Hindawi
Complexity
Volume 2020, Article ID 2862186, 17 pages
https://doi.org/10.1155/2020/2862186

Research Article

Modelling and Solving Rescheduling Problems in Dynamic Permutation Flow Shop Environments

Pablo Valledor,[1] Alberto Gomez ,[2] Paolo Priore,[2] and Javier Puente[2]

[1] *ArcelorMittal Inc., Global R&D Asturias, Gijón, Spain*
[2] *University of Oviedo, Department of Business Administration, Polytechnic School of Engineering, 33203 Gijón, Spain*

Correspondence should be addressed to Alberto Gomez; albertogomez@uniovi.es

Received 7 December 2019; Revised 25 May 2020; Accepted 18 June 2020; Published 24 July 2020

Academic Editor: Roberto Natella

The aim of this paper is to analyse, model, and solve the rescheduling problem in dynamic permutation flow shop environments while considering several criteria to optimize. Searching optimal solutions in multiobjective optimization problems may be difficult as these objectives are expressing different concepts and are not directly comparable. Thus, it is not possible to reduce the problem to a single-objective optimization, and a set of efficient (nondominated) solutions, a so-called Pareto front, must be found. Moreover, in manufacturing environments, disruptive changes usually emerge in scheduling problems, such as machine breakdowns or the arrival of new jobs, causing a need for fast schedule adaptation. In this paper, a mathematical model for this type of problem is proposed and a restarted iterated Pareto greedy (RIPG) metaheuristic is used to find the optimal Pareto front. To demonstrate the appropriateness of this approach, the algorithm is applied to a benchmark specifically designed in this study, considering three objective functions (makespan, total weighted tardiness, and steadiness) and three classes of disruptions (appearance of new jobs, machine faults, and changes in operational times). Experimental studies indicate the proposed approach can effectively solve rescheduling tasks in a multiobjective environment.

1. Introduction

Scheduling in production systems addresses the problem of sequencing the manufacturing of a series of jobs assigned to different machines in a production environment subject to certain requirements. Since 1950s, these complex problems of (NP-hard) type as shown by Garey et al. [1] have been studied in depth by the scientific community.

Flow shop systems seek the optimal scheduling of "n" jobs $\{j_1, j_2, \ldots, j_n\}$ in "m" machines $\{m_1, m_2, \ldots, m_m\}$. Each job consists of "$m$" tasks, the i^{th} task being processed by the machine m_i. In each specific period of time, every machine processes a single task. Thus, to finish a job on the machine i, this machine must be available and the job on machine "$m - 1$" should have been fully processed.

When job scheduling is identical in each machine, there is a permutation flow shop problem (PFSP), which usually assumes no precedence among different job tasks or interruptions on them. PFSP is the problem tackled in this paper as it has many real-world applications, although other environments may appear, such as the blocking flow shop problem [2] or the integrated planning and scheduling production system [3, 4].

Due to the combinatorial nature of the problem, of factorial order, the number of schedules increases to $n!$

Multiobjective optimization seeks to recognize the best advantageous solution by simultaneously analysing multiple discrepant objectives (for example, cost, quality, or time). Every objective can be measured in different units or have different meanings, making them incomparable. Consequently, they do not allow a possible optimal solution for all criteria to be obtained. While finding efficient solutions (known as nondominated), the final objective would be to obtain the group of nondominated solutions, also called Pareto-optimal solutions, making up the Pareto front. Subsequently, the decision maker will select the solution that better meets the business needs.

Multiobjective optimization problems need to use metrics which allow comparing the performance efficiency of the Pareto-Front obtained with every algorithm. Based on the multiobjective optimization literature, the following metrics have been selected:

(i) Hypervolume [5]

(ii) Unary epsilon indicator [6–8]

The methods used to solve multiobjective optimization problems are usually based on scalar [9] or metaheuristic techniques [10]. Regarding scalar methods, many strategies can be used to set up the objective weighting: conventional (CWA), dynamic (DWA), random (RWA), and bang-bang weighted aggregation (BWA). All of them try to convert a multiobjective problem into a single one, by modifying the weights of the objective functions at each process step in order to find the Pareto-front solution set. Other scalar methods are the epsilon-constraint, goal programming, and lexicographical order.

Concerning metaheuristic methods, there are techniques supported by evolutionary algorithms [11–17], particle swarm optimization systems [18], tabu search [19], simulated annealing [20–22], ant colonies [23, 24], greedy algorithms [25], or local search methods [26].

Most of the papers on these methods study biobjective problems which are more easily represented graphically and whose results are easier to analyse. The limitations of current algorithms are usually analysed to solve problems with a greater number of objective functions and subsequently design new techniques which allow the specific problems to be efficiently tackled [27, 28].

This paper is structured in the following way. First, rescheduling schemes studied in the literature are commented. Once the problem's mathematical formulation is defined as an integer linear programming model, the rescheduling architecture is presented to solve the problem based on a predictive-reactive strategy. Next, the RIPG metaheuristic is described and validated on Dubois-Lacoste Instances in a static, biobjective environment. After calibrating the parameters, the RIPG metaheuristic is applied to a benchmark that was specifically designed for this paper. Finally, comparative results are obtained, and the discussion and the main conclusions of the paper are shown.

2. Rescheduling Systems

In classical scheduling problems, an initial set of known jobs must be scheduled considering the available machines in the system and the existing technological constraints. This static (also so-called offline) approach assumes that the jobs to be sequenced and the configuration of the system are known in advance and constant in time. In real production environments, there may be new job arrivals, as well as certain events, not known beforehand, that induce variations requiring a new schedule, such as breakdowns or preventive maintenance activities in machinery or changes in the priority of jobs. These conditions make it necessary to incorporate dynamism in the scheduling systems, through the reorganization of jobs, as the static approach is not optimal anymore. Rescheduling is the process of actualizing the current scheduling to adapt to the disruptions that may appear. The basic phases of this process are the following:

(i) The rescheduling strategy to be used, defining when necessary the need for a new production schedule

(ii) The designated reprogramming policy, specifying the time and manner of carrying out the rescheduling process

(iii) The designed technique to update the original schedule

There are mainly three rescheduling strategies described in the literature: predictive-reactive, proactive (or robust), and dynamic [29–32], among which the predictive-reactive one was selected in this paper. Furthermore, three policies are usually used to complete the rescheduling of the jobs within this approach: event-driven, periodic, and hybrid [33], of which the second one was chosen. There are also three main procedures available to actualize a non-feasible schedule due to the occurrence of disruptions on the production system: right-shift rescheduling [34, 35], partial regeneration [34], and complete regeneration [36, 37], the last of them being used in this paper.

Metrics normally employed in evaluating the reprogramming performance can be based on efficiency (for example, makespan, mean flow time, or total weighted tardiness) or robustness (for example, the system's average stability, as proposed by Pfeiffer et al. [33]).

In the literature, there are hardly any papers which cover rescheduling in dynamic, flow shop environments with permutation and multiple objective functions. Itayef et al. [38] and Liefooghe et al. [39] developed algorithms focused on the search for nondominated solutions in a PFSP environment with a proactive-reactive strategy without using the objective function weighting method. Itayef et al. [38] used the Multiobjective Simulated-Annealing Algorithm (MOSA) while Liefooghe et al. [39] proposed metaheuristics based on the hypervolume, known as the Indicator-Based Evolutionary Algorithm (IBEA). Liu et al. [40] proposed a modified iterated greedy algorithm (eIG_Rep), introducing an effective escape mechanism from a local optimum, to consider the arrival of new orders in a PFSP.

Likewise, the literature does not abound when rescheduling analysis is extended to environments which are different from the PFSP environment. Xiong et al. [41] adapted the NSGA-II, Nondominated Sorting Genetic Algorithm [12], for a stochastic flexible job shop problem (FJSP). Iima [42] developed a multiobjective genetic algorithm to minimize the schedule's total tardiness and stability in an environment with parallel machines and variations in job delivery dates. Zhang et al. [43] proposed the application of a multiobjective evolutionary algorithm based on decomposition (MOEA/D) to solve a hybrid flow shop rescheduling problem with 2 objectives (makespan and stability: as the number of jobs assigned to different machines versus the original schedule) and random machine breakdowns. He et al. [44] implemented the NSGA-III algorithm [27] to solve the rescheduling problem of rush orders in a hybrid flow shop system considering three objectives: makespan, total transportation time, and sequence stability. In reviews on rescheduling techniques by Vieira et al. [45] and Ouelhadj and Petrovic [46], the predictive-reactive strategy is identified as the one most frequently used.

3. Proposed Solution

First, having found no references in the literature, a mathematical formulation for solving this type of problem as an integer linear programming model is proposed. The model tries to optimize three objective functions: makespan, total weighted tardiness (TWT), and stability in order to increase productivity in (1) the production environment (by reducing the makespan), (2) the customer service (by minimizing TWT), and (3) the schedule stability in the rescheduling process when faced with different sudden disruptions.

Subsequently, the architecture for the proposed rescheduling system, its implementation, and the applied RIPG method are described. RIPG is a non-population-based algorithm recently developed by Ciavotta et al. [25] and applied to a biobjective PFSP environment. In this paper, the algorithm was used in a different context, having been adapted to the dynamic rescheduling architecture developed and used to optimize the three objective functions defined: makespan, total weighted tardiness, and stability.

3.1. Mathematical Model. Before presenting the mathematical model formulation, the scientific literature dealing with uncertainty and multiobjective optimization approaches will be briefly discussed. Razmi et al. [47] presented a mathematical model that included uncertainty through fuzzy processing times, that is to say, a stochastic model in which processing times are not input data for the model, but random variables with a known probability distribution. For multiobjective environments, Itayef et al. [38] formulated a biobjective mathematical model to address the arrival of new jobs and thus minimize the maximum completion time for the tasks and improve the schedule stability. In such a model, two independent sets of jobs are considered: new arrivals to the system, and jobs which were already scheduled in the prior rescheduling period.

Fattahi and Fallahi [48] formulated a dynamic FJSP which considered the starting time of the rescheduling period and the rescheduling time (RT), trying to optimize two objectives: makespan and stability. In order to model the FJSP system, restrictions on operations precedence were included (this type of constraint is not present in PFSP problems).

