J SYST SCI SYST ENG
Vol. 30, No. 3, June 2021, pp. 253–275
DOI: https://doi.org/10.1007/s11518-021-5490-8
ISSN: 1004-3756 (paper), 1861-9576 (online)
CN 11-2983/N

Predictive-reactive Strategy for Flowshop Rescheduling Problem: Minimizing the Total Weighted Waiting Times and Instability

Ayoub Tighazoui, Christophe Sauvey, Nathalie Sauer
Université de Lorraine, LGIPM, F-57000 Metz, France
ayoub.tighazoui@univ-lorraine.fr (✉), christophe.sauvey@univ-lorraine.fr, nathalie.sauer@univ-lorraine.fr

Abstract. Due to the fourth revolution experiencing, referred to as Industry 4.0, many production firms are devoted to integrating new technological tools to their manufacturing process. One of them, is rescheduling the tasks on the machines responding to disruptions. While, for static scheduling, the efficiency criteria measure the performance of scheduling systems, in dynamic environments, the stability criteria are also used to assess the impact of jobs deviation. In this paper, a new performance measure is investigated for a flowshop rescheduling problem. This one considers simultaneously the total weighted waiting time as the efficiency criterion, and the total weighted completion time deviation as the stability criterion. This fusion could be a very helpful and significant measure for real life industrial systems. Two disruption types are considered: jobs arrival and jobs cancellation. Thus, a Mixed Integer Linear Programming (MILP) model is developed, as well as an iterative predictive-reactive strategy for dealing with the online part. At last, two heuristic methods are proposed and discussed, in terms of solution quality and computing time.

Keywords: Rescheduling, flowshop, predictive-reactive strategy, weighted waiting time, stability, weighted completion time deviation

1. Introduction and Literature Review

Thanks to smart technological tools, the customer's practices are changed. Henceforth, a customer can, at any time, create or cancel an order Zhao et al. (2019). This, has a direct influence on the production management, especially the production scheduling (Ivanov et al. 2016, Moeuf et al. 2018, Uhlmann and Frazzon 2018). Most of scheduling problems assume that the work environment is static and that no event occurs during the job execution. In fact, the real production environment is dynamic (Pinedo et al. 2015, Khorsi et al. 2020), and it is subjected to unexpected events that disrupt the established schedule, such as: machine breakdowns, new jobs arrival, jobs cancelation, uncertain processing times, shortage of raw materials. (Li et al. 1003, Sabuncuoglu and Karabuk 1999, Zhang et al. 2003, Katragjini et al. 2013). Consequently, the production companies are forced to quickly react for dealing with this changed situation. Hence, rescheduling process is required for revising the initial schedule in a cost-effective way (Vieira et al. 2003).

According to Vieira et al. (2003), rescheduling is the process of updating an established schedule in response to disruptions. In the literature, some works have already presented literature reviews of developing research for this area, such as: Vieira et al. (2003), Li and Le (2008), Ouelhadj and Petrovic (2009), and recently Uhlmann and Frazzon (2018). Moreover, rescheduling problems are taking interest in studying different types of machine environments, for instance: single machine (Unal et al. 1997, Hall and Potts 2004, Hall et al. 2017), parallel machines (Curry and Peters 2005, Wang et al. 2018, Kovalyov et al. 2016, Tighazoui 2020), jobshop (Biewirth and Mat-

tfeld 1999, Muluk et al. 2003, Mason et al. 2004, Salido et al. 2017, **?**), or open-shop Liu and Zhou (2013). As well, interesting papers on flowshop environment. Katragjini et al. (2013) studied a rescheduling problem on permutation flowshop environments subjecting to simultaneous random disruptions: new jobs arrival, machine breakdowns, and release times delays. The authors considered the makespan as the schedule efficiency measure, and the number of tasks who's the starting times have been altered as the stability measure. They proposed an iterated greedy algorithm for solving the described problem. Rahmani and Ramezanian (2016) considered a flexible flowshop problem with unexpected new jobs arrival. A predictive-reactive strategy is adopted, which consists in generating an initial schedule for minimizing the weighted tardiness as the efficiency measure. After disruption occurrence, a reactive schedule is generated, measuring simultaneously the efficiency and the stability, where the schedule stability is measured by the absolute deviation of completion time between the initial and the new schedule. For other related papers, the reader may refer to Table 1, which presents a bibliography on flowshop rescheduling problems.

To fill the void in existing literature, this work investigates a new performance measure. Firstly, in terms of schedule efficiency, the Total Weighted Waiting Times (*TWWT*) is considered as a criterion. On scheduling literature, the waiting time is the total period of job's waiting in the system Jurcisin and Sebo (2015). Guo et al. (2013) is among the first works that studied the waiting time on a single machine rescheduling problem. The authors considered the sum of waiting times of rework jobs and the original loads as a criterion, they proved that the problem is NP-hard, and proposed a dynamic insert heuristic algorithm of polynomial-time to solve it. They considered then a rescheduling problem with maximum waiting time as an efficiency objective Guo et al. (2016). Their problem is inspired from quartz manufacturing industry, in which waiting times represent the waiting for certain materials before the reheating step in oven. Thus, the authors proved the complexity of the rescheduling problem on a single machine and developed a methodology to decide where to insert the rework jobs in an initial sequence of regular jobs. Guo et al. (2017) formulated two mixed integer programming models for a single machine rescheduling problem with the total waiting time as an objective. The studied problem came also from a quartz glass factory, describing the waiting of materials before the welding step, where the waiting minimization saves the energy consumption. In our work, the *TWWT* is considered for the first time as an efficiency measure, it could be a very helpful and significant, either in hospital or industrial environments. For instance, in production systems, it can be regarded as the waiting period of a job in front of a workstation, considering the job priority as a weight. In hospital systems, it can represent the delay between a patient's arrival and his actual treatment, considering the emergency level as a weight. Kan (2012) is among the first works that defined the *TWWT* and classified it as NP-hard problem for a single machine. To the best of our knowledge, there is not any flowshop rescheduling problem in the literature which discussed this criterion.

