	426.99 ± 3.68
MAR3 + LIFO	1612.42 ± 246.40	163.16 ± 7.54	639.83 ± 6.40
MAR1 + random dispatching rule	1436.55 ± 0.032	44.51 ± 0.89	384.68 ± 1.91
MAR2 + random dispatching rule	1439.34 ± 1.76	110.21 ± 2.70	434.59 ± 3.90
MAR3 + random dispatching rule	1427.97 ± 19.92	179.72 ± 9.50	639.70 ± 6.74
ε-MOEA based rescheduling method	875.25 ± 0.18	18.43 ± 0.38	404.70 ± 3.15
NSGA-II based rescheduling method	876.31 ± 2.06	18.54 ± 0.38	409.67 ± 3.74
SPEA2 based rescheduling method	880.29 ± 1.09	18.73± 0.48	407.61± 3.76

5.5 Comparisons to existing static algorithms

With the aim to observe the consequence caused by using the existing static algorithms to solve MODFJSSP, our ε-MOEA-based rescheduling method is compared with two classical static MOEAs (we call them static NSGA-II and static SPEA2 here). It is assumed that MODFJSSP can be divided into some static FJSSP, and the two static MOEAs are run many times to solve each static problem. Here, static NSGA-II and static SPEA2 are different from the ones employed in Section 5.3.2 in that: (1) the initial population are generated at random and no heuristic strategies are incorporated at each rescheduling point, so the two MOEAs regenerate a new schedule from scratch; and (2) stability is not considered as an objective and only three shop efficiency related objectives (make-span, tardiness, the maximal machine workload) are optimized. On the other hand, to make static NSGA-II and static SPEA2 solve each static FJSSP, individual representations in Section 4.2.1 and two problem specific variation operators designed in Section 4.2.5 are used in static NSGA-II and static SPEA2.

The comparison method in the dynamic job shop is similar to that used in Section 5.3.2. Average performances, effect size of Cohen's *d*, and Wilcoxon signed-rank tests of our ε-MOEA-based rescheduling method, static NSGA-II and static SPEA2 across rescheduling points are given in Table 9. It can be seen from Table 9(a) that ε-MOEA achieves the best mean value on all the four metrics *HVR*, *GD*, *Spacing*, and *Spread*. It can be found from Table 9(b) that *HVR* and *GD* obtained by ε-MOEA are significantly better than those of static NSGA-II and static SPEA2. These results indicate that ε-MOEA has much better convergence performance than static MOEAs. In terms of the distribution metrics, *Spacing* produced by ε-MOEA is significantly better than that of static NSGA-II, and there is no significant difference between ε-MOEA and static SPEA2. However, the effect size of ε-

MOEA versus static NSGA-II is slightly larger than that of static SPEA2 versus NSGA-II. For *Spread*, ε-MOEA is significantly better than that of static NSGA-II and static SPEA2.

From above comparison results, it can be concluded that the proposed heuristic population initialization and simultaneous consideration of the stability and efficiency in the job shop are very useful for solving MODFJSSP, since they can capture the correlations between the sequence of multi-objective FJSSPs and guide the search of ε-MOEA so that our ε-MOEA-based rescheduling method can find the solution to the new problem effectively and efficiently. In contrast, each static FJSSP is regarded as independent by static NSGA-II and static SPEA2 and they just search the large decision space from scratch each time, so their searching efficiency is highly reduced.

Table 9 Comparisons of the overall performances of ε-MOEA-based rescheduling method, static NSGA-II and static SPEA2 across rescheduling points in the MODFJSSP instance

(a) Average performances across rescheduling points (*HVR* is to be maximized, and *GD*, *Spread* and *Spacing* are to be minimized. Here, 'std.' is short for the pooled standard deviation. The best of mean value and std. on each metric is in red/yellow (dark / light grey)).

Average Performances across Rescheduling Points		*HVR*	*GD*	*Spacing*	*Spread*
ε-MOEA-based rescheduling method	mean	0.9206	0.0648	0.0621	0.6294
	std.	0.1568	0.1117	0.0873	0.3029
static NSGA-II	mean	0.8375	0.2339	0.1041	0.7134
	std.	0.1644	0.1898	0.0929	0.3792
static SPEA2	mean	0.8586	0.1673`	0.0641	0.6901
	std.	0.1608	0.1057	0.0893	0.3493

(b) Effect size and statistical tests of ε-MOEA-based rescheduling methods, static NSGA-II and static SPEA2 across rescheduling points

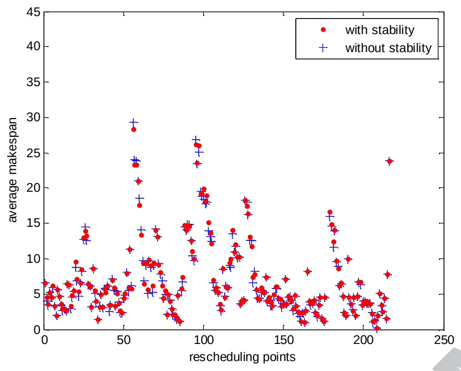
Effect Size and Statistical Tests across Rescheduling Points		*HVR*	*GD*	*Spacing*	*Spread*
ε-MOEA vs. NSGA-II	effect size (*p*-value) sign	0.4999 (0.0020) +	1.0731 (0.0039) +	0.4387 (0.0054) +	0.3626 (0.0098) +
ε-MOEA vs. SPEA2	effect size (*p*-value) sign	0.3729 (0.0020) +	0.6524 (0.0488) +	− − (0.1602) ≡	0.3093 (0.0020) +
SPEA2 vs. NSGA-II	effect size (*p*-value) sign	− − (0.1602) ≡	0.6260 (0.0137) +	0.2691 (0.0059) +	− − (0.4922) ≡

5.6 Further analysis

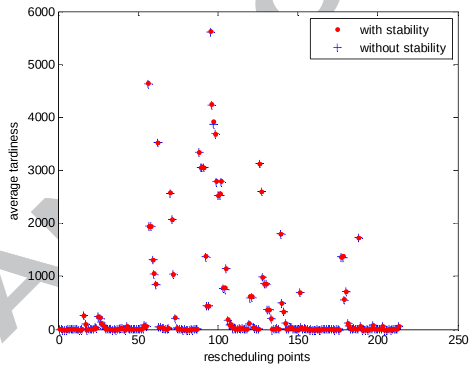
In this section, we will investigate the effectiveness of some strategies employed in the proposed MOEA-based rescheduling methods. All the experiments in this section were performed in the same job shop instance as before.