Ramezanian et al. [49] proposed a mathematical model to formulate a biobjective PFSP with the aim of minimizing the linear combination of the stock jobs cost and the late releases cost, including the starting time of the jobs at the machines (release time). The thesis written by Thörnblad [50] on mathematical formulations in FJSP explained the importance of analysing the objective functions when developing the problem-solving method. For this analysis, different formulations were proposed, amongst which the so-called time-indexed model, based on the temporal discretization of the scheduling period, stood out. When different formulations were evaluated taking into consideration the objectives of makespan and the average between the tardiness and the completeness time for the jobs, a great difference in performance was observed, depending upon the objective to be reached.

Yenisey and Yagmahan [51] considered a static, multiobjective PFSP formulation which includes the analysis of three time types: waiting times (for jobs at the machines), job preprocessing times (before jobs are processed by the system-ready times), and processing times. Li et al. [52] proposed a mathematical formulation for the PFSP rescheduling problem that includes recovery times for broken machines with the aim of minimizing a bi-criteria function while weighting the makespan and the system stability (calculated as the number of jobs that have a release time which is different from the planned release time calculated within the prior rescheduling point). Finally, Yuan and Yin [53] developed a fuzzy multiobjective local search-based decomposition (FMOLSD) algorithm to solve a biobjective problem to minimize makespan and total flow time, while considering fuzzy processing times for the jobs scheduled in a permutation flow shop problem.

The mathematical formulation to solve the problem is detailed in the following. This formulation is based on the previously mentioned approaches and incorporates new characteristics. Because this is a PFSP, each job must access each machine following the same processing order, that is to say, from the first to the last machine. In the problem, the following considerations are considered:

(1) Overlapping is not permitted in consecutive operations for the same job; that is, a new operation in a job cannot begin until the preceding operation for that job is completed.

(2) Job pre-emption is not permitted; that is to say, each job must wait until the preceding job has finished in order to be processed on the same machine.

(3) There are infinite buffers (intermediate storage areas) between two machines.

(4) All the jobs that have begun to be processed in the system must maintain their scheduling order when any type of disruption arises.

(5) All jobs interrupted due to a disruption must remain on their assigned machine, with the pending jobs being delayed if necessary. Jobs not impacted by the disruption, so not delayed, will go on with their normal process.

(6) Rescheduling is carried out periodically; therefore, two types of jobs exist in the system: those which are already being processed whose schedule order cannot vary, and those whose order can vary (new jobs and jobs which have not yet begun to be processed). It is important to consider the jobs that are being processed because, depending upon the disruptions

which arise (for example, machine breakdowns), a certain minimum starting time will be established for the next incoming job in the system (in order to be processed by the first machine).

(7) Machine breakdowns can arise at any time and not only when the machines are busy.

(8) If a machine breakdown arises when a job is being processed, the job will restart processing at the point where the interruption occurred.

(9) When a variation in the processing time of a job arises, the new processing time is fixed until the next processing time variation occurs on the same job.

As it is a predictive-reactive rescheduling process, the Mixed-Integer Linear Programming (MILP) model can be run at each rescheduling point, meaning that the past disruptions (new jobs, processing time variations, and machine breakdowns) that have happened during each rescheduling period are received as inputs in the model. Those disruptions impact the next schedule. Therefore, there are no predicted disruptions considered in this formulation.

The problem is defined by three finite sets:

(i) J: the set of new jobs to be scheduled from $1, \ldots, n$

(ii) M: the set of machines from $1, \ldots, m$

(iii) O: the set of operations for each job at each machine from $1, \ldots, m$

Here, J_i is the job at i^{th} position of the permutation.

The parameters used in the model are as follows:

(i) p_{ij}: the processing time for job $J_i \in J$ on machine j.

(ii) d_i: the delivery date for job $J_i \in J$.

(iii) w_i: the priority (weight) of job i with respect to the rest of the jobs.

(iv) rl_i (release time): once a new job has arrived at the system, the release time is the time necessary for job i to be available in order to be processed again by the production environment.

(v) rt_j (ready time): the time in which machine j is ready to process a new job. This input data vector represents the initial situation of the machines before undertaking the new scheduling. It is calculated based on the current time, the disruptions which have arisen, and the scheduling scheme calculated at the previous rescheduling point. It is an important factor as it cannot be supposed that all the machines are initially available.

(vi) RT: current temporal instant. This represents the current point of rescheduling in the production environment.

(vii) To take into account the schedule stability criterion, the number of unprocessed jobs in the base schedule ($n_{\text{not processed}}$) needs to be defined, including new arrivals to the system, the starting time for the jobs at each machine in the base schedule ($S_{i,j}^{\text{baseline}}$), the current rescheduling instant (RT), and a scale constant (scale = 10 in accordance with the value used in Pfeiffer et al. [33], used to apply a quadratic penalty method to the starting time variation in the base schedule with respect to RT).

(viii) B_{start}^{J}: a breakdown instant for machine j.

(ix) B_{end}^{J}: recovery instant from the breakdown of machine j.

(x) Big$M$: large constant (Big$M \longrightarrow \infty$).

In terms of decision variables, the following were considered:

(i) Binary variables which represent the position of each of the jobs in the planned schedule:

$$\forall i,\ j \in J\colon\ x_{ij} = f(x) = \begin{cases} 1, & \text{if } J_i \text{ is at } j \text{ position,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

(ii) C_i: completion time for the job at position i.

(iii) $C_{i,j}$: completion time for the job at position i, on machine j.

(iv) $S_{i,j}$: starting time of job i, on machine j.

(v) Binary variables that indicate the circumstances under which a machine breakdown has occurred (three situations are modelled):

(a) $\forall i \in J \wedge j \in M$: $y_{i,j,1}$: it takes the value 1 if the operation $O_{i,j}$ finishes before the breakdown occurs; that is to say, the time the operation takes to finish is less than the machine's breakdown starting point.

(b) $\forall i \in J \wedge j \in M$: $y_{i,j,2}$: it takes the value 1 if the operation $O_{i,j}$ is not yet finished at the moment when the breakdown occurs; that is to say, the time the operation takes to finish overlaps with the machine's breakdown starting point.

(c) $\forall i \in J \wedge j \in M$: $y_{i,j,3}$: it takes the value 1 if the operation $O_{i,j}$ has a starting time which is after the moment when the machine breaks down.

(vi) $C_{\max}$, TWT, STB: these variables correspond to the maximum completion time for the tasks, the total weighted tardiness, and the schedule stability, respectively.

The mathematical model for the MILP formulation of the permutation flow shop rescheduling problem was based on the following equations.

The objectives of the problem were to minimize the makespan, the total weighted tardiness (TWT), and the stability of the schedule to be planned (STB), obtaining the optimal Pareto frontier (PF):

$$\min \text{PF}\,(C_{\max}, \text{TWT}, \text{STB}). \tag{2}$$

The makespan was calculated as the maximum completion time for all the jobs, and it corresponds to the completion time for the last scheduled job, that is, the time when the last task in the last machine finishes:

$$C_{\max} = \max_{i=1,\ldots,n} \left(C_i(\pi)\right) = C_n. \tag{3}$$

The total completion time for each job, C_i, is defined by the following equation, according to the completion time in the last machine M:

$$\forall i \in J: \quad C_i = C_{i,j=M}, \tag{4}$$

where M is the set of machines in the system.

More specifically, the completion time for each job i on each machine j takes into consideration the following components:

(i) The time at which the job i can begin to be processed by machine j. This time depends on the following:
  (a) The completion time of job i on the previous machine $(C_{i,j-1})$.
  (b) The completion time of the previous job, located at position $(i-1)$, on the current machine j $(C_{i-1,j})$.
  (c) The maximum of these terms will be the instant of time in which job i can begin to be processed by machine j $(S_{i,j})$.

(ii) The processing time for job i on machine j $(p_{i,j})$.

(iii) Time of breakdown on machine j when job i is being processed $(B^J_{\text{end}} - B^J_{\text{start}})$.

When calculating the completion time along with machine breakdowns, the following equations are modelled, considering the three possible breakdown situations:

If the operation finishes before the breakdown appears,

$$\begin{aligned}
&\forall i \in J \wedge \forall j \in M\colon C_{i,j} + (1 - y_{i,j,1}) \cdot \text{Big}M \geq S_{i,j} + \sum_{k=1}^{n} x_{ki} \cdot p_{kj},\\
&\forall i \in J \wedge \forall j \in M\colon C_{i,j} - (1 - y_{i,j,1}) \cdot \text{Big}M \leq S_{i,j} + \sum_{k=1}^{n} x_{ki} \cdot p_{kj}.
\end{aligned} \tag{5}$$

If the breakdown appears while the operation $O_{i,j}$ is being processed, it is necessary for the job to wait until the machine is ready again in order to continue its processing:

$$\begin{aligned}
&\forall i \in J \wedge \forall j \in M\colon\\
&\quad C_{i,j} + (1 - y_{i,j,2}) \cdot \text{Big}M \geq S_{i,j} + \sum_{k=1}^{n} x_{ki} \cdot p_{kj} + (B^J_{\text{end}} - B^J_{\text{start}}),\\
&\forall i \in J \wedge \forall j \in M\colon\\
&\quad C_{i,j} - (1 - y_{i,j,2}) \cdot \text{Big}M \leq S_{i,j} + \sum_{k=1}^{n} x_{ki} \cdot p_{kj} + (B^J_{\text{end}} - B^J_{\text{start}}).
\end{aligned} \tag{6}$$

Finally, if the operation $O_{i,j}$ has not yet begun, the work must wait until the machine breakdown has finished:

$$\begin{aligned}
&\forall i \in J \wedge \forall j \in M\colon\\
&C_{i,j} + (1 - y_{i,j,3}) \cdot \text{Big}M \geq \text{Max}(S_{i,j}, B^J_{\text{end}})) + \sum_{k=1}^{n} x_{ki} \cdot p_{kj},\\
&\forall i \in J \wedge \forall j \in M\colon\\
&C_{i,j} - (1 - y_{i,j,3}) \cdot \text{Big}M \leq \text{Max}(S_{i,j}, B^J_{\text{end}}) + \sum_{k=1}^{n} x_{ki} \cdot p_{kj}.
\end{aligned} \tag{7}$$

Although the above equations contain the max function (that is not linear), they can be expressed in a linearized way using a big-M plus binary variables representation, being automatically transformed by most of the commercial solvers and respecting the MILP formulation of the problem.