Secondly, in terms of schedule stability, the Total Weighted Completion Time Deviation (*TWCTD*) is considered as a criterion. Indeed, the stability measures are used to assess the impact of the jobs deviation (Wu et al. 1993, Pfeiffer et al. 2007, Zhang et al. 2013). In fact, when the schedule is deviated, this matter may generate supplementary costs, such as raw-materials reordering costs, reallocation costs. Rahmani and Ramezanian (2016). He et al. (2020) studied a hybrid flowshop reschedul-

Table 1 Bibliography on Flowshop Rescheduling Problems

Reference	Problem	Disruption type	Performance measure			Resolution method
			Efficiency	Stability	Robustness	
Lodree et al. (2004)	FSRP with due date and release date	JA	Number of tardy jobs	No	No	EADD based on Moore Hodgson algorithm
Yan-hai et al. (2005)	FSRP	JA	Weighted mean flow time of rush orders and the original jobs	No	No	Ant colony optimization
Chaari et al. (2011)	HFSRP	JPV	Makespan	No	Makespan deviation	Genetic Algorithm
Chiu and Shih (2012)	FSRP with preventive maintenance	JA	Makespan	No	No	Johnson rule
El-Bouri (2012)	FSRP with due date	JA	Mean tardiness	No	No	Cooperative Dispatching
Kianfar et al. (2012)	DFFS with SDST	JA	Average tardiness	No	No	Hybrid genetic algorithm
Weng et al. (2012)	HFSRP with due date	JA	Total earliness and tardiness	No	Checked by simulation	Routing strategy
Rahmani et al. (2013)	DFFS	JA	Weighted flow time	The sum of completion times deviation	TCO	Solved by GAMS
Rahmani and Heydari (2014)	FSRP	JA, JPV	Makespan	The sum of completion times deviation	Makespan deviation	FIFO, M-SPT, M-LPT, Johnson rule

Tokola et al. (2014)	FSRP with release date	JA	Tardiness	No	No	Solved by CPLEX
Katragjini et al. (2015)	Permutation FSRP	MB, JA, JRV, JC, JPV, JSM, JDDV, JWV	Makespan, Total weighted tardiness	Number of deviated jobs	No	Iterated Greedy, Local search
Li et al. (2015)	Permutation FSRP	MB, JA, JC, JPV, JRV	Makespan	Number of deviated jobs	No	Discrete teaching learning-based optimization
Li et al. (2015)	HFSRP	MB, JPV	The average sojourn time, earliness, tardiness, the cast-break penalty	Number of altered jobs	No	Hybrid fruit fly optimization algorithm
Nagasawa et al. (2015)	FSRP with setup time	JPV	Peak power consumption and inventory cost	No	Peak power deviation	Solved by Gurobi Optimizer
Tang et al. (2016)	DFFS with unrelated parallel machine	MB, JA	Makespan and energy consumption	No	No	Particle swarm optimization
Han et al. (2017)	Blocking lot streaming FSRP	MB	Makespan and tardiness	The sum of completion times deviation	Objective function deviation	Robust EMO
Liu et al. (2017)	Permutation FSRP	MB, JA	Total flow time	Manager's, customer's, and Shopfloor Operator's dissatisfaction	Expected schedule performance	Hybridized EMO method

Liu et al. (2017)	Permutation FSRP with common due date	JA	Makespan, maximum tardiness	No	No	Heuristic based on Match-up and Real-time strategy
Qin et al. (2018)	HFSRP with unrelated parallel machine	JPV	Delivary deviation	No	No	Ant colony algorithm
Liu (2019)	Outsourcing FSRP	JA	Makespan and outsourcing cost	Number of disrupted jobs	No	Hybrid variable neighborhood search
Peng et al. (2019)	HFSRP	MB, JA, JRV	Makespan	Number of deviated jobs	No	Multi-Start Variable Neighbourhood Descent