5.6.1 Influence of the stability objective

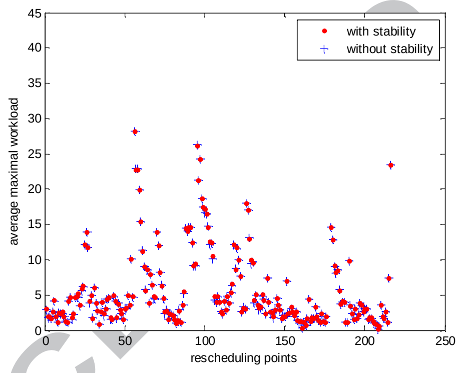
In this section, we will study the impact that the stability objective has on the performance of the MOEA-based methods. Results from two types of approaches were compared. There were four objectives in *approach*$_1$, and in *approach*$_2$ only three objectives on the shop efficiency were considered. Both approaches used the three MOEA-based methods (ε-MOEA, NSGA-II, and SPEA2). First, at each rescheduling point, 30 independent runs of each MOEA in *approach*$_1$ were performed. All the solution sets obtained in each run of each MOEA were merged, and the non-dominated solutions at each rescheduling point were determined. Then they were averaged along each of the four objectives, and plotted in Fig. 9. The same method was also adopted for *approach*$_2$ (only three objectives on efficiency were considered in the non-domination comparisons in MOEA-based methods. The stability values were evaluated only for the obtained non-dominated solutions to compare with *approach*$_1$). The reason for using the average values along each objective is that we expect to check the overall performance improvement (or deterioration) on individual objectives by using the stability as one of the multiple objectives.



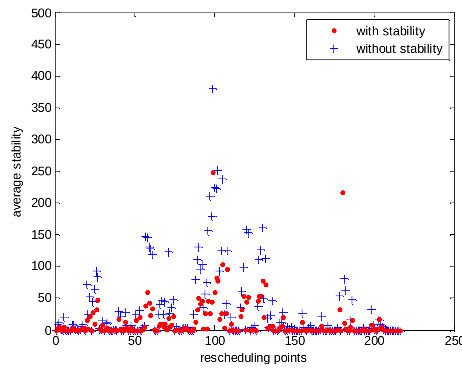
(a) Average Make-span



(b) Average Tardiness



(c) Average Workload



(d) Average Stability

Fig. 9. Average performance values of the non-dominated solutions obtained at each rescheduling point. (All the four performances are to be minimized. + means solutions found by *approach*$_1$ (with stability), and · denotes solutions produced by *approach*$_2$ (without stability))

At the rescheduling point t_l, the quantitative improvement (or deterioration) of *approach*$_1$ (with stability) over *approach*$_2$ (without stability) on each objective is calculated as follows:

$$Imp_i(t_l) = -\frac{(Avg\_f_i^{approach_1}(t_l) - Avg\_f_i^{approach_2}(t_l))}{Avg\_f_i^{approach_2}(t_l)} \times 100\% \tag{20}$$

where $Avg\_f_i^{approach_1}(t_l)$ and $Avg\_f_i^{approach_2}(t_l)$ represent the average values of the non-dominated solutions obtained by *approach*$_1$ and *approach*$_2$ on the objective f_i at t_l, respectively. Since all the objectives considered are to be minimized, we use a negative sign in Eq. (20). The overall improvement (or deterioration) on each objective f_i is the average value of $Imp_i(t_l)$

over all the rescheduling points during the interval of last 1000 new job arrivals, which are listed in Table 10. It can be seen that compared to *approach*$_2$ (without stability), *approach*$_1$ (with stability) improves the system stability significantly with a small sacrifice in the shop efficiency. The improvement in stability is much more than the deterioration in efficiency, which suggests that if we solve MODFJSSP by simultaneously considering stability and efficiency, we will have a high chance of obtaining more stable solutions without severely affecting the efficiency. This result is very practical since stability is an important factor in the real-world job shop.

Table 10 Performance improvement (or deterioration) of *approach*$_1$ (with stability) over *approach*$_2$ (without stability) on each objective (the positive value means improvement, and the negative value means deterioration)

Objective	Make-span	Tardiness	Maximum workload	Stability
Improvement of *approach*$_1$ over *approach*$_2$	-2.03%	-12.24%	-1.44%	36.76%

5.6.2 Influence of the heuristic initialization strategy

As indicated in Section 4.2.4, at each rescheduling point, to guide the search of the ε-MOEA-based rescheduling method, three heuristic methods which are utilizing dynamic event features, history information and machine assignment rules are incorporated in the population initialization. Here, we will investigate the influence of each method on the performance of ε-MOEA. The comparison method in the dynamic job shop is similar to that used in Section 5.3.2. Average performances, effect size of Cohen's *d*, and Wilcoxon signed-rank tests of different initialization methods across rescheduling points are given in Table 11, where *Proposed_Ini* is the ε-MOEA-based method using the initialization strategy proposed in this paper. The other four methods are similar to *Proposed_Ini*, except that *Norepair_Ini* and *Nohistory_Ini* do not make use of the dynamic event characteristics and the history information, respectively. *Norepair_nohistory_Ini* only employs the machine assignment rules and random individuals. There is no heuristic method in *Random_Ini*, and all the initial individuals are generated at random which regenerates a new schedule from scratch.