If a machine breakdown appears, only one of these three situations can arise; this can be ensured by including the following equation:

$$\forall i \in J \wedge \forall j \in M\colon \sum_{k=1}^{3} y_{i,j,k} = 1. \tag{8}$$

The starting time for each job i on each machine j depends upon the following equation:

$$\forall i \in J \wedge j \in M\colon \quad S_{i,j} = \text{Max}(C_{i,j-1}, C_{i-1,j}). \tag{9}$$

In order to define the starting times in the MILP model, the following equations are used.

Operation $O_{i,j}$ cannot begin until the operation for the previous job has finished on the same machine:

$$\forall i > 1 \in J \wedge \forall j \in M\colon \quad S_{i,j} \geq C_{i-1,j}, \tag{10}$$

and operation $O_{i,j}$ cannot begin until the previous operation for the same job has finished:

$$\forall i \in J \wedge \forall j > 1 \in M\colon\ S_{i,j} \geq C_{i,j-1}. \tag{11}$$

The starting time for job i on the first machine must be greater than the overall release time, as this is the moment when the job is ready to begin to be processed:

$$\forall i \in J\colon\ S_{i,1} \geq rl_i. \tag{12}$$

Because not all the machines are initially available, in order to be able to process the next job, it is important to take into consideration the fact that the starting time for the jobs cannot begin until the machine is available; that is to say,

$$\forall i \in J \wedge \forall j \in M\colon\ S_{i,j} \geq rt_j. \tag{13}$$

In case there are delays—that is, when a job completion time is greater than its delivery date—the total weighted tardiness (TWT) is calculated as the weighted sum of differences between the total completion time for each job and its delivery date, according to their assigned priority weights (w_i):

$$\text{TWT} = \sum_{i=1}^{n} \text{Max}(C_{im} - d_i, 0) \cdot w_i. \tag{14}$$

For modelling purposes in the MILP problem, the following equations were included:

$$\begin{aligned}
&\text{TWT} \geq \sum_{i=1}^{n} (C_{im} - d_i) \cdot w_i,\\
&\text{TWT} \geq 0.
\end{aligned} \tag{15}$$

The third objective function considered in the problem is the stability (STB), with respect to the base schedule calculated in the previous period (baseline). Stability is expressed as the average of the absolute differences of unprocessed job starting times, adding a penalty which depends upon the variation in the starting time for the base schedule regarding the current RT. Unprocessed tasks are defined as those jobs which have previously been scheduled at the prior point of rescheduling and which have not yet begun to be processed by the system at the current RT:

$$\mathrm{STB} = \frac{1}{n_{\text{not processed}}} \cdot \sum_{i=1}^{n}\left(\left|S_{i,1} - S_{i,1}^{\text{baseline}}\right| + \frac{\text{scale}}{\sqrt{S_{i,1}^{\text{baseline}} - \mathrm{RT}}}\right). \tag{16}$$

The absolute value function used in calculating the stability objective is nondifferentiable in addition to being nonlinear. Nevertheless, it can be expressed in an integer linear way, as follows:

$$\mathrm{STB} = \frac{1}{n_{\text{not processed}}} \cdot \sum_{i=1}^{n}\left(\text{DeltaStart Time}_i + \frac{\text{scale}}{\sqrt{S_{i,1}^{\text{baseline}} - \mathrm{RT}}}\right). \tag{17}$$

Two new binary variables d_{1j} and d_{2j} were introduced, as well as a large constant Big$M$, and the following equations for MILP formulation:

$$\begin{aligned}
&\forall j \in J:\ 0 \le S_{j,1} \le \mathrm{Big}M,\\
&\forall j \in J:\ 0 \le \text{DeltaStart Time}_j - \left(S_{j,1} - S_{j,1}^{\text{baseline}}\right) \le 2 \cdot \mathrm{Big}M \cdot d_{2j},\\
&\forall j \in J:\ 0 \le \text{DeltaStart Time}_j - \left(S_{j,1}^{\text{baseline}} - S_{j,1}\right) \le 2 \cdot \mathrm{Big}M \cdot d_{1j},\\
&\forall j \in J:\ d_{1j} + d_{2j} = 1.
\end{aligned} \tag{18}$$

If $d_1 = 1$, then delta at the starting time would take the value $S_{j,1} - S_{j,1}^{\text{baseline}}$; otherwise, if $d_2 = 1$, it would take the value $S_{j,1}^{\text{baseline}} - S_{j,1}$.

Finally, the model ensures that a job must be in one single position and that only a single job can exist at any position of the schedule. This is done through the following equations:

$$\begin{aligned}
\forall i \in J:&\quad \sum_{j=1}^{n} x_{ij} = 1,\\
\forall j \in J:&\quad \sum_{i=1}^{n} x_{ij} = 1.
\end{aligned} \tag{19}$$

3.2. Rescheduling Architecture. The rescheduling strategy used was based on the predictive-reactive approach; thus, a baseline schedule was first generated and then updated according to the disruptions that arise in system. A periodic rescheduling policy was used through time windows. Finally, the rescheduling method consisted of a complete rescheduling based on heuristic/metaheuristic algorithms (as rescheduling method we proposed RIPG algorithm, described in detail in the next section). RIPG is selected because its efficient behaviour in static multiobjective permutation flow shop problems [25] makes it a promising technique for the dynamic environment under study. The proposed rescheduling architecture is shown in Figure 1.

Initially, the baseline schedule was generated according to both the flow shop system involved and the jobs to be scheduled. Then, a synchronous control was established—every T units of time—in order to perform a rescheduling process when disruptions arise in the system, obtaining the Pareto-front output from the solutions found in each executed rescheduling interval. It is important to specify an appropriate period, based on the dimensions of the problem, for each rescheduling process. In this paper, this period was calculated as the maximum completion time for jobs ($C_{\max}$) divided by the total number of rescheduling points to be executed on the system (we have considered 5 rescheduling points as it is an appropriate number to evaluate the architecture adaptation capacity and does not excessively increase the number of executions in the system).

Likewise, at each point of rescheduling, it is necessary to select a representative solution for the calculated Pareto front. The selected solution will be used as a baseline schedule to be employed for each new rescheduling. The fundamental importance of this base schedule lies in its use in evaluating the stability objective function. To calculate the representative solution for the Pareto front, a knee point method was used [54–56]. A detailed description of the knee point method used to calculate this most representative solution can be found in Valledor et al. [57].

3.3. Restarted Iterated Greedy Algorithm. The implementation of the developed RIPG algorithm was based on the design proposed by Ciavotta et al. [25]. Their application of the technique was explained using a biobjective PFSP, and its results on Taillard Instances were described, comparing the RIPG technique with 17 algorithms, including MOSA and Multiobjective Tabu Search (MOTS). Ciavotta et al. [25] described the RIPG as a state-of-the-art method because of its excellent results when compared with the best approaches previously presented in the literature. The RIPG algorithm is an extension of the IG (Iterated Greedy) technique proposed by Ruiz and Stützle [58]. These techniques belong to the stochastic local search (SLS) category. The RIPG technique consists of 5 phases, as shown in Figure 2:

Phase 1. First, an Initial Solution Set (ISS) is determined based upon the execution of different simple heuristics. More specifically, NEH heuristics [59, 60] are used, each one being applied to obtain acceptable solutions with different objective functions. For each of the two solutions obtained via heuristics, the greedy phase (described below in phase three) is applied, generating an initial set of nondominated solutions (working set).

Phase 2. Second, the optimization loop begins with the process of selecting a solution via the Modified Crowding Distance Assignment (MCDA) method, in order to improve the working set during the greedy phase. The MCDA process receives the working set as an input parameter and calculates the distance to each one of the solutions. The selected solution is the one

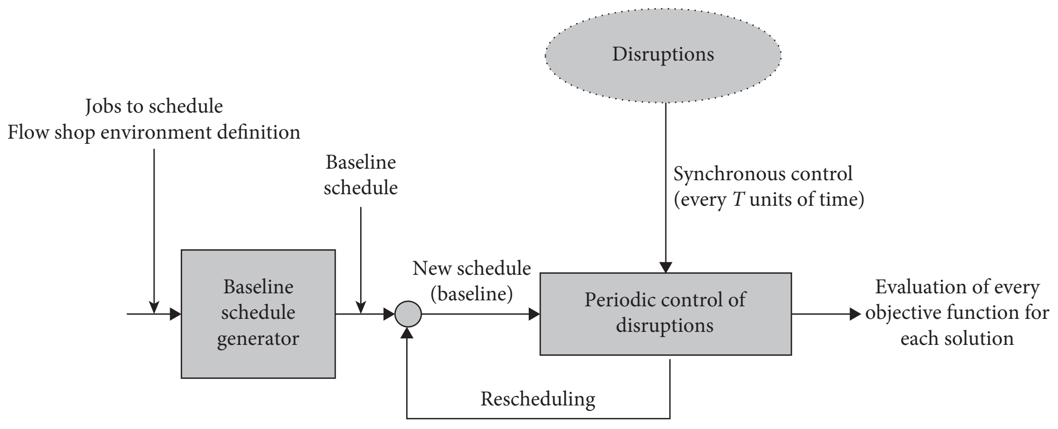
Disruptions
Jobs to schedule
Flow shop environment definition
Baseline schedule
Synchronous control (every T units of time)
Baseline schedule generator
New schedule (baseline)
Periodic control of disruptions
Evaluation of every objective function for each solution
Rescheduling

Figure 1: Rescheduling architecture.