MB: Machine breakdowns, JA: Jobs arrivals, JC: Jobs cancellation, JPV: Job processing variation, JRV: Job release variation, JSM: Job Sequence Modification, JDDV: Job Due Date Variation, JWV: Job Weight Variation, FSRP: Flowshop Rescheduling problem, HFSRP: Hybrid Flowshop Rescheduling problem, DFFS: Dynamic Flexible Flow-shop Scheduling, SDST: Sequence Dependent Setup Times, FIFO: First Input First Output, M-SPT: Modified Shortest Processing Time, M-LPT: Modified Longest Processing Time, EADD: Earliest Adjusted Due Date, EMO: Evolutionary Multiobjective Optimization, TCO: Total change in order of jobs

ing problem with job insertion considering the makespan and total the transportation time as an efficiency measure. They also evaluated the system stability by the alteration of job between the initial chosen machine, and the machine chosen after the disruption. Pfeiffer et al. (2007) proposed a new stability measure, combining the starting time deviation which is the difference between the job starting time in the initial sequence and the new one, as well as the actuality penalty related to the deviation of the job starting time from the current time. This stability criterion is also used by Valledor et al. (2018). Furthermore, other related measures have been found, such as: starting time deviation and total deviation penalty Rangsaritratsamee et al. (2004), the amount of operations and starting times operations which have been altered Peng et al. (2018), the absolute positional disruption, the positional disruption and absolute completion time disruption Hoogeveen et al. (2012), the total sequence disruption Yuan and Mu (2007), the maximum time disruption (the delivery times of jobs to customers changes) Liu and Ro (2014). Differently from previous works, this one considers the *TWCTD* as a stability criterion, which consists in evaluating the deviation between the job's completion times in the original schedule and the new one, associating for each job a weight. Thereby, when a job has a high weight (more priority) it will be difficult to disrupt it. This new approach is closer to the operating rooms scheduling in a hospital environment, where the emergency operations has more priority than others Bakker and Tsui (2017).

Vieira et al. (2003) and Herrmann (2006) classified the rescheduling strategies in two basic categories. The first is the dynamic scheduling strategy, which is based on the jobs dis-

patching when it is necessary using the available information at the moment of dispatching. This strategy uses dispatching rules and other heuristics to choose the sequence of jobs that will be proceeded on the machine (El-Bouri 2012, Nurre and Sharkey 2018, Jun et al. 2019). The second is the predictive-reactive rescheduling strategy, which is based on the generation of an initial schedule in a first step, then, updating the schedule at each disruption apparition (Vieira et al. 2003, Duenas and Petrovic 2008, Nouiri et al. 2020). Yang and Geunes (2008) implemented a predictivereactive scheduling strategy on a single machine with uncertain future jobs. They determined the amount of planned idle time for uncertain jobs and their positions in the predictive schedule. Then, the schedule reacts when a disruption occurs, through including the new jobs in the idle times. Gürel et al. (2010) proposed an anticipative scheduling approach, based on a predictive-reactive strategy for a parallel non-identical machines' environment with controllable processing times. Firstly, an initial schedule is generated for minimizing the total manufacturing costs of the jobs. After a disruption caused by machines breakdowns, a reactive schedule is created, and the remaining schedule is repaired. In our study, the predictive-reactive rescheduling strategy is adopted. It is consisting in generating firstly an initial schedule with the objective of minimizing the *TWWT*. After the disruption apparition, a second schedule is generated with the objective of simultaneously minimizing the *TWWT* and *TWCTD*. Both parts of the objective function are associated by a coefficient α (α *efficiency* + (1–α) *stability*), denoted by *efficiency-stability coefficient*. The impact of this coefficient on the system is analyzed hereafter. As the second objective function considers the association of the efficiency and the stability criterion, it plays a more significant role for reducing the actual costs in real life industrial systems. Although some models of flowshop with rescheduling assumption determine new schedules, but they are only based on the classical measures, they may disregard several costs.

This paper addresses a flowshop rescheduling problem under two disruption types, namely: jobs arrival and jobs cancellation. The choice of flowshop environment is justified by its relevance, it is still an important problem from an optimization point of view, as complicated application of flowshop, such as robotic flowshops that have recently studied (Foumani et al. 2020, Foumani and Jenab 2013). A MILP model is developed to describe this problem, as well as an iterative predictive-reactive rescheduling strategy for dealing with the online part. The problem resolution provides, for each solving step, an optimal solution. This rescheduling problem has not been considered in the research literature before, and contributes to operations research area by:

- Studying a new performance measure inspired from real life system, where the *TWWT* measures the system efficiency, and the *TWCTD* evaluates the system stability.
- Developing for this problem a MILP model and an iterative predictive-reactive strategy for dealing with the online part, under two types of disruption.
- Proposing two heuristic methods, which allow to browse even more jobs in a reasonable time.

The rest of paper is organized as follows. In section 2, the problem is described, as well as the proposed methodology. In section 3, the MILP formulation is provided. In section 4, the proposed heuristics are presented and evaluated. Finally, we conclude and propose some perspectives in section 5.

2. Problem Description and Methodology

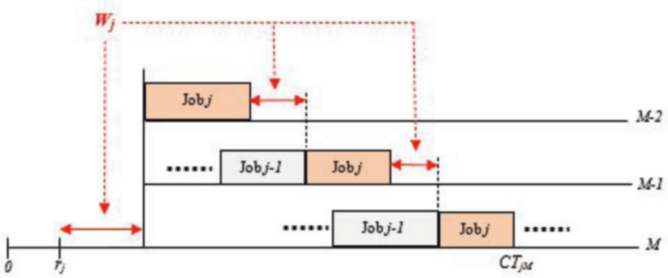

Figure 1 The Waiting Time of Job *j*

2.1 Problem Description

We study a flowshop rescheduling problem, without blocking constraints, under two types of disruptions, new jobs arrival and jobs cancellation. For each job j, is associated a release date r_j, a weight w_j, and a processing time of job j in the machine m, p_{jm}. When a job j starts the execution on the machine m, it is processed to completion time CT_{jm} without interruption. Buffer space capacity between the machines is unlimited. Indeed, there is no blocking constraints in this system. So, each machine will be immediately available to execute an operation, after its previous operation is achieved and the next job is available for treatment.