Table 11 Comparisons of the overall performances of different initialization methods across rescheduling points in MODFJSSP

(a) Average performances across rescheduling points (*HVR* is to be maximized, and *GD*, *Spread* and *Spacing* are to be minimized. Here, 'std.' is short for the pooled standard deviation. The best of mean value and std. on each metric is in red/yellow (dark / light grey)).

Average Performances across Rescheduling Points		*HVR*	*GD*	*Spacing*	*Spread*
Proposed_Ini	mean	0.9206	0.0648	0.0621	0.6294
(with MA rules)	std.	0.1568	0.1117	0.0873	0.3029
Norepair_Ini	mean	0.9155	0.0689	0.0616	0.6258
(with MA rules)	std.	0.1761	0.1136	0.0898	0.2983
Nohistory_Ini	mean	0.9186	0.0680	0.0601	0.6246
(with MA rules)	std.	0.1581	0.1245	0.0838	0.3046
Norepair_nohistory_Ini	mean	0.8858	0.0766	0.0624	0.6171
(with MA rules)	std.	0.1930	0.1206	0.0868	0.3023
Random_Ini	mean	0.8464	0.1214	0.0653	0.6516
(without MA rules)	std.	0.2385	0.1798	0.0917	0.2979

(b) Effect size and statistical tests of different initialization methods across rescheduling points

Effect Size and Statistical Tests across Rescheduling Points		*HVR*	*GD*	*Spacing*	*Spread*
Proposed_Ini vs. *Norepair_Ini*	effect size (*p*-value) sign	0.1526 (0.0273) +	− − (0.0840) ≡	− − (0.2840) ≡	− − (0.6953) ≡
Proposed_Ini vs. *Nohistory_Ini*	effect size (*p*-value) sign	− − (0.0645) ≡	− − (0.0840) ≡	− − (0.2324) ≡	− − (0.3750) ≡
Proposed_Ini vs. *Norepair_nohistory_Ini*	effect size (*p*-value) sign	0.1982 (0.0020) +	0.1019 (0.0020) +	− − (0.7695) ≡	0.0406 (0.0039) −
Proposed_Ini vs. *Random_Ini*	effect size (*p*-value) sign	0.3680 (0.0020) +	0.3781 (0.0020) +	− − (0.0645) ≡	0.0741 (0.0020) +
Norepair_nohistory_Ini vs. *Random_Ini*	effect size (*p*-value) sign	0.1817 (0.0020) +	0.2923 (0.0020) +	− − (0.0840) ≡	0.1151 (0.0020) +

In terms of convergence, *Proposed_Ini* achieves the best average performances (mean and std. value in Table 11(a)) on *HVR* and *GD*. According to Table 11(b), although *GD* values of *Proposed_Ini*, *Norepair_Ini* and *Nohistory_Ini* are not significantly different, *Proposed_Ini* is significantly better than *Norepair_nohistory_Ini* on *HVR* and *GD*, and also significantly better than *Norepair_Ini* on *HVR*, which indicates that the combined use of the dynamic characteristics and history information in initialization can help improve the convergence performance of the ε-MOEA-based rescheduling method a lot. It can be found that *Norepair_nohistory_Ini* is significantly better than *Random_Ini* on both *HVR* and *GD*, which shows that incorporating the machine assignment rules in initialization is also helpful in improving the convergence. Moreover, the effect sizes of *Proposed_Ini* vs. *Random_Ini* on *HVR* and *GD* are larger than that of *Norepair_nohistory_Ini* vs. *Random_Ini*, which further validates the effectiveness of making use of dynamic characteristics and history information.

With respect to distributions of non-dominated solutions, there is no significant difference among the five initialization methods for *Spacing*. For *Spread*, *Norepair_nohistory_Ini* achieves the best performance, which demonstrates that employing machine assignment rules in initialization can help improve the spread performance, but the incorporation of dynamic characteristics and history information may deteriorate the spread a little. The reason is that *Proposed_Ini* uses the history solution, the schedule repair solution and their variants as parts of the initial population, which can help speed the convergence, but on the other hand may limit the search space explored by the algorithm. However, considering *Proposed_Ini* vs. *Norepair_nohistory_Ini* in Table 11(b), *Proposed_Ini* improves *HVR* and *GD* (effect sizes are 0.1982 and 0.1019, respectively) much more than it degrades *Spread* (effect size is 0.0406).

Overall, by making use of the dynamic event features, history information and machine assignment rules, the proposed heuristic initialization method at each rescheduling point can make ε-MOEA adapt to the new environment quickly and find the solution to the new problem efficiently, since compared to the random initialization, it is very effective in improving the convergence performance of ε-MOEA, and it can also maintain a good distribution of non-dominated solutions.

5.6.3 Utility of the dynamic decision-making method

To demonstrate the utility of the decision-making procedure presented in Section 4.3, we used six different preferences over the four objectives. At each rescheduling point t_l, suppose the objective vector of the final selected solution is $(f_i^*(t_l))_{1\times 4}$, and the best and worst values of each objective among the obtained non-dominated solutions are $(f_i^{\min}(t_l))_{1\times 4}$ and $(f_i^{\max}(t_l))_{1\times 4}$. Then the proportions $Pro_i(t_l) = (f_i^*(t_l) - f_i^{\min}(t_l)) / (f_i^{\max}(t_l) - f_i^{\min}(t_l)) \times 100\%$ ($i = 1, 2, 3, 4$) are calculated. 30 simulation replications were performed, and the average values of $Pro_i(t_l)$ across different rescheduling points/simulation replications in the six preference cases are listed in Table 12. The first case of *w* = [0.3512, 0.1887, 0.1089, 0.3512] is the preference employed in all previous simulations as indicated in Section 4.3. The second case of *w* = [0.25, 0.25, 0.25, 0.25] reflects the equal importance of four objectives. The last four rows in Table 12 emphasize four extreme cases which give 100% importance to one objective and do not care about others. In the first case, the performances of objectives f_1 and f_4 are good due to more preferences over them. In the second case, similar performances are obtained among the four objectives because equal importance is assigned to each. In the last four cases, the extreme solution with the best value on the objective given 100% importance is always chosen. These results suggest that the proposed decision-making procedure is able to select a solution which fits into the DM's preferences from a set of trade-off solutions.