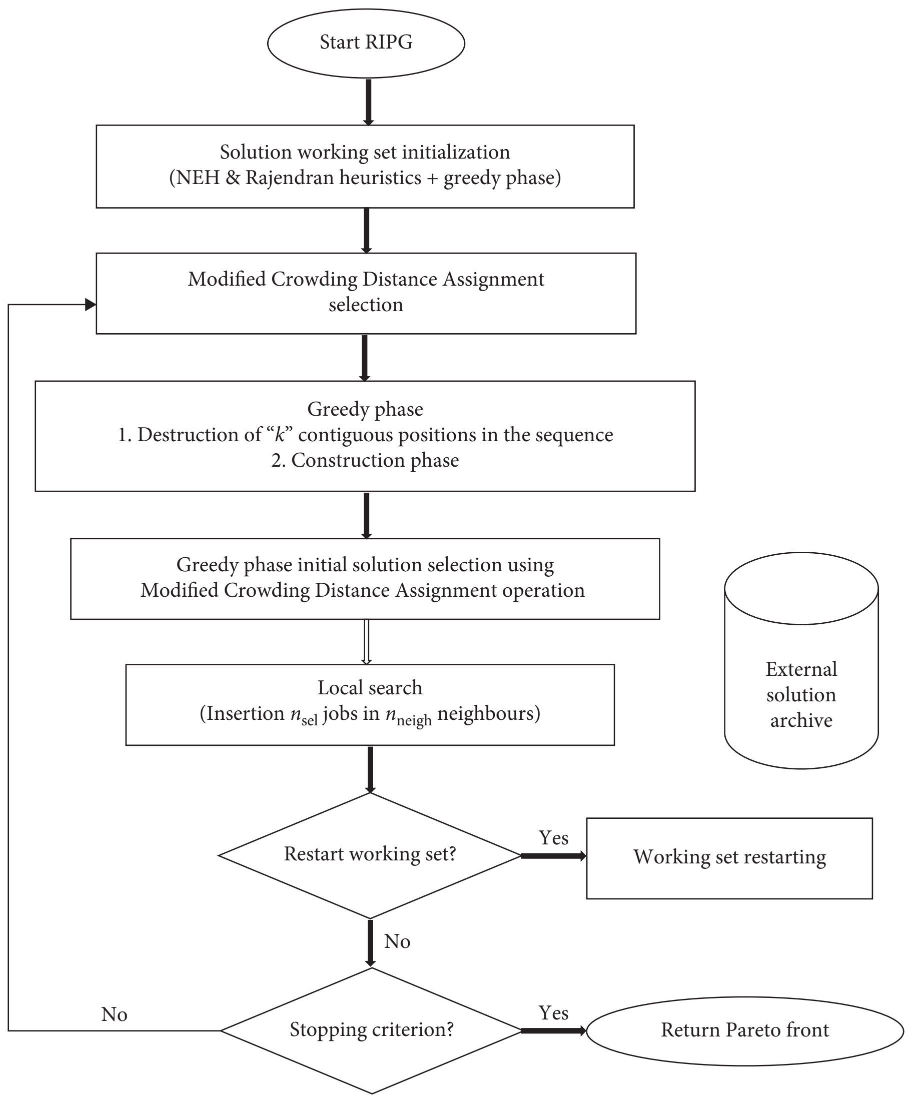
Start RIPG
Solution working set initialization (NEH & Rajendran heuristics + greedy phase)
Modified Crowding Distance Assignment selection
Greedy phase
1. Destruction of "k" contiguous positions in the sequence
2. Construction phase
Greedy phase initial solution selection using Modified Crowding Distance Assignment operation
External solution archive
Local search (Insertion n_{sel} jobs in n_{neigh} neighbours)
Restart working set?
Yes
Working set restarting
No
No
Stopping criterion?
Yes
Return Pareto front

Figure 2: Flow diagram for the RIPG technique.

with the greatest modified crowding distance (MCD) value, which coincides with solutions in sparsely populated search spaces. The main difference between MCDA and Crowding Distance Assignment (CDA), proposed originally in the NSGA-II [12], is that the first method considers the number of times that a solution has been previously selected, giving it less opportunity of being reselected.

Phase 3. As the next step, the greedy phase is executed. The selected solution is modified by applying both a destruction operation to eliminate a sub-schedule of jobs from the solution and a greedy process, known as a construction operation, to insert each of the eliminated jobs in one of the possible schedule positions. In this way, a solution pool is created.

Phase 4. Next, once again by the MCDA method, a solution is selected from the solution pool generated during the greedy phase, and a local search method is applied for improvement. As local search, we applied the same classical approach as proposed by Ciavotta et al. [25], where n_{sel} jobs are randomly selected; then, each of them is eliminated from the schedule and reinserted in n_{neigh} consecutive positions, half of which precede the initial position of the selected job and the other half follow it. Once all the movements are undertaken for the selected jobs, the generated solutions are subsequently evaluated, and the dominated solutions are eliminated. The nondominated solutions are added to the working set.

Phase 5. Finally, a restarting mechanism is implemented on the set of solutions to prevent the algorithm from reaching only a local optimal solution and causing a stagnation effect. The mechanism is simple and reliable; it consists of storing the set of nondominated solutions in the working set in an external file and, subsequently, randomly restarting the working set. This restart process is launched if the size of the working set, that is, the number of nondominated solutions found until that time, does not vary in a pre-set number of iterations.

The RIPG algorithm requires the configuration of 5 input parameters: the size of the set of solutions (working set), k (the number of consecutive elements eliminated during the destruction operation of the greedy phase), n_{sel} (the number of elements randomly selected in the local search phase), n_{neigh} (the number of consecutive positions in which a job is reinserted in the local search phase), and the restart threshold (the maximum number of iterations allowed without modifying the size of nondominated solution set; once this limit is surpassed, the working set automatically restarts).

4. Results and Discussion

4.1. Experimental Design. The lack of a common scheme to assess rescheduling problems in a PFSP leads to the generation of databases that permit to work with diverse disruptions in the production environment (appearance of new jobs, machine failures, etc.) and evaluate a multiobjective optimization problem. The main database to assess PFSPs was established by Taillard [61] whose instances were initially generated to be applied into single-objective environments, minimization of the completion time of the jobs. Subsequently, these instances were modified to incorporate a due date to each job, similarly to Minella et al. [62] and Hasija and Rajendran [63], allowing the rescheduling system to be assessed under different objective functions, such as total weighted tardiness (TWT) or maximum delay of jobs [64].

In the proposed rescheduling system, after the original schedule comes into effect, different disruptions may occur over time. NEH heuristic was used [59, 65] to achieve the original schedule in order to optimize the makespan. Minella et al. [62] also used it to have a feasible original schedule. Additionally, in our generated benchmark, the disruptions to be considered and the way in which they occur are defined. Although most rescheduling systems focus on one type of disruption, in this paper we use three: machine faults, new job arrivals, and variable processing times.

Machine faults were defined through an exponential distribution defined by the level of failure reached by the system, the mean time to repair (MTTR), and the mean time between faults (MTBF). The breakdown level (Ag), percentage of machine failure time, was estimated by means of the following expression:

$$\text{Ag} = \frac{\text{MTTR}}{\text{MTTR} + \text{MTBF}}. \tag{20}$$

We have followed the machine faults parameterization proposed by Adibi et al. [66] and Mason et al. [67]. Regarding the arrivals of new jobs, an exponential distribution was also adopted with an arrival rate based on the quotient between the degree of use of the system and the average operating time [68]. Concerning the deviation of the job processing times, they were randomly generated (1% probability) with a variation factor of 30%, in line with the parameters established by Swaminathan et al. [69] with a symmetric triangular distribution.

From the 120 instances of Taillard, another 120 rescheduling instances were created, and from them a subset of 50 was selected to be evaluated with the RIPG algorithm. The reason for not evaluating all 120 instances is the high execution time required. Thus, for each of these instances, 5 rescheduling points were executed, during which the system was updated according to the events in the flow shop environment, and, therefore, the technique was executed 5 times per instance. The 50 instances used were randomly selected from the set of 120 Taillard Instances, excluding those of greater size (with 500 jobs), because it is not usual to work with such a high number of jobs in environments with more than one objective function. At the website Instances & Solutions, the following files have been included:

(i) Training instances used with the Irace Package
(ii) Evaluation instances
(iii) Pareto-front solutions from the RIPG for each one of the 10 repetitions undertaken for each evaluation instance

4.2. Validation of the RIPG Algorithm on Dubois-Lacoste Instances in a Static, Biobjective Environment. Since we have

Table 1: Classical parametrization for the RIPG algorithm.

Parameter	Default value
Size of the set of job solutions after the restart phase (working set)	100
k (number of consecutive elements eliminated in the greedy phase's destruction operation).	5
n_{sel} (number of elements selected randomly in the local search phase). Dynamic parameter dependent on the number of jobs in the instance to be solved.	$n_{\text{sel}} = n_{\text{count}}$ if $n_{\text{count}} \leq n * 0.5$, else $n_{\text{sel}} = n * 0.5$ n is the number of jobs and n_{count} is the number of times that a solution has been selected
n_{neigh} (consecutive positions where a job is reinserted in the local search phase).	5
Restart threshold (maximum number of iterations allowed without modifying the size of the nondominated solutions set. Once this value is surpassed, the working set randomly restarts).	$2 * n$ (where n is the number of jobs)

not found a benchmark for the dynamic three-objective problem in the literature, we adapted the algorithm to a static, biobjective approach and used the multiobjective performance metrics proposed by Dubois-Lacoste et al. [26] as a way to validate the correct behaviour of the algorithm. The aim of Dubois-Lacoste et al. was minimizing both the maximum completion time for the tasks and the total weighted tardiness. The reason for selecting this environment is the ability to compare the results of the applied algorithm with the best results obtained in the literature through a Two-Phase Local Search + Pareto Local Search (TP + PLS) hybrid algorithm.

In our experiments, the RIPG metaheuristic used classic parametrization as proposed in the literature [25] and recommended for biobjective environments (see Table 1).

Each experiment was repeated 5 times (independently) in order to evaluate the multiobjective metrics of hypervolume and the unary epsilon indicator (I_{ε}^{1}) on each of the Dubois-Lacoste et al. instances [26]. Based upon a preliminary statistical study with the Wilcoxon test, the RIPG algorithm was inferred to behave in a similar way to the TP + PLS algorithm in a static, biobjective environment, as can be seen in the box plot in Figure 3. Similar results are obtained with the unary multiplicative epsilon indicator, as can be seen in Figure 4.

In Figure 5, a comparison between the Pareto fronts obtained by the RIPG algorithm and the best results found in the literature is shown [26] for some of the Taillard Instances.

The RIPG algorithm approaches the best Pareto fronts obtained in the literature, following the same trends without excessively deviating from the reference nondominated solutions. New nondominated solutions have even been detected in some cases, as can be seen in instances ta018 and ta024.