Efficiency criterion: The waiting time W_j of job j is defined as the total period of job's j waiting in the system Jurcisin and Sebo (2015) (See Figure 1).

The periods during which the job has waited are shown by the red periods in Figure 1. These ones represent the waiting periods before the starting of job on each machine. Thus, we define the waiting time W_j of job j by: $W_j = CT_{jM} - r_j - \sum_{m=1}^{M} p_{jm}$, where CT_{jM} is the completion time of job j on the last machine.

Before the disruptions, we supposed that we have a set of jobs, $N = \{1, 2, \cdots, n\}$, which are already available to be scheduled at the time t=0 in the set of machines F= $\{1, 2, \cdots, M\}$, an initial schedule should be established with the objective of minimizing the $TWWT$, $\sum_{j=1}^{n} w_j W_j$. Thus, a mathematical model is implemented to solve this classical flowshop problem. This initial problem can be represented in standard notation by $F|r_i|\sum w_i W_i$. During the execution of jobs, the initial schedule can be disrupted. In the case of new job arrival, the set of jobs is updated. $N' = \{1, 2, \cdots, n'\}$ is the new set of jobs, it contains the already existing jobs, as well as new arrival one. Then, a new reactive schedule is established updating the existing one. The jobs that have been already initiated by the machine before the disruption date, will keep the same position. However, after the disruption apparition, the objective function will be simultaneously consider the schedule efficiency measured by the $TWWT$, $\sum_{j=1}^{n'} w_j W_j$ and the schedule stability measured by the deviation from the initial schedule, as described hereafter.

Stability criterion: Let CTo_{jm} to be the original completion time of job j at the machine m, it is the completion time when the job j is scheduled for the first time, and considered as the due date communicated to the customer. However, when a disruption occurs, the schedule may change, and a new sequence is established. Hence, the job j actually finishes in the real completion time CT_{jm}. Accordingly, the difference between CTo_{jm} and CT_{jm} can be used to estimate the completion time deviation. Furthermore, a weight w_j is associated to each job j, to penalize more largely the important jobs. In the real system, the association of the weights to the stability criterion makes difficult to deviate the important orders. Thus, the stability objective is defined as $\frac{1}{M}\sum_{j=1}^{n'-n_j}\sum_{m=1}^{M} w_j(CT_{jm} - CTo_{jm})$, referred

to as the *Total Weighted Completion Times Deviation* (*TWCTD*). As can be seen, the stability is measured for only for the jobs existing in the previous schedule (n'–n_j), where n_j is the number of new jobs. Once the new jobs are scheduled in their first sequence, they will be concerned by the stability measure in the ulterior schedules. The stability criterion is divided by M to normalize it with respect to the efficiency criterion.

Besides, for associating both parts of the objective function, the coefficient α *efficiency-stability coefficient* is integrated. Thus, α is the weight of efficiency part and (1–α) the one of stability part, where α is a real number between 0 and 1. The influence of this parameter on the system performance will be studied in the numerical results section. Ultimately, we define the new objective function that simultaneously considers the schedule efficiency and the schedule stability as $\alpha \sum_{j=1}^{n'} w_j W_j + (1-\alpha)\frac{1}{M} \sum_{j=1}^{n'-n_j} \sum_{m=1}^{M} w_j (CT_{jm} - CTo_{jm})$, this function is used at each disruption apparition. The flowshop rescheduling problem can be denoted in standard notation by: $F|r_i|\alpha \sum_i w_i W_i + (1-\alpha)\frac{1}{M} \sum_i \sum_m w_i (C_{im} - Co_{im})$.

In the case of jobs cancellation, the same approach is adopted, considering N' as the set of remaining jobs. In the following, the adopted predictive-reactive strategy is detailed, explaining the rescheduling policy.

2.2 Proposed Predictive-Reactive Rescheduling Strategy

The adopted predictive-reactive rescheduling strategy consists in the predictive phase to solve an initial scheduling problem, assuming that all the jobs' information is available. This initial problem consists in reducing the *TWWT* of the jobs which is considered as a measure of the schedule efficiency. After establishing an initial schedule, the reactive phase starts, it consists of going through the simulation horizon step by step. At each step, the schedule is updated in response to a disruption. In this reactive phase, the methodology measures not only the schedule efficiency, but also the schedule stability through a combination of two criteria, *TWWT* and *TWCTD*. Through this strategy, the decisions are taken locally, which allows to obtain at each rescheduling step an optimal solution, unlike other approaches which use proactive methods consisting in predicting the disruptions occurrences and take them into consideration in the initial schedule formulation Rahmani and Ramezanian (2016).

At each rescheduling step, the schedule can be disturbed by a maximum of two events: the arrival of one job and the cancellation of one job. These two cases can simultaneously or separately appear. In the case of new job arrival, the new arrived job will be combined with the set of uninitiated jobs for rescheduling. In the case of job cancelation, the concerned job is omitted from the set of uninitiated jobs. The set of uninitiated jobs is the set of existing jobs, except the jobs that have already been initiated by the machine before the disruption date. To handle, at each step, the disruptions, we have firstly discretized the finite time horizon [0, T], into periods Δt (see Figure 2). Indeed, Δt is the time period length which represents the time step. The occurrence of disruptions may be possible only at these times. In fact, when a job occurs at time t, this date will be its release date. In order to simplify the calculations, it is assumed that Δt = 1 unit of time, the same time unit that we used for the jobs data p_{jm} and r_j.