Table 12 Comparisons of results obtained by the dynamic decision-making method over six different preferences (A smaller value of average proportion indicates a better performance on f_i, i=1, 2, 3, 4)

Different preferences (weight vectors)	Average proportion on f_1	Average proportion on f_2	Average proportion on f_3	Average proportion on f_4
[0.3512,0.1887,0.1089,0.3512]	0.1935	0.2993	0.3529	0.1676
[0.25,0.25,0.25,0.25]	0.2742	0.2368	0.2265	0.2644
[1 0 0 0]	0	0.5617	0.4668	0.5343
[0 1 0 0]	0.5524	0	0.5185	0.5707
[0 0 1 0]	0.5218	0.4942	0	0.5633
[0 0 0 1]	0.5611	0.5078	0.4779	0

6 Conclusions

This paper proposed an MOEA-based rescheduling method to regenerate new schedules in respond to random events in dynamic flexible job shops. Our first contribution is the construction of a dynamic multi-objective optimization model for MODFJSSP. In the model, considering the updated job shop state at each rescheduling point, four objectives with respect to both the shop efficiency and stability are optimized simultaneously. In addition, six categories of constraints to the search space which change dynamically with the occurrences of real-time events are addressed, and a more sophisticated definition for stability is presented.

Our second contribution is an ε-MOEA-based predictive-reactive scheduling method. The novelty is the employment of heuristic strategies in population initialization so that the method does not regenerate a new schedule from scratch at each rescheduling point. In addition, the proposed method can deal with multiple scheduling policies (operation sequencing and machine assignment) simultaneously as a result of individual representations and two kinds of problem specific variation operators. Experimental results show that our ε-MOEA-based method can achieve much better performances on total make-span and tardiness than combinations of popular dispatching rules with machine assignment rules. Three MOEA-based rescheduling methods (ε-MOEA, NSGA-II and SPEA2) are compared. Results in both static and dynamic flexible job shops show that ε-MOEA is the most competitive approach for evolving efficient non-dominated solutions for MODFJSSP. Statistical tests indicate that it has the best overall performances across rescheduling points in dynamic environments. Moreover, its computational cost is much less than NSGA-II and SPEA2. In addition, Pareto front obtained at each rescheduling point provides much better knowledge about various trade-offs in the objective space for the DM to make an informed decision, which cannot be achieved by combinations of existing scheduling rules or by the methods using a weighted sum approach. Further analyses show that introduction of the stability objective can improve the system stability much more than it degrades efficiency, and heuristic initialization strategies are very effective in improving the convergence of the ε-MOEA-based method.

Our third contribution is the design of a dynamic decision making procedure. Simulation results demonstrate that the proposed method can select a solution which corresponds to the DM’s preferences from the trade-off solution set generated at each rescheduling point.

As future work, the proposed MOEA-based rescheduling methods should be applied to more job shop conditions, e.g. different levels of job arrival rates, and other kinds of job shop scenarios with more dynamic events such as variations of processing times. Moreover, other constraints in the real-world job shop must be added, e.g. sequence dependent setup time. Evolution of other scheduling policies such as due-date assignment will also be considered.

Acknowledgement This work was partially supported by the “Outstanding Young and Middle-Aged University Teachers and Presidents Training Abroad Project” in Jiangsu Province, China, and a Scientific Research Fund of Nanjing University of Information

Science and Technology, China (No. S8111127001), through a scholarship to the first author, and by an EPSRC Grant (No. EP/K001523/1) on "Evolutionary Computation for Dynamic Optimization in Network Environments". This work was done while the first author was with CERCIA, School of Computer Science, University of Birmingham, UK.

Appendix A. The dynamic flexible job shop instance used in our experiments

A summary of the parameters used to design the flexible job shop instance in our experiments is presented in Table A.1.

Table A.1 Summary of the parameters used in the design of experiments

	Characteristics	Specifications
Job Shop	Size	m=10 machines
	Shop Utilization	0.8

	Machine Breakdowns	MTBF=U[100, 300]
		MTTR=U[20, 120]

	Distribution of TBF	Exponential with the mean of MTBF
	Distribution of TTR	Exponential distribution with the mean of MTTR
Jobs	# of operations in each job	U[1, m]
	# of alternative machines to process an operation	U[1, m]
	Distribution of the processing time of each operation	Exponential distribution with the mean of 1
	Mean time between job arrivals (MTBJA)	0.625
	Distribution of the time between new job arrivals	Exponential distribution with the mean of MTBJA
	Job release policy	Immediate
	Distribution of the tightness factor in the due date	Normal distribution with the mean of 1.5 and variance of 0.5.
	weights	20% of the jobs are 1, 60% are 2, and 20% are 4
Simulation Analysis	Warm-up period	240 jobs
	Simulation period	1000 jobs

* # denotes the number, and U(a, b) denotes a number generated uniformly at random from the interval of [a, b]

The problem instance used in our experiments is generated using the parameters given in Table A.1. The data is described below.

A.1 The initial 10×10 static FJSSP

The initial number of available machines is 10, and the initial number of jobs is also 10.

The due dates, weights, numbers of operators and processing times of the initial ten jobs are given in Table A.2 and Table A.3, respectively.