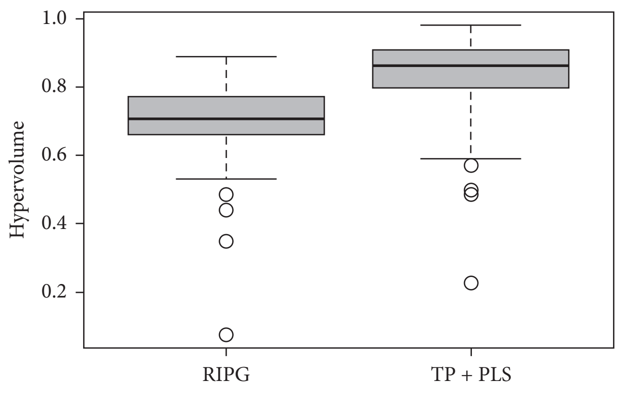

Figure 3: Box plot showing hypervolume in a biobjective environment.

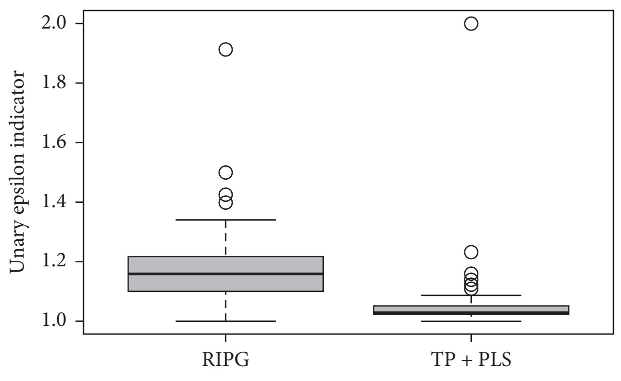

Figure 4: Box plot showing the unary epsilon indicator in a biobjective environment.

4.3. Parameter Calibration. As classical parameters in static environments could not adjust properly to dynamic scenarios, we have run a parameter calibration of RIPG in our rescheduling multiobjective environment. The hardware environment for experimentation consisted of a machine with a Quad-Core Intel Xeon X5675 processor, 3.07 GHz, and 16 GB of RAM memory. The operating system used was Windows 7 Enterprise Edition, 64 bit architecture. The software was developed on the .NET platform with the C# programming language using the Microsoft Visual Studio 2010 Development Framework.

In order to define the stopping criterion, we analysed different approaches from the literature [58, 70–74]. Most use a maximum execution time, calculated depending on the size of the problem that is defined by the number of jobs and machines that the system has available, as the stopping criterion. For this study, the time limit was calculated with the following equation: $t = n \times m^2 \times 100$, where t is the time in milliseconds, n is the number of jobs, and m is the number of machines. The above expression was used to calculate the

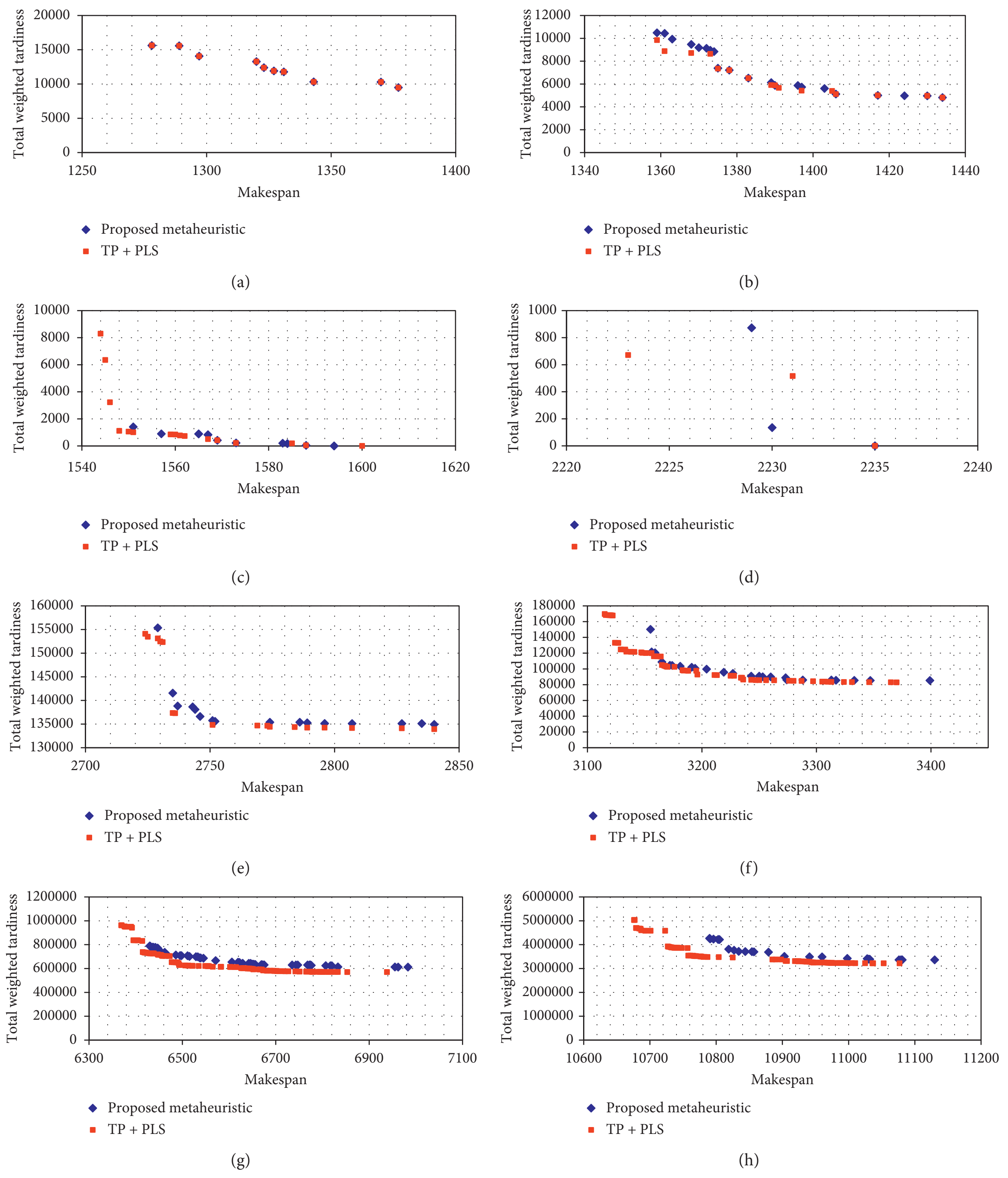

Figure 5: Comparison of Pareto frontiers between the RIPG metaheuristic and the TP + PLS algorithm in a static, biobjective flow shop environment. (a) Problem instance ta001, (b) problem instance ta002, (c) problem instance ta018, (d) problem instance ta024, (e) problem instance ta031, (f) problem instance ta047, (g) problem instance ta084, and (h) problem instance ta100.

execution time because both the number of jobs and the number of machines played a role in the difficulty of the problem to be solved. In addition, it allowed the use of more calculation time in problems with greater complexity.

Likewise, the automatic configuration of the parameters defined by the RIPG algorithm was proposed based on an extension of the iterated F-Race method and implemented in the Irace tool [75]. To accomplish this, hypervolume was defined as a metric to evaluate the configuration schemes, as this is the most common criterion in multiobjective problems. To select the training instances, a complexity analysis was used depending on the results provided in the literature regarding Taillard Instances [26].

Table 2: Automatic RIPG configuration via Irace.

Parameter	Value	Data type	Range established for Irace
Size of the set of job solutions after the restart phase (working set).	124	Integer	[50, 200]
k (number of consecutive elements eliminated in the greedy phase's destruction operation).	5	Integer	[1, 10]
n_{sel} (number of elements selected randomly in the local search phase). Dynamic parameter dependent on the number of jobs in the instance to be solved.	$n_{\text{sel}} = n_{\text{count}}$ if $n_{\text{count}} \leq n * 0.1$ else $n_{\text{sel}} = n * 0.1$ n is the number of jobs and n_{count} is the number of times that a solution has been selected	Decimal	[0.1, 1]
n_{neigh} (consecutive positions where a job is reinserted in the local search phase).	8	Integer	[1, 12]
Restart threshold (maximum number of iterations allowed without modifying the size of the nondominated solutions set. Once this value is surpassed, the working set randomly restarts).	$3 * n$ (where n is the number of jobs)	Integer	[1, 4]

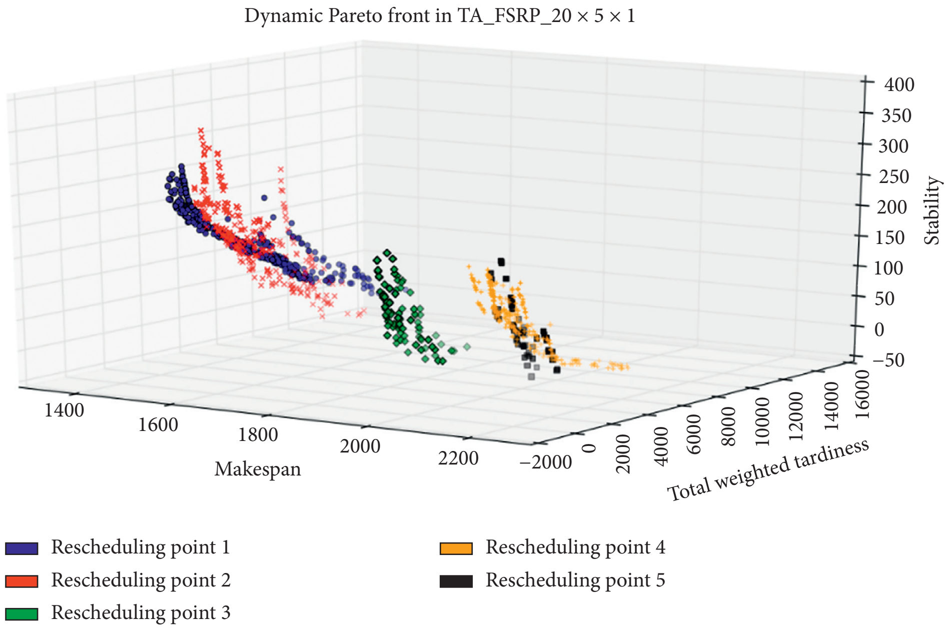

Figure 6: RIPG Pareto front in TA_20_5_1.

As representative instances, a total of 6 problems were selected (ta02, ta031, ta047, ta061, ta080, and ta024), in which two have low complexity, two have medium complexity, and two are of high complexity in terms of solving difficulty. The criterion chosen to evaluate the complexity of an instance was based on a trade-off between the number of nondominated solutions found in the literature for biobjective, static problems (makespan and total weighted tardiness), combined with the relative error found in mono-objective problems considering the makespan criterion. The instances used in the calibration phase were not subsequently used for the evaluation of the algorithm in order to avoid overfitting.