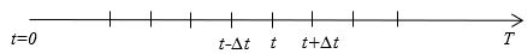

Figure 2 Time Discretization

To simultaneously handle both types of disruptions, the two-hereafter binary variables are introduced, they define the disruption types.

Algorithm 1: handle both types of disruptions

```
Solve the initial problem;
Obtain a LIST = {J_1, ··· , J_n};
Memorize solution;
for (t in 1 to T) do
    if β(t) = 1 then
        Cancel the concerned job from the LIST;
        Solve the new problem;
    end
    if θ(t) = 1 then
        Add the new job to the LIST;
        Solve the new problem, with Constraint 15;
    end
    Memorize solution;
end
```

$$\theta(t) = \begin{cases} 1, & \text{if a new job is occurred} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$\beta(t) = \begin{cases} 1, & \text{if a job is cancelled} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

In each step, the state of $\theta(t)$ and $\beta(t)$ is randomly generated. It is assumed that one job arrives when $\theta(t)$ =1, and one job is canceled when $\beta(t)$=1. Algorithm 1 describes the strategy for handling both types of disruptions.

This algorithm traverses the time horizon step by step, and checks whether a job arrives or a job is canceled, thanks to the state of $\theta(t)$ and $\beta(t)$. If the state of one of these variables is equal to one, this means that a disruption occurs. Consequently, the algorithm updates the set of jobs (LIST) and reschedules the new set of jobs. If, at the same time, a job is canceled and another job arrives, the methodology solves the cancellation problem. Then, without increment the time, it adds the new job and solves the problem. Constraint 15 is used in the case of job arriving. It consists of having the completion time bigger than the original one. This is explained in detail in the MILP formulation section. Figure 3 summarizes the implemented predictive-reactive strategy.

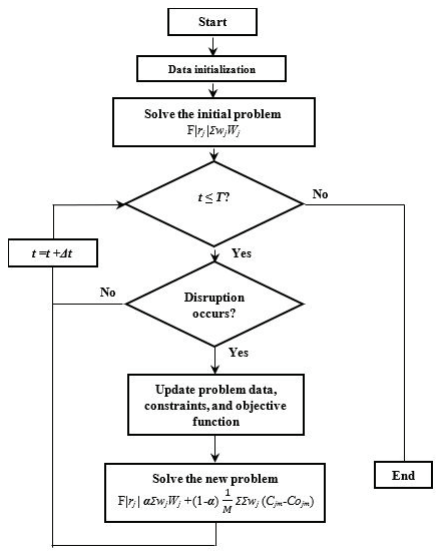

Figure 3 The Proposed Predictive-reactive Strategy

3. MILP Formulation

A MILP model is implemented in this section. This model is based on the predictive-reactive strategy previously described. It is formulated in two phases, the first one is the offline phase, it presents the mathematical model before the occurrence of disruptions, the second one is the online phase, it presents the mathematical model generated after the occurrence of disruptions. In Section 3.3, the numerical results of the MILP resolution are presented.

3.1 Offline Mathematical Model

To build our offline mathematical model, we adapted a flowshop environment model with mixed blocking constraints presented in Trabelsi et al. (2012), in which they minimized the makespan, to consider now the total weighted waiting times as the objective, without blocking constraints. The parameters, variables and constraints are given below:

Parameters:

N: set of jobs $\{1, 2, \cdots, n\}$
K: set of positions $\{1, 2, \cdots, n\}$
F: set of machines $\{1, 2, \cdots, M\}$
j: index of job, $j = 1, 2, \cdots, n$
k: index of position, $k = 1, 2, \cdots, n$
m: index of machine, $m = 1, 2, \cdots, M$
w_j: weight of job j
r_j: release date of job j
p_{jm}: processing time of the job j on the machine m
$bigM$: Big value.

Decision variables:

$$X_{jk} = \begin{cases} 1, & \text{if the job } j \text{ is assigned to } k^{th} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

S_{km}: Starting time of the job in k^{th} position on the machine m.
C_{km}: Completion time of the job in k^{th} position on the machine$m$.
CT_{jm}: Completion time of job j on the machine m.
W_j: Waiting time of job j.

Objective function:

$$\min \sum_{j=1}^{n} w_j W_j \text{ s.t.}$$

$$\sum_{k=1}^{n} X_{jk} = 1, \forall j \in N \quad (3)$$

$$\sum_{j=1}^{n} X_{jk} = 1, \forall k \in K \quad (4)$$

$$C_{km} = S_{km} + \sum_{j=1}^{n} p_{jm} X_{jk}, \forall k \in K, \forall m \in F \quad (5)$$

$$S_{km} \geq C_{km-1}, \forall k \in K, \forall m \in \{2, \cdots, M\} \quad (6)$$

$$S_{km} \geq C_{k-1m}, \forall k \in \{2, \cdots, n\}, \forall m \in F \quad (7)$$

$$S_{km} \geq \sum_{j=1}^{n} r_j X_{jk}, \forall k \in K, \forall m \in F \quad (8)$$

$$CT_{jm} \geq C_{km} - bigM(1 - X_{jk}), \quad \forall j \in N, \forall k \in K, \forall m \in F \quad (9)$$

$$CT_{jm} \leq C_{km} + bigM(1 - X_{jk}), \quad \forall j \in N, \forall k \in K, \forall m \in F \quad (10)$$

$$W_j = CT_{jM} - r_j - \sum_{m=1}^{M} p_{jm}, \forall j \in N \quad (11)$$

$$X_{jk} \in \{0, 1\}, \forall j \in N, \forall k \in K \quad (12)$$

$$S_{km}, C_{km}, CT_{jm}, W_j \geq 0, \forall j \in N, \forall k \in K, \forall m \in F \quad (13)$$