Table A.2 Due dates, weights and numbers of operators of the initial ten jobs

Job number	1	2	3	4	5	6	7	8	9	10
Due date	11.3868	2.5035	2.2035	3.2497	4.8579	8.8373	3.3804	10.7493	0.6042	4.0552
Weights	2	2	2	1	1	2	2	2	2	4
# of operators	6	9	1	4	5	5	5	9	1	5

Table A.3 Processing times of the initial ten jobs

p_{ij}^k	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1 O_{11}	∞	1.5929	0.4713	0.6089	0.9498	0.3568	3.0133	∞	0.3139	∞
O_{12}	0.3683	∞	∞	0.3831	1.0458	1.5021	∞	0.0672	∞	1.2794
O_{13}	1.5514	1.1718	0.3505	0.3004	1.9272	0.8895	∞	0.8923	1.7593	∞
O_{14}	1.2497	2.9106	4.5696	2.4650	0.2619	0.1324	1.2787	0.0984	1.4564	2.3571
O_{15}	∞	0.0204	0.6969	1.8902	0.2029	0.2410	∞	1.7994	3.9043	1.6718
O_{16}	∞	∞	∞	1.5782	∞	0.0405	0.2971	0.8711	2.0635	∞

J_2	O_{21}	0.5818	3.6391	1.9767	1.2716	0.3420	0.0927	0.7917	0.1709	∞	2.7091
	O_{22}	0.7960	4.0006	∞	∞	∞	∞	0.4459	∞	1.3633	2.8225
	O_{23}	0.8205	1.2625	0.7756	5.5463	3.6130	∞	∞	0.1478	2.4985	0.1623
	O_{24}	0.5487	1.2152	1.2999	0.7223	1.6280	0.8747	1.6926	0.2280	0.5254	0.7552
	O_{25}	0.6449	0.7731	∞	0.1760	0.0458	0.0433	∞	1.0591	∞	0.9215
	O_{26}	∞	0.8113	0.7191	∞	∞	∞	∞	∞	∞	3.2824
	O_{27}	0.2813	0.4945	1.5254	1.3661	∞	1.1051	3.9508	1.7132	0.2092	0.6487
	O_{28}	0.2496	0.1919	0.5184	0.0885	0.0206	1.6698	0.3020	1.3353	∞	∞
	O_{29}	0.9233	1.1829	0.1585	0.7979	0.1791	1.4378	0.2417	0.8223	0.3893	∞
J_3	O_{31}	0.9771	1.0099	2.9758	∞	1.3394	0.4705	0.6757	0.8319	0.3779	0.0446
J_4	O_{41}	∞	1.0357	∞	∞	0.7537	∞	0.1385	∞	∞	∞
	O_{42}	∞	1.3208	∞	∞	∞	∞	∞	1.8026	0.3236	∞
	O_{43}	5.6919	∞	0.6640	0.1421	1.2372	0.4312	∞	2.3704	0.0265	∞
	O_{44}	1.2405	∞	0.3849	∞	∞	∞	∞	∞	∞	∞
J_5	O_{51}	3.4409	∞	∞	∞	0.1332	0.5820	0.9092	∞	1.7438	0.9701
	O_{52}	∞	∞	0.0113	∞	∞	∞	0.0739	0.2343	∞	0.3678
	O_{53}	0.2369	∞	∞	∞	0.8359	∞	0.4202	∞	∞	∞
	O_{54}	0.5677	∞	∞	0.2594	0.6281	∞	∞	∞	∞	∞
	O_{55}	∞	0.1008	∞	0.6411	∞	0.2383	∞	0.7075	∞	∞
J_6	O_{61}	0.5516	∞	0.4784	1.5560	∞	∞	∞	1.0649	0.8064	∞
	O_{62}	∞	0.2604	∞	0.4006	∞	0.0954	∞	1.3528	∞	∞
	O_{63}	1.8110	∞	1.4192	∞	∞	∞	∞	∞	∞	∞
	O_{64}	∞	0.1603	∞	∞	0.9575	∞	0.7134	∞	6.3752	∞
	O_{65}	0.0516	0.0442	0.7801	0.9099	0.8919	0.2765	∞	0.6066	1.2999	0.2917
J_7	O_{71}	∞	0.0968	∞	∞	0.3768	1.8021	∞	1.3239	∞	0.8450
	O_{72}	0.8529	∞	0.0520	0.3783	∞	∞	0.3147	1.1333	0.1240	2.5777
	O_{73}	∞	0.6212	∞	∞	∞	0.1327	0.5453	0.0029	∞	∞
	O_{74}	0.8029	∞	∞	∞	∞	∞	∞	∞	∞	∞
	O_{75}	1.0230	∞	2.1644	0.7203	1.1795	0.5878	0.2262	0.5861	1.9826	0.4364
J_8	O_{81}	∞	∞	∞	∞	∞	∞	∞	∞	0.5780	∞
	O_{82}	0.4718	∞	∞	0.6534	0.2169	∞	0.2381	∞	0.4372	0.5237
	O_{83}	∞	∞	∞	∞	∞	0.7083	∞	∞	0.5230	∞
	O_{84}	0.9827	∞	0.1177	∞	∞	∞	0.3315	∞	0.4755	∞
	O_{85}	0.7264	0.1525	0.7264	0.1085	1.1185	∞	0.2082	1.4826	0.3010	1.0829
	O_{86}	1.1461	2.7974	0.0672	0.8124	∞	0.6622	0.6470	3.0756	2.4451	∞
	O_{87}	1.0050	0.0910	1.9157	0.0031	0.0763	1.5097	0.0632	0.2600	1.8009	0.5334
	O_{88}	1.0312	∞	∞	1.2478	∞	∞	∞	0.8171	2.3766	∞
	O_{89}	∞	1.9825	∞	∞	∞	∞	∞	∞	∞	∞
J_9	O_{91}	0.2641	2.1988	2.1656	0.6222	0.0620	∞	1.0994	0.5537	0.1268	2.6729
J_{10}	$O_{10,1}$	0.2214	0.7747	∞	0.3649	0.2022	0.1460	1.3065	3.3038	0.7544	∞
	$O_{10,2}$	∞	∞	∞	∞	2.4277	∞	∞	∞	∞	∞
	$O_{10,3}$	0.2825	∞	∞	∞	∞	∞	0.0503	∞	∞	∞
	$O_{10,4}$	1.7011	0.0960	0.1790	1.3843	∞	0.2157	0.4788	∞	1.3449	∞
	$O_{10,5}$	1.0015	1.0611	0.0505	0.9126	0.1459	0.3104	1.8697	0.9845	1.6904	0.2118

* ∞ denotes the operation cannot be processed by the corresponding machine

A.2 The dynamic environment

Each of the ten machines was assigned different mean time between failure (MTBF) and mean time to repair (MTTR). MTBF and MTTR of ten Machines are given in Table A.4.