After analysing the Irace results, the RIPG algorithm parameters' configurations did not have a great impact on the results; therefore, this algorithm is robust in terms of parameter variation. In Table 2, the results of the optimal Irace automatic configuration for the RIPG algorithm are shown.

4.4. Performance of the RIPG Algorithm in Multiobjective Dynamic Environments. With the aim of verifying the performance of the RIPG metaheuristic, it was executed on the 50 benchmark problems which were developed specifically in this study for a flow shop rescheduling environment. The RIPG algorithm was executed 10 times (independently) for each of the experiments in order to carry out a reliable statistical analysis. As it was impossible to compare the RIPG algorithm with other metaheuristics in the proposed environment, the results obtained from the simulations in 4 selected instances are presented below. In Figures 6–9, the

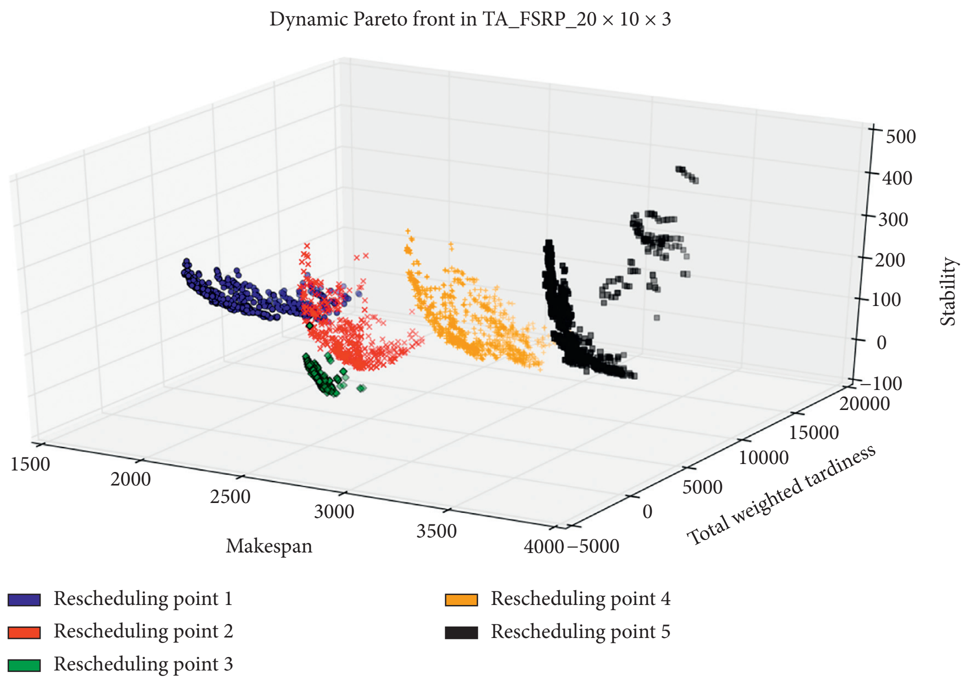

Figure 7: RIPG Pareto front in TA_20_10_3.

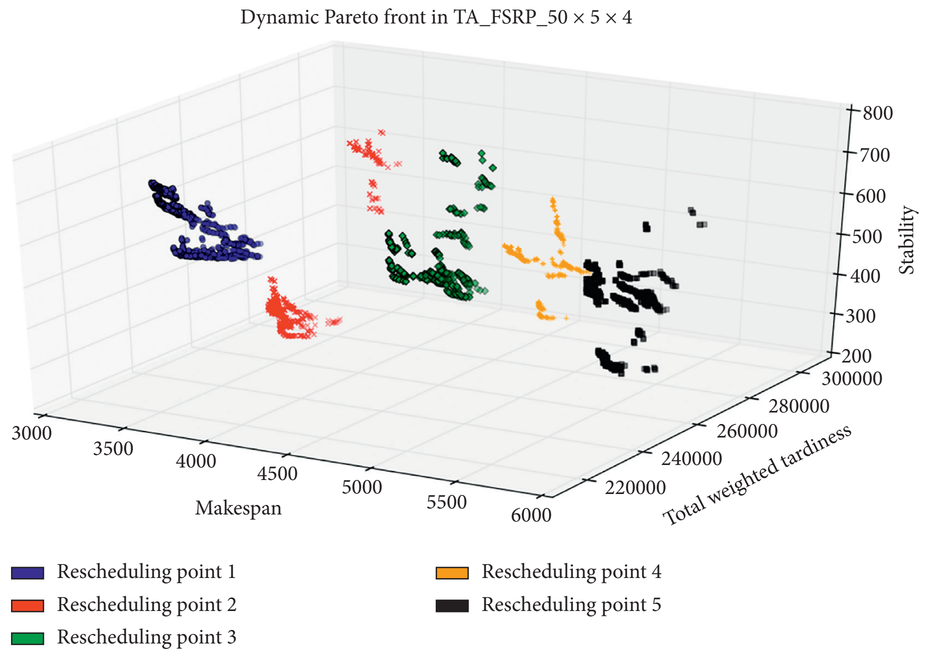

Figure 8: RIPG Pareto front in TA_50_5_4.

Pareto fronts obtained in the 4 chosen instances are shown. Five Pareto fronts are shown from each instance, corresponding to the five generated at each point of rescheduling. We can see how the number of nondominated solutions found goes down as the problem's size increases.

All the Pareto fronts obtained from the RIPG algorithm can be viewed at Instances & Solutions. https://www.dropbox.com/s/l4mqu5p17wov5s6/Multiobjective Rescheduling-Instances%26Solutions.zip?dl=0.

Finally, with the aim of viewing the rescheduling processes when disruptions appear, a small instance was selected (TA_20_5_3) to show the evolution of the manufacturing schedule. It should be noted that, as this is a small instance, machine breakdowns did not occur;

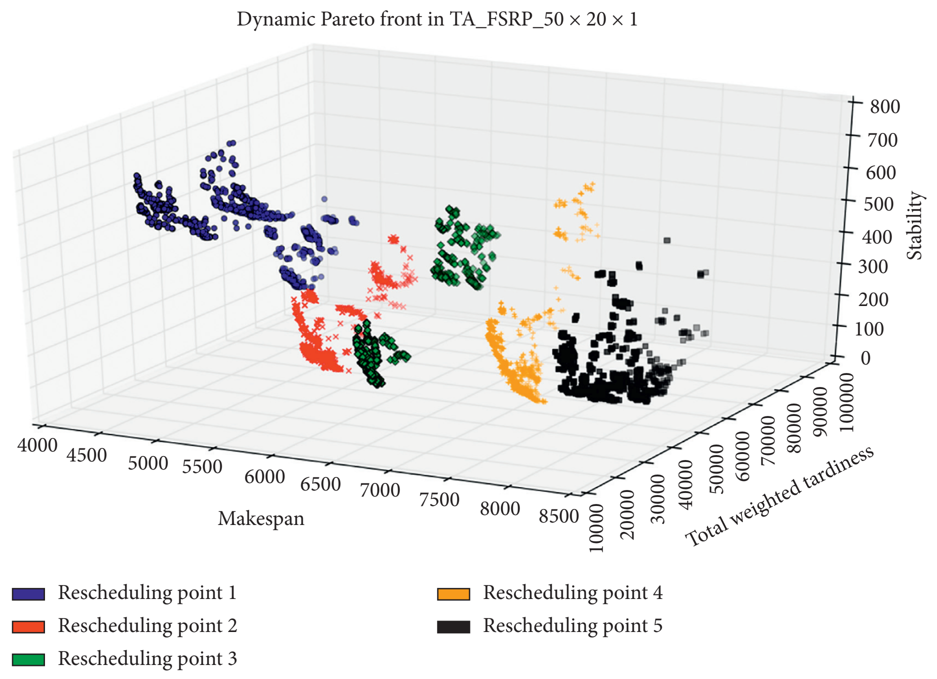

FIGURE 9: RIPG Pareto front in TA_50_20_1.

TABLE 3: Disruptions in TA_20_5_3.

Time	Disruptions
$t_1 = 232$	7 new jobs, 2 variations in processing time
$t_2 = 464$	3 new jobs, 1 variation in processing time (job 9)
$t_3 = 696$	4 new jobs, 4 variations in processing time
$t_4 = 928$	5 new jobs, 3 variations in processing time
$t_5 = 1160$	4 new jobs, 2 variations in processing time

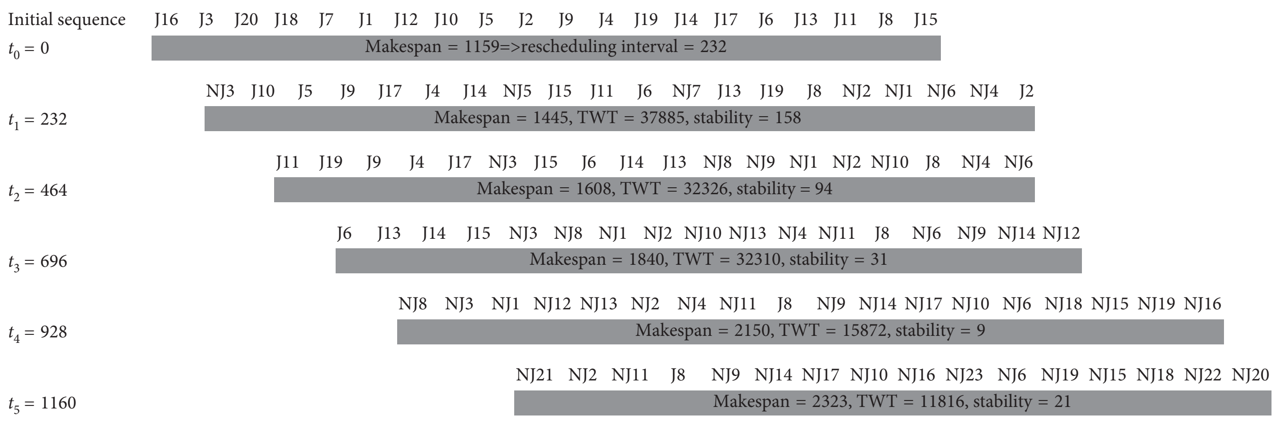
Initial sequence
J16 J3 J20 J18 J7 J1 J12 J10 J5 J2 J9 J4 J19 J14 J17 J6 J13 J11 J8 J15
t0 = 0
Makespan = 1159=>rescheduling interval = 232
NJ3 J10 J5 J9 J17 J4 J14 NJ5 J15 J11 J6 NJ7 J13 J19 J8 NJ2 NJ1 NJ6 NJ4 J2
t1 = 232
Makespan = 1445, TWT = 37885, stability = 158
J11 J19 J9 J4 J17 NJ3 J15 J6 J14 J13 NJ8 NJ9 NJ1 NJ2 NJ10 J8 NJ4 NJ6
t2 = 464
Makespan = 1608, TWT = 32326, stability = 94
J6 J13 J14 J15 NJ3 NJ8 NJ1 NJ2 NJ10 NJ13 NJ4 NJ11 J8 NJ6 NJ9 NJ14 NJ12
t3 = 696
Makespan = 1840, TWT = 32310, stability = 31
NJ8 NJ3 NJ1 NJ12 NJ13 NJ2 NJ4 NJ11 J8 NJ9 NJ14 NJ17 NJ10 NJ6 NJ18 NJ15 NJ19 NJ16
t4 = 928
Makespan = 2150, TWT = 15872, stability = 9
NJ21 NJ2 NJ11 J8 NJ9 NJ14 NJ17 NJ10 NJ16 NJ23 NJ6 NJ19 NJ15 NJ18 NJ22 NJ20
t5 = 1160
Makespan = 2323, TWT = 11816, stability = 21