Constraint (3) specifies that each job is affected to only one position. Constraint (4) specifies that each position is occupied by only one job. Constraint (5) specifies that, for all machines, the completion time of k^{th} position is equal to the starting time of k^{th} position plus the assigned processing time. Constraint (6) consists in making the starting time of k^{th} position greater than or equal to its completion time in the previous machine. Constraint (7) consists in having, for all machines, the starting time of k^{th} position greater than or equal to the previous position completion time. Constraint (8) consists in making, for all machines, the starting time of k^{th} position greater than or equal to its assigned release date. Constraint (9) defines, for all machines, the completion time of job j, which is greater than or equal to the completion time of its assigned position, where $bigM$ must be sufficiently large. In this model, it is assumed that the value of the $bigM$ corresponds to the Makespan obtained through sequencing jobs in the ascending order of the release dates. It is an upper bound for the jobs' completion time. Inspired by Guo et al. (2017), constraint (10) is included, it is a

cut that allows reducing the computing time. When X_{jk} =1, this constraint helps the system to provide one possible solution which is CT_{jm} = C_{km}. Constraint (11) defines the waiting time of job j, it is depending on the completion time in the last machine, release date, and processing time. Constraint (12) constraints the variable X_{jk} to be a binary decision variable. Constraints (13) are non-negativity constraints, making all decision variables greater than or equal to zero.

3.2 Online Mathematical Model

The second model is generated after occurrence of disruptions. the new objective function will consider the efficiency measure "*TWWT*" combined with the stability measure "*TWCTD*". This combination plays a more significant role for reducing the actual costs in real life industrial systems. Although some models of flowshop with rescheduling assumption determine new schedules, but they are only based on the classical measures, they may disregard several costs. Thereby, in the case of new jobs arrival, the parameters r_j, w_j, and p_{jm} are used in the formulation, as well as nj the number of arrived jobs. The new formulation becomes:

New parameters:
n'=n+nj
N': set of jobs $\{1, 2, \cdots, n'\}$
K': set of positions $\{1, 2, \cdots, n'\}$
j: index of job, $j = 1, 2, \cdots, n'$
k: index of position, $k = 1, 2, \cdots, n'$
CTo_{jm} : original completion time of job j on the machine m (obtained by the resolution of the initial problem).
α: efficiency-stability coefficient.

The decision variables X_{jk}, CT_{jm}, S_{km}, C_{km}, and W_j are also used for this new formulation. Thus, the new objective function is:

$$\min \alpha \sum_{j=1}^{n'} w_j W_j + (1-\alpha)\frac{1}{M} \sum_{j=1}^{n'-n_j} \sum_{m=1}^{M} w_j (CT_{jm} - CTo_{jm})$$

The constraints (3) until (13) are also used for this online model. As well, the constraint (14), which allows saving the in-course sequence:

$$if\, S_{k1} < t_d, X_{jk} = Xo_{jk}, \forall j \in N, \forall k \in K \quad (14)$$

In fact, the disruption occurs while the machine executes a job. Thus, it is noted by t_d the current time when a disruption occurs (see Figure 4).

Then, all the jobs which have begun their treatment on the flowshop system before t_d, must keep the same position until the end. It is also noted by Xo_{jk}, the variable assigning job j to position k in the original schedule. The values of Xo_{jk} are memorized when the jobs are scheduled for the first time. So, constraint (14) specifies that, if the starting time of a position is less than t_d, the position keeps the same job. On each machine, the jobs have the same sequencing. Thus, the starting times of jobs on the first machine are sufficient to indicate the jobs positions. Therefore, constraint (14) is applied only for the first machine. In addition, the preemption is not authorised.

Considering only the efficiency measure, CTo_{jm} is obtained when the job is scheduled for the first time. In this case, the job is positioned in an ideal position, and it will not be moved backward after a rescheduling. Thus, in all rescheduling steps, it is assumed that:

$$CT_{jm} \geq CTo_{jm}, \forall j \in N, \forall m \in F \quad (15)$$

In the case of jobs cancellation, we have used the same formulation, without constraint (15). In fact, when a job is canceled, its position will be occupied by one of its previously following jobs. Therefore, the jobs will be moved backward. In this case, we can actually have $CT_{jm} \leq CTo_{jm}$. On the other hand, the jobs can be canceled, only if the sequence is being executed. Therefore, the condition $max\ (S_{k1}) > t_d$ is also considered in the job's cancellation case.