Table A.4 MTBF and MTTR of the ten machines

Machine number	1	2	3	4	5	6	7	8	9	10
MTBF	209.0739	129.2992	232.5994	267.6733	225.5985	206.5362	181.7854	192.4251	190.6878	127.2050
MTTR	86.0711	66.8782	62.3333	84.9613	74.8550	54.7553	61.6485	30.5769	94.9891	20.1627

For each machine, the time interval between fails (TBF) and the time to repair (TTR) are assumed to follow an exponential distribution with the mean of MTBF and MTTR, respectively. TBF and TTR of the ten machines are shown in Table A.5 and Table A.6, respectively (Due to the space limitation, only the first ten time intervals are listed).

Table A.5 TBF of the ten machines (The first ten time intervals are listed)

TBF	1	2	3	4	5	6	7	8	9	10	…
M_1	60.1815	341.4744	183.5197	414.6272	293.0613	176.2809	363.8440	215.5021	207.7758	155.5732	…
M_2	11.8945	105.8802	200.6703	70.0234	232.3056	233.1006	60.7633	180.2931	7.5569	371.6800	…
M_3	51.9138	270.0075	155.2262	298.4262	121.1599	494.0131	367.2544	189.6868	183.6290	189.6868	…
M_4	8.1471	1.0067	952.0060	27.8017	347.8132	645.7056	185.1724	449.9411	204.1842	14.4955	…
M_5	3.0207	280.3702	102.2818	488.1254	207.7280	764.9804	211.9077	22.5843	366.7581	226.4633	…
M_6	49.7856	274.1227	571.4564	263.5927	16.2753	211.5855	430.9105	70.5444	295.9994	155.1191	…
M_7	344.2344	33.6427	245.5943	276.7207	135.7361	211.3870	281.6149	160.6632	198.1626	670.8063	…
M_8	125.4887	40.3707	17.5687	288.3028	42.3520	230.3891	158.9897	358.4110	124.0999	37.6119	…
M_9	296.9732	361.4254	358.3395	124.2674	234.0589	456.5467	201.2278	72.9277	133.7395	279.5786	…

Table A.6 TTR of the ten machines (The first ten time intervals are listed)

TTR	1	2	3	4	5	6	7	8	9	10	…
M_1	271.9629	106.7211	246.1584	44.7464	96.7911	166.9721	135.7452	130.6676	94.0891	11.9935	⋯
M_2	33.6536	154.8170	47.7032	24.4864	104.9593	13.6237	87.1751	6.8407	106.5141	42.0916	⋯
M_3	181.4899	30.1449	67.6488	96.0800	6.4491	45.1603	42.7101	167.3514	39.5006	9.5077	⋯
M_4	5.6822	102.9366	27.6108	38.7185	93.9815	90.4160	54.4585	106.1199	5.1494	19.5327	⋯
M_5	202.5777	59.1750	87.6478	2.6217	77.6089	46.8037	19.2985	189.1264	9.1386	31.9112	⋯
M_6	37.3848	138.7584	177.7283	7.8361	120.8651	0.0281	3.4760	161.9860	18.6557	53.7578	⋯
M_7	19.9020	51.1838	1.1964	126.2360	144.2601	13.2633	53.0933	14.4923	21.3423	37.0561	⋯
M_8	5.4266	2.8717	137.4882	5.2254	12.3110	28.8929	79.9172	32.5647	1.0783	8.2916	⋯
M_9	329.4294	18.8285	15.9770	25.9160	47.7416	61.2770	20.4254	70.8265	137.1752	5.5885	⋯
M_{10}	2.2202	9.1865	2.7268	22.9990	31.1073	25.4911	1.0184	10.8054	12.9061	3.7713	⋯

There are totally 1230 new jobs which arrive at the job shop dynamically following the Poisson distribution. The time between new job arrivals is distributed exponentially with the mean of $MTBJA$=0.625 . The time between new job arrivals are listed in Table A.7, and due dates, weights and numbers of operators of the new arrival jobs are listed in Table A.8 (Due to the space limitation, only the information of the first ten out of the 1230 new jobs is given).

Table A.7 Time between new job arrivals (The first ten out of 1230 new job arrivals are listed)

Job number-Job number	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	…
The time between new job arrivals	0.0535	1.7011	0.0981	0.4584	0.9182	0.8659	0.0600	1.5530	0.2514	0.4173	…

Table A.8 Due dates, weights and numbers of operators of the new arrival jobs (the first ten out of 1230 new jobs are listed)

Job number	1	2	3	4	5	6	7	8	9	10	…
Due date	20.2707	4.0544	4.4483	13.1379	19.3717	10.4268	7.6107	19.8581	7.5316	18.0237	…
Weights	2	1	2	2	1	2	2	2	2	4	…
Number of operators	9	3	1	6	7	4	3	10	1	5	…

Processing times of the new arrival jobs are shown in Table A.9 (due to the space limitation, only the processing times of the first three out of the 1230 new jobs are given).