FIGURE 10: Schedule of jobs at the different rescheduling points in TA_20_5_3.

however, the rest of the disruptions did appear (see Table 3). In Figure 10, the schedule of the jobs at the different points of rescheduling is shown. J16 is the first job in the initial base schedule. NJX identifies new jobs. The base schedule is presented ($t = 0$) as well as 5 rescheduling runs (at times $t = 232$, $t = 464$, $t = 696$, $t = 928$, and $t = 1160$). Due to the appearance of new jobs and variations in job processing times, the schedule was modified. Figure 11 shows the RIPG Pareto front for each rescheduling point in the instance TA_20_5_3.

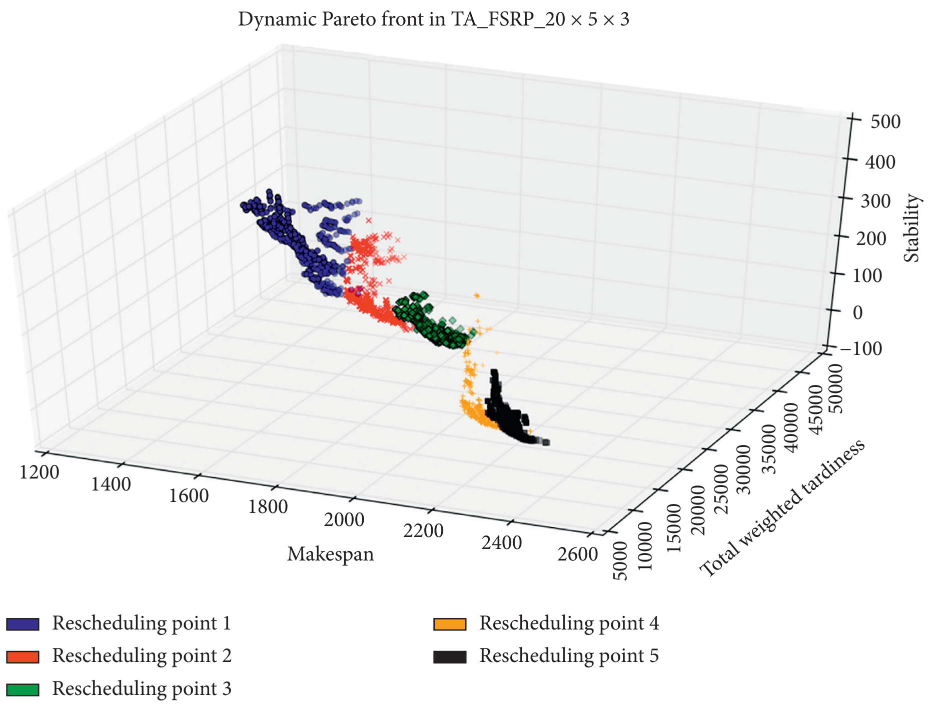

Figure 11: RIPG Pareto front in TA_20_5_3.

5. Conclusions and Future Lines of Research

This research proposes several relevant contributions. First, a new MILP mathematical formulation has been proposed to represent the dynamic, triobjective rescheduling problem to be solved. Likewise, a set of problems (benchmark instances) has been defined to model dynamic rescheduling environments, including the specification of different disruptions which can arise in a production environment. Additionally, a rescheduling architecture has been designed and implemented, based on a predictive-reactive strategy with periodic rescheduling. Moreover, the RIPG metaheuristic has been applied and its proper operation in dynamic rescheduling systems verified.

To validate the proposed approach and having found no references in the literature, a benchmark based on a static biobjective environment was used. It is important to note that RIPG algorithm approaches the best Pareto frontiers obtained in the literature, following the same trends without excessively deviating from the reference nondominated solutions set in a static biobjective environment. New nondominated solutions have even been detected in some cases, as can be seen in instances ta018 and ta024. Likewise, RIPG is very robust because any parametrization of the algorithm yields similar results in terms of the median value. As it has been impossible to compare the RIPG algorithm with other metaheuristics in the proposed environment, due to the lack of techniques applied in such problems, Pareto frontiers obtained for several instances of problems of different sizes have been shown. The number of nondominated solutions found goes down as the problem's size increases.

For future lines of research, it may be useful to delve deeper into analysing the influence of the base schedule (baseline) used to evaluate the stability objective function on the algorithm's results. In addition, although this paper has analysed all the experiments using 5 rescheduling points, the dynamic system could be evaluated for a variable number of rescheduling points between 0 and 1000, as Sabuncuoglu and Karabuk [76] considered in their work. Likewise, the RIPG metaheuristic could be adjusted to dynamic situations in problems known as *many-objective optimization*, in which the number of objective functions is greater than three. Lastly, another existing metaheuristic could be adapted to resolve this problem, so that the results may be compared with those obtained through RIPG.

Data Availability

Instances and Solutions Dataset referred in this paper can be accessed at https://doi.org//10.17632/4p4jcwdwpt.2 (Mendeley Website).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and Jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[2] G. Lebbar, A. El Barkany, A. Jabri, and I. El Abbassi, "Hybrid metaheuristics for solving the blocking flowshop scheduling problem," *International Journal of Engineering Research in Africa*, vol. 36, pp. 124–136, 2018.

[3] Z. I. M. Hassani, A. El Barkany, A. Jabri, I. El Abbassi, and M. Darcherif, "New approach to integrate planning and

scheduling of production system: heuristic resolution," *International Journal of Engineering Research in Africa*, vol. 39, pp. 156–169, 2018.

[4] Z. I. M. Hassani, A. El Barkany, I. El Abbassi, A. Jabri, and M. Darcherif, "New model of planning and scheduling for job-shop production system with energy consideration," *Management and Production Engineering Review*, vol. 10, no. 1, pp. 89–97, 2019.

[5] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[6] J. Knowles, E. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *Computer Engineering and Networks Laboratory (TIK)*, ETH Zurich, Zurich, Switzerland, 2006.

[7] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Fonseca da, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[8] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: on the design of Pareto-compliant indicators via weighted integration," in *Evolutionary Multi-Criterion Optimization*, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., Springer, Berlin, Heidelberg, Germany, pp. 862–876, 2007.

[9] G. Lebbar, I. El Abbassi, A. Jabri, A. El Barkany, and M. Darcherif, "Multi-criteria blocking flowshop scheduling problems: formulation and performance analysis," *Advances in Production Engineering & Management*, vol. 13, no. 2, pp. 136–146, 2018.

[10] G. Minella, R. Ruiz, and M. Ciavotta, "A review and evaluation of multiobjective algorithms for the flowshop scheduling problem," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 451–471, 2008.

[11] H. Amirian and R. Sahraeian, "Multi-objective differential evolution for the flow shop scheduling problem with a modified learning effect," *International Journal of Engineering - Transactions C: Aspects*, vol. 27, no. 9, p. 1395, 2014.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and Elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[13] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 119–124, Nagoya, Japan, May 1996.

[14] N. Karimi and H. Davoudpour, "A high performing metaheuristic for multi-objective flowshop scheduling problem," *Computers & Operations Research*, vol. 52, pp. 149–156, 2014.

[15] G. Lebbar, I. El Abbassi, A. El Barkany, A. Jabri, and M. Darcherif, "Solving the multi objective flow shop scheduling problems using an improved NSGA-II," *International Journal of Operations and Quantitative Management*, vol. 24, no. 3, pp. 211–230, 2018a.

[16] B.-B. Li and L. Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 3, pp. 576–591, 2007.

[17] T. Pasupathy, C. Rajendran, and R. K. Suresh, "A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs," *International Journal of Advanced Manufacturing Technology*, vol. 27, pp. 804–815, 2006.

[18] D. Y. Sha and H.-H. Lin, "A particle swarm optimization for multi-objective flowshop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 45, no. 7-8, pp. 749–758, 2009.

[19] M. A. Allouche, "Manager's preferences modeling within multi-criteria flowshop scheduling problem: a metaheuristic approach," *International Journal of Business Research and Management*, vol. 1, no. 2, pp. 33–45, 2010.

[20] A. Jabri, A. El Barkany, and A. El khalfi, "Multi-pass turning operation process optimization using Hybrid Genetic-Simulated Annealing algorithm," *Modelling and Simulation in Engineering*, vol. 2017, Article ID 1940635, 10 pages, 2017.

[21] S.-W. Lin and K.-C. Ying, "Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm," *Computers & Operations Research*, vol. 40, no. 6, pp. 1625–1647, 2013.

[22] T. K. Varadharajan and C. Rajendran, "A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs," *European Journal of Operational Research*, vol. 167, no. 3, pp. 772–795, 2005.

[23] M. Gravel, W. L. Price, and C. Gagné, "Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic," *European Journal of Operational Research*, vol. 143, no. 1, pp. 218–229, 2002.

[24] A. Rabanimotlagh, "An efficient ant colony optimization algorithm for multiobjective flow shop scheduling problem," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 5, no. 3, pp. 598–604, 2011.