3.3 Numerical Results

The proposed MILP model has been written on FICO Xpress IVE and the simulations have

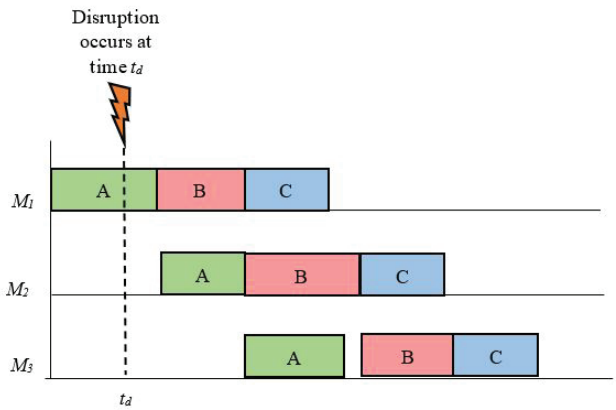

Figure 4 Disruption Occurrence and Disruption Time Definition

been performed on a Core i5 2.40GHz laptop. According to Anand et al. (2017), Xpress solver is considered among the three well-known optimization solvers, it can solve very large optimization problems especially mixed integer. The model implementation has been developed accordingly to the chart presented in Figure 3. The simulations have been tested on the instances described hereafter, and it allowed giving, at each rescheduling step, the optimal solution.

In this subsection, we present the numerical results obtained through the model application on a flowshop problem composed of 5 machines. The data used in this section are presented in Table 2.

Table 2 Parameter Values

Parameters	Values
w_j	$\sim U(1,5)$
p_{jm}	$\sim U(1,4)$ (ut)
T	48 (ut)

The data that we have chosen is adapted for real cases, either in a hospital or in an industrial environment. The simulation will be over an 8-hours' time horizon (480 min) representing a hospital or a factory opening time. The weight values can represent the 5-emergency levels or 5 priority levels of customers. It is assumed also that Δt =1 unit of time (ut), and 1 ut is equivalent to 10 minutes. Thus, T= 8 h = 480 min = 48 ut. The processing times can represent the durations of operations or a product manufacturing time and follow a discrete uniform distribution with values between 1 and 4, obtaining durations of operations between 1 ut (10 minutes) and 4 ut (40 minutes). The values of the variable $\theta(t)$ (respectively $\beta(t)$) are randomly generated with the Bernoulli distribution which gives, at each time t, the value 1 with probability p_θ (respectively p_β), and 0 with probability 1–p_θ (respectively 1–p_β). We first fixed p_β in 0.1 (small cancellation probability), we tested then different values of p_θ.

On the presented results, we tested on 10 different instances the computing time (CPU time) for the execution of all iterations. The sets of tested instances contain respectively 5 initial jobs. They will be disrupted by disruptions over a horizon of T = 48 ut. The appearance probability p_θ is varied for different values (0.2; 0.5; 0.8) to analyze its impact on the CPU time. Then, we calculated the minimum, maximum, average, and standard deviation of CPU time in second. The obtained results are presented in Table 3.

The computing time depends on the appearance frequency of jobs. When p_θ increases, the average and the standard deviation of CPU time increase also, which makes difficult to estimate the computing time for solving the problem. Indeed, when p_θ exceeds 0.5, the

Table 3 Computing Time Study with Different Values of p_θ

α	p_θ ($ANAJ$)	Min CPU	Max CPU	Avg CPU	Std dev CPU
1	0.2 (10 jobs)	1.66	6.68	2.47	1.48
	0.5 (24 jobs)	45.82	1710	348.84	491.09
	0.8 (39 jobs)	9049	> 12h	14772	5303
0.75	0.2 (10 jobs)	1.81	5.47	2.31	1.11
	0.5 (24 jobs)	36.97	4198	850.8	1321
	0.8 (39 jobs)	10153	> 12h	16195	6720
0.5	0.2 (10 jobs)	1.81	7.01	2.49	1.59
	0.5 (24 jobs)	25.14	1188	374.1	387.43
	0.8 (39 jobs)	9014	> 12h	14062	5341

$ANAJ$: The Average Number of Appearing Jobs

set of jobs that must be executed at each iteration increases too much, as well as *CPU* time average values. Thus, when the number of jobs concerned by rescheduling is bigger, the problem resolution is more difficult to obtain in a reasonable time. Indeed, when $p_\theta = 0.8$, the MILP software failed to constantly provide solutions. Only six among the ten tested instances gave solutions in a reasonable time. For the rest, the simulation has been interrupted after 12 hours. The average and standard deviation of *CPU* time is then calculated for the instances in which the solution is obtained.

In fact, when a job with a long processing time occupies the machine and disruptions still appear, the set of jobs concerned by the rescheduling process increases, because it contains a lot of unexecuted jobs that are a waiting since the machine is busy. This explains the big values of the average and standard deviation of *CPU* time when $p_\theta = 0.8$.

The variation of p_β is also studied to interpret its impact on the computing time. p_θ is actually fixed in 0.8, and different values of p_β are tested. We start by $p_\beta = 0.5$, assuming that the order cancellation probability is 50%, $p_\beta = 0.3$ is then tested, until $p_\beta = 0.1$, the last case which has already been tested in the variation of p_θ. The obtained results are presented in Table 4.

When p_β increases, the number of canceled jobs increases also. Although there are many arriving jobs since p_θ=0.8, but when p_β is large, many jobs are canceled also. This makes a diminution of the set of jobs to reschedule at each iteration, helping the MILP to solve the problem in a reasonable time. However, when p_β decreases, the set of jobs to reschedule increases, making difficult the resolution in a reasonable time. This confirms the conclusion deduced from Table 3.Therefore, two heuristics are proposed hereafter to fasten the computing time of the MILP resolution.