Table A.9 Processing times of the new arrival jobs (the first three out of 1230 new jobs are listed)

p_{ij}^{k}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{11} $O_{11,1}$	∞	0.3052	1.9727	∞	∞	0.8753	0.9758	∞	∞	0.3342
$O_{11,2}$	∞	2.6245	∞	∞	∞	∞	∞	∞	∞	∞
$O_{11,3}$	∞	0.7278	1.0555	∞	∞	∞	∞	∞	∞	∞
$O_{11,4}$	4.2175	∞	0.3228	∞	0.2146	∞	0.1748	0.7009	∞	1.9349
$O_{11,5}$	∞	∞	∞	∞	∞	∞	0.9689	∞	∞	∞
$O_{11,6}$	0.4287	4.4217	0.2878	1.2293	0.3602	0.1132	0.0453	1.3656	1.3325	0.3720
$O_{11,7}$	0.2671	0.5823	0.0608	0.1812	0.2479	3.3421	∞	0.7710	0.2648	1.2957
$O_{11,8}$	∞	∞	∞	∞	0.4674	0.0940	1.0082	∞	∞	∞
$O_{11,9}$	0.7081	1.1823	∞	∞	2.8905	0.3217	∞	0.7101	0.6052	1.2965
J_{12} $O_{12,1}$	∞	∞	∞	0.6595	∞	2.0985	1.0571	0.1704	0.7779	∞
$O_{12,2}$	0.4386	0.6790	∞	0.0374	∞	0.5916	1.4045	0.4547	0.5833	0.0623
$O_{12,3}$	0.2729	0.3201	0.3932	0.1881	0.7544	0.7872	0.1155	0.1735	0.0086	0.7471
J_3 O_{31}	0.7003	2.3103	0.9669	0.8928	0.7072	1.5008	0.1126	0.9777	1.7132	7.1134
… …	…	…	…	…	…	…	…	…	…	…

* ∞ denotes the operation cannot be processed by the corresponding machine

References

[1] R.J. Abumaizar, J.A. Svestka, Rescheduling job shops under random disruptions. *International Journal of Production Research* 35 (7) (1997) 2065-2082.

[2] M.A. Adibi, M. Zandieh, M. Amiri, Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications* 37 (1) (2010) 282-287.

[3] N. Al-Hinai, T.Y. ElMekkawy. Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics* 132 (2) (2011) 279-291.

[4] S. Balin, A robust scheduling method based on a multi-objective immune algorithm. *Information Sciences* 181 (2011) 3551-3569.

[5] G. Chryssolouris, E. Subramanian, Dynamic scheduling of manufacturing job shops using genetic algorithm. *Journal of Intelligent manufacturing,* 12 (2001) 281-293.

[6] L.K. Church, R. Uzsoy, Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing* 5 (3) (1992) 153-163.

[7] J. Cohen, Statistical power analysis for the behavioral sciences (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum. (1988).

[8] K. Deb, S. Chaudhuri, K. Meitinnen. Towards estimating nadir objective vector using evolutionary approaches. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2006) 643-650.

[9] K. Deb, M. Mohan, S. Mishra, Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal. *Evolutionary Computation* 13 (4) (2005) 501-525.

[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6 (2) (2002) 182-197.

[11] K. Deb, N. Udaya Bhaskara Rao, S. Karthik, Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. *In Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science 4403* (2007) 803-817.

[12] Y. Demir, S. Kürşat İşleyen, Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling* 37 (2013) 977-988.

[13] I. Durgun, A.R. Yildiz, Structural design optimization of vehicle components using Cuckoo search algorithm, *Materials Testing* 54 (2012) 185-188.

[14] P. Fattahi, A. Fallahi. Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability. *CIRP Journal of Manufacturing Science and Technology* 2 (2) (2010) 114-123.

[15] P. Fattahi, M. Saidi Mehrabad, F. Jolai, Mathematical modelling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing* 18 (2007) 331-342.

[16] J. Fülöp, Introduction to decision making methods. Working Paper 05-6, Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest. http://academic.evergreen.edu/projects/bdei/documents/decisionmakingmethods.pdf (2005)

[17] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research* 1 (1976) 117-129.

[18] M. Gen, Y. Tsujimura, E. Kubota, Solving job shop scheduling problem by genetic algorithms. I*n Proceedings of IEEE International Conference on Systems, Man, and Cybernetic* (1994) 1577-1582.

[19] T. Hildebrandt, J. Heger, B. Scholz-Reiter, Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. *In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (2010) 257-264.

[20] M.T. Jensen, Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Transactions on Evolutionary Computation* 7 (3) (2003) 275-288.

[21] L. Jourdan, E. Talbi, Combinatorial optimization of stochastic multi-objective problems: an application to the flow-shop scheduling problem. *In Proceedings of Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science 4403* (2007) 457-471.

[22] I. Kacem, S. Hammadi, P. Borne, Approach by localization and multi-objective evolutionary optimization for flexible job shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 32 (1) (2002) 1-13.

[23] D. Lei, Simplified multi-objective genetic algorithms for stochastic job shop scheduling. *Applied Soft Computing* 11 (8) (2011) 4991-4996.

[24] L. Liu, H. Gu, Y. Xi, Robust and stable scheduling of a single machine with random machine breakdowns. *International Journal of Advanced Manufacturing Technology* 31 (2007) 645-654.

[25] C. Mészáros, T. Rapcsák, On sensitivity analysis for a class of decision systems. *Decision Support Systems* 16 (1996) 231-240.

[26] S. Nguyen, M. Zhang, M. Johnston, K. Tan, Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation* 18 (2) (2013) 193-208.

[27] L. Nie, L. Gao, P. Li, X. Li, A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. *Journal of Intelligent Manufacturing* 24 (4) (2013) 763-774.

[28] L. Nie, L. Gao, P. Li, X. Shao, Reactive scheduling in a job shop where jobs arrive over time. *Computers & Industrial Engineering* 66 (2) (2013) 389-405.

[29] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling* 12 (4) (2009), 417-431.

[30] C. Ozguven, L. Ozbakir, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling* 34 (2010) 1539-1548.

[31] S.S. Panwalkar, W. Iskander, A survey of scheduling rules. *Operations Research* 25 (1977) 45-61.