[25] M. Ciavotta, G. Minella, and R. Ruiz, "Multi-objective sequence dependent setup times permutation flowshop: a new algorithm and a comprehensive study," *European Journal of Operational Research*, vol. 227, no. 2, pp. 301–313, 2013.

[26] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "A hybrid TP + PLS algorithm for bi-objective flow-shop scheduling problems," *Computers & Operations Research*, vol. 38, no. 8, pp. 1219–1236, 2011.

[27] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, Part I: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[28] Y. Yuan, H. Xu, and B. Wang, "An improved NSGA-III procedure for evolutionary many-objective optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 661–668, New York, NY, USA, 2014.

[29] J. Kuster, D. Jannach, and G. Friedrich, "Applying local rescheduling in response to schedule disruptions," *Annals of Operations Research*, vol. 180, no. 1, pp. 265–282, 2008.

[30] K. Lee, F. Zheng, and M. L. Pinedo, "Online scheduling of ordered flow shops," *European Journal of Operational Research*, vol. 272, no. 1, pp. 50–60, 2019.

[31] W. Liu, Y. Jin, and M. Price, "New scheduling algorithms and digital tool for dynamic permutation flowshop with newly arrived order," *International Journal of Production Research*, vol. 55, no. 11, pp. 3234–3248, 2017.

[32] Z. Zakaria and S. Petrovic, "Genetic algorithms for match-up rescheduling of the flexible manufacturing systems," *Computers & Industrial Engineering*, vol. 62, no. 2, pp. 670–686, 2012.

[33] A. Pfeiffer, B. Kádár, and L. Monostori, "Stability-oriented evaluation of rescheduling strategies, by using simulation," *Computers in Industry*, vol. 58, no. 7, pp. 630–643, 2007.

[34] R. J. Abumaizar and J. A. Svestka, "Rescheduling job shops under random disruptions," *International Journal of Production Research*, vol. 35, no. 7, pp. 2065–2082, 1997.

[35] S. F. Smith, "Reactive scheduling systems," in *Intelligent Scheduling Systems*, D. E. Brown and W. T. Scherer, Eds., Springer, Boston, MA, USA, 1995.

[36] L. K. Church and R. Uzsoy, "Analysis of periodic and event-driven rescheduling policies in dynamic shops," *International Journal of Computer Integrated Manufacturing*, vol. 5, no. 3, pp. 153–163, 1992.

[37] G. E. Vieira, J. W. Herrmann, and E. Lin, "Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies," *International Journal of Production Research*, vol. 38, no. 8, pp. 1899–1915, 2000.

[38] A. B. Itayef, T. Loukil, and J. Teghem, "Rescheduling a permutation flowshop problems under the arrival a new set of jobs," in *Proceedings of the International Conference on Computers Industrial Engineering*, pp. 188–192, Troyes, France, July 2009.

[39] A. Liefooghe, M. Basseur, J. Humeau, L. Jourdan, and E.-G. Talbi, "On optimizing a bi-objective flowshop scheduling problem in an uncertain environment," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3747–3762, 2012.

[40] W. Liu, Y. Jin, and M. Price, "New meta-heuristic for dynamic scheduling in permutation flowshop with new order arrival," *The International Journal of Advanced Manufacturing Technology*, vol. 98, pp. 1817–1830, 2018.

[41] J. Xiong, L. Xing, and Y. Chen, "Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns," *International Journal of Production Economics*, vol. 141, no. 1, pp. 112–126, 2013.

[42] H. Iima, "Genetic algorithm approach to multiobjective rescheduling on parallel machines," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 139–144, 2005.

[43] B. Zhang, Q. Pan, L. Gao, and Y. Zhao, "MOEA/D for multi-objective hybrid flowshop rescheduling problem," in *Proceedings of the ASME 2018 13th International Manufacturing Science and Engineering Conference*, College Station, TX, USA, June 2018.

[44] X. He, S. Dong, and N. Zhao, "Research on rush order insertion rescheduling problem under hybrid flow shop based on NSGA-III," *International Journal of Production Research*, vol. 58, no. 4, pp. 1161–1177, 2019.

[45] G. E. Vieira, J. W. Herrmann, and E. Lin, "Rescheduling manufacturing systems: a framework of strategies, policies, and methods," *Journal of Scheduling*, vol. 6, no. 1, pp. 39–62, 2003.

[46] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[47] J. Razmi, R. T. Moghaddam, and M. Saffari, "A mathematical model for a flow shop scheduling problem with fuzzy processing times," *Journal of Industrial Engineering*, vol. 3, pp. 39–44, 2009.

[48] P. Fattahi and A. Fallahi, "Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability," *CIRP Journal of Manufacturing Science and Technology*, vol. 2, no. 2, pp. 114–123, 2010.

[49] R. Ramezanian, M. B. Aryanezhad, and M. Heydari, "A mathematical programming model for flow shop scheduling problems for considering Just in time production," *International Journal of Industrial Engineering and Production Research*, vol. 21, no. 2, 2010, https://www.researchgate.net/publication/49591605_A_Mathematical_Programming_Model_for_Flow_Shop_Scheduling_Problems_for_Considering_Just_in_Time_Production.

[50] K. Thörnblad, *Mathematical Optimization in Flexible Job Shop Scheduling, Modelling, Analysis, and Case Studies*, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden, 2013.

[51] M. M. Yenisey and B. Yagmahan, "Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends," *Omega*, vol. 45, pp. 119–135, 2014.

[52] J. Li, Q. Pan, and K. Mao, "A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 279–292, 2015.

[53] F. Yuan and M. Yin, "A novel fuzzy model for multi-objective permutation flow shop scheduling problem with fuzzy processing time," *Advances in Mechanical Engineering*, vol. 11, no. 4, pp. 1–9, 2019.

[54] J. Bukchin and M. Masin, "Multi-objective lot splitting for a single product m-machine flowshop line," *IIE Transactions*, vol. 36, no. 2, pp. 191–202, 2004.

[55] P. M. Chaudhari, D. R. V. Dharaskar, and D. V. M. Thakare, "Computing the most significant solution from Pareto front obtained in multi-objective evolutionary," *International Journal of Advanced Computer Science and Applications*, vol. 1, no. 4, pp. 63–68, 2010.

[56] H. A. Taboada and D. W. Coit, "Data clustering of solutions for multiple objective system reliability optimization problems," *Quality Technology & Quantitative Management*, vol. 4, no. 2, pp. 191–210, 2007.

[57] P. Valledor, A. Gomez, P. Priore, and J. Puente, "Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions," *International Journal of Production Research*, vol. 56, no. 19, pp. 6363–6377, 2018.

[58] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[59] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[60] C. Rajendran, "Heuristics for scheduling in flowshop with multiple objectives," *European Journal of Operational Research*, vol. 82, no. 3, pp. 540–555, 1995.

[61] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

[62] G. Minella, R. Ruiz, and M. Ciavotta, "Restarted iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems," *Computers & Operations Research*, vol. 38, no. 11, pp. 1521–1533, 2011.

[63] S. Hasija and C. Rajendran, "Scheduling in flowshops to minimize total tardiness of jobs," *International Journal of Production Research*, vol. 42, no. 11, pp. 2289–2301, 2004.

[64] P. Perez-Gonzalez and J. M. Framinan, "Scheduling permutation flowshops with initial availability constraint: analysis of solutions and constructive heuristics," *Computers & Operations Research*, vol. 36, no. 10, pp. 2866–2876, 2009.

[65] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 65–74, 1990.

[66] M. A. Adibi, M. Zandieh, and M. Amiri, “Multi-objective scheduling of dynamic job shop using variable neighborhood search,” *Expert Systems with Applications*, vol. 37, no. 1, pp. 282–287, 2010.

[67] S. J. Mason, S. Jin, and C. M. Wessels, “Rescheduling strategies for minimizing total weighted tardiness in complex job shops,” *International Journal of Production Research*, vol. 42, no. 3, pp. 613–628, 2004.

[68] L. Tang, W. Liu, and J. Liu, “A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment,” *Journal of Intelligent Manufacturing*, vol. 16, no. 3, pp. 361–370, 2005.

[69] R. Swaminathan, M. E. Pfund, J. W. Fowler, S. J. Mason, and A. Keha, “Impact of permutation enforcement when minimizing total weighted tardiness in dynamic flowshops with uncertain processing times,” *Computers & Operations Research*, vol. 34, no. 10, pp. 3055–3068, 2007.

[70] A. Abdelhadi and L. H. Mouss, “An efficient hybrid approach based on multi agent system and emergence method for the integration of systematic preventive maintenance policies in hybrid flow-shop scheduling to minimize makespan,” *Journal of Mechanical Engineering Research*, vol. 5, no. 6, pp. 112–122, 2013.

[71] A. Costa, F. A. Cappadonna, and S. Fichera, “A hybrid metaheuristic approach for minimizing the total flow time in A flow shop sequence dependent group scheduling problem,” *Algorithms*, vol. 7, no. 3, pp. 376–396, 2014.

[72] S. Hatami, R. Ruiz, and C. A. Romano, “Two simple constructive algorithms for the distributed assembly permutation flowshop scheduling problem,” in *Managing Complexity*, C. Hernández, A. López-Paredes, and J. M. Pérez-Ríos, Eds., Springer, Cham, Switzerland, 2014.

[73] D. L. de Souza, F. S. Lobato, and R. Gedraite, “A comparative study using bio-inspired optimization methods applied to controllers tuning,” *Frontiers in Advanced Control System*, vol. 7, 2012, http://www.intechopen.com/books/frontiers-in-advanced-control-systems/a-comparative-study-using-bio-inspired-optimization-methods-applied-to-controllers-tuning.

[74] E. Vallada and R. Ruiz, “Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem,” *Omega*, vol. 38, no. 1-2, pp. 57–67, 2010.

[75] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, “The irace package: iterated racing for automatic algorithm configuration,” *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[76] I. Sabuncuoglu and S. Karabuk, “Rescheduling frequency in an FMS with uncertain processing times and unreliable machines,” *Journal of Manufacturing Systems*, vol. 18, no. 4, pp. 268–283, 1999.

[77] P. Czyżżak and A. Jaszkiewicz, “Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization,” *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 1, pp. 34–47, 1998.

[78] H. Ishibuchi, T. Yoshida, and T. Murata, “Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.