[32] X. Qiu, H.Y.K. Lau, An AIS-based hybrid algorithm with PDRs for multi-objective dynamic online job shop scheduling problem. *Applied Soft Computing* 13 (3) (2013) 1340-1351.

[33] R. Rangsaritratsamee, W.G. Frell, M.B. Kurz, Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering* 46 (2004) 1-15.

[34] C. Raquel, X. Yao, Dynamic multi-objective optimization: a survey of the state-of-the-art. In: Evolutionary computation for dynamic optimisation problems (2013), 85-106. Berlin: Springer-Verlag.

[35] T.L. Saaty, The analytic hierarchy process. New York: McGraw Hill (1980).

[36] T.L. Saaty, L.G. Vargas, Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios. *Mathematical Modelling* 5 (1984) 309-324.

[37] J.R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

[38] A.I. Sivakumar, A.K. Gupta, Online multiobjective Pareto optimal dynamic scheduling of semiconductor back-end using conjunctive simulated scheduling. *IEEE Transactions on Electronics Packaging Manufacturing* 29 (2) (2006) 99-109.

[39] X.N. Shen, Y. Guo, Q.W. Chen, W.L. Hu, A multi-objective optimization evolutionary algorithm incorporating preference information based on fuzzy logic. *Computational Optimization and Applications* 46 (2010) 159-188.

[40] L. Song, L.L. Minku, X. Yao, The impact of parameter tuning on software effort estimation using learning machines. *In Proceedings of the 9th International Conference on Predictive Models in Software Engineering* (2013) 10p.

[41] D.A. Van Veldhuizen, G. B Lamont, Multiobjective evolutionary algorithm test suites. *In Proceedings of the 1999 ACM Symposium on Applied Computing* (1999) 351-357.

[42] D.A. Van Veldhuizen, G. B. Lamont, Multiobjective evolutionary algorithm research: a history and analysis. Technical Report, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Ohio, 1998.

[43] Y. Wang, K. Chin, A linear programming approximation to the eigenvector method in the analytic hierarchy process. *Information Sciences* 181 (2011) 5240-5248.

[44] S. D. Wu, R. H. Storer, P. C. Chang, One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research* 20(1) (1993) 1-14.

[45] S. Yao, Z. Jiang, N. Li, H. Zhang, N. Geng, A multi-objective dynamic scheduling approach using multiple attribute decision making in semiconductor manufacturing. *International Journal of Production Economics* 130 (1) (2011) 125-133.

[46] W.C. Yeh, P.J. Lai, W.C. Lee, M.C. Chuang, Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. *Information Sciences* 269 (2014) 142-158.

[47] A.R. Yildiz, A comparative study of population-based optimization algorithms for turning operations. *Information Sciences* 210 (2012) 81-88.

[48] A.R. Yildiz, Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach. *Information Sciences* 220 (2013) 399-407.

[49] A.R. Yildiz, Hybrid Taguchi-differential evolution algorithm for optimization of multi-pass turning operations. *Applied Soft Computing* 13 (2013) 1433-1439.

[50] A.R. Yildiz, A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations. *Applied Soft Computing* 13 (2013) 1561-1566.

[51] A.R. Yildiz, Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *International Journal of Advanced Manufacturing Technology* 64 (2013) 55-61.

[52] A.R. Yildiz, A new design optimization framework based on immune algorithm and Taguchi method. *Computers in Industry* 60 (2009) 613-620.

[53] A.R. Yildiz, Comparison of evolutionary based optimization algorithms for structural design optimization. *Engineering Applications of Artificial Intelligence* 26 (2013) 327-333.

[54] A.R. Yildiz, Hybrid Taguchi-harmony search algorithm for solving engineering optimization problems. *International Journal of Industrial Engineering Theory, Applications and Practice* 15 (2008) 286-293.

[55] A.R. Yildiz, A novel particle swarm optimization approach for product design and manufacturing, *International Journal of Advanced Manufacturing Technology* 40 (2009) 617-628.

[56] A.R. Yildiz, A new hybrid particle swarm optimization approach for structural design optimization in automotive industry, *Journal of Automobile Engineering* 226 (2012) 1340-1351.

[57] A.R. Yildiz, K.N. Solanki, Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach, *International Journal of Advanced Manufacturing Technology* 59 (2012) 367-376.

[58] A.R. Yildiz, K. Saitou, Topology synthesis of multicomponent structural assemblies in continuum domains, *ASME Journal of Mechanical Design* 133 (2011) 1-9.

[59] A.R. Yildiz, A novel hybrid immune algorithm for global optimization in design and manufacturing, *Robotics and Computer-Integrated Manufacturing* 25 (2009) 261-270.

[60] A.R. Yildiz, An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry, *Journal of Materials Processing Technology*, 50 (2009) 224-228.

[61] A.R. Yildiz, Hybrid immune-simulated annealing algorithm for optimal design and manufacturing, *International Journal of Materials and Product Technology*, 34 (2009) 217-226.

[62] L. Zhang, L. Gao, X. Li, A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem. *International Journal of Production Research* ahead-of-print (2013) 1-16.

[63] F. Zhao, J.Z. Wang, J.B. Wang, J. Jonrinaldi, A dynamic rescheduling model with multi-agent system and its solution method. *Journal of Mechanical Engineering* 58 (2) (2012) 81-92.

[64] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for Multiobjective Optimization. *In Proceedings of the EUROGEN2001 Conference* (2001) 95-100.

[65] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation* 3 (4) (1999) 257-271.

[66] X.Q. Zuo, H.W. Mo, J.P. Wu, A robust scheduling method based on a multi-objective immune algorithm. *Information Sciences* 179 (2009) 3359-3369.

Highlights

- A new mathematical model for multi-objective dynamic scheduling is constructed.
- Novel multi-objective evolutionary rescheduling methods are proposed.
- The proposed methods have been shown to outperform existing approaches.
- Advantages of our stability objective and initialization strategies are validated.
- Our decision-making method can select a suitable solution for user preferences.