4. Heuristic Methods

In this section, two heuristic methods are proposed and evaluated. Accordingly, two subsections are hereafter presented.

4.1 Description of the Heuristic Methods

When a disruption occurs, all the jobs which have begun their treatment on the flowshop system before t_d, must keep the same position. Accordingly, the set of jobs is divided in two parts, the part to fix containing the already initiated jobs by the machine, and the rest of jobs constituting the part to reschedule (see Figure 5).

As concluded in Tables 3 and 4, it is difficult to solve the problem in a reasonable time when the part to reschedule is large. Thus, the larger part to reschedule, the longer computing time becomes. Therefore, to design an efficient heuristic method, the part to reschedule must be reduced, so as to have just a reduced num-

Table 4 Computing Time Study with Different Values of p_β

α	p_β $(ANCJ)$	Min CPU	Max CPU	Avg CPU	Std dev CPU
1	0.5 (24 jobs)	32.94	42.42	37.99	3.01
	0.3 (15 jobs)	50.54	633.02	178.91	236.37
	0.1 (5 jobs)	9049	>12h	14772	5303
0.75	0.5 (24 jobs)	33.07	44.81	40.17	3.52
	0.3 (15 jobs)	47.45	329.29	119.56	89.11
	0.1 (5 jobs)	10153	>12h	16195	6720
0.5	0.5 (24 jobs)	25.03	45.89	36.74	6.43
	0.3 (15 jobs)	45.03	396.27	121.71	107.47
	0.1 (5 jobs)	9014	>12h	14062	5341

$ANCJ$: The Average Number of Canceled Jobs

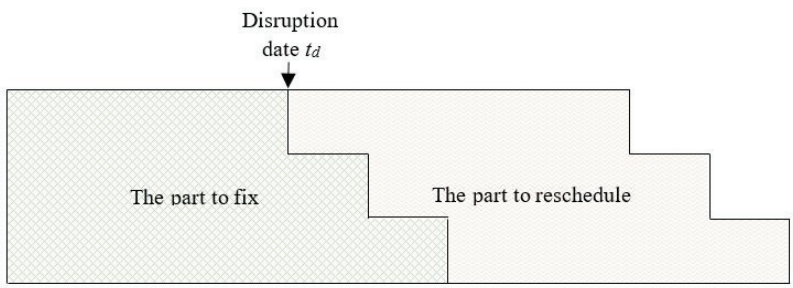

Figure 5 Flowshop System Divided in Two Parts

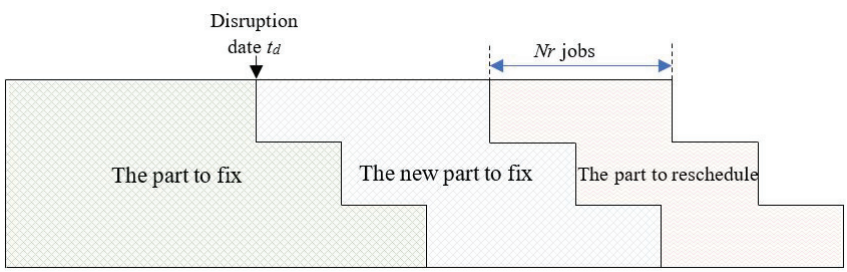

Figure 6 Flowshop System Divided in Three Parts

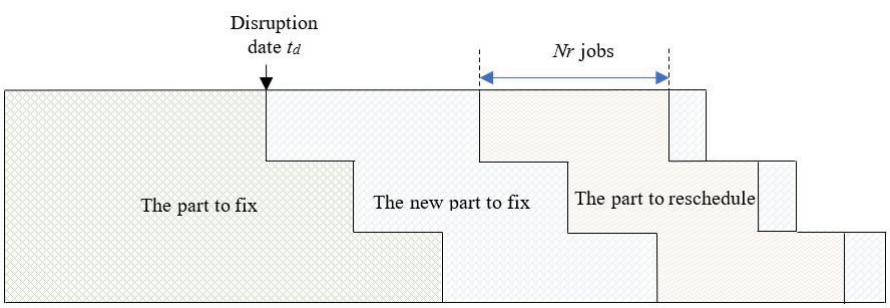

Figure 7 The New Divided Flowshop System when *Nr* is Shifted

ber of jobs that can be solved in a reasonable time. The aim is to act on this part, by reducing the number of jobs to reschedule.

Let Nr be the number of jobs that can be solved in a reasonable time by the MILP model. Following that, the flowshop system will be divided in three parts, the part to fix, the new part to fix and Nr the part to reschedule which includes also the new arrived job (see Figure 6).

Now, the part to reschedule contains only Nr jobs, the rest of jobs will be fixed. Thus, the reduced problem is solved.

After the resolution, the position of the new arrived job is checked, if this one is placed in the left position (1^{st} position in Nr), it means that it is possible that this job could be placed earlier in the optimal solution. In a such case, the set of Nr jobs will be shifted to the left, fixing the last Nr–1 jobs (see Figure 7).

After shifting the Nr jobs, the problem is resolved and the position of the new arrived job is rechecked, if this one still in the left position, the methodology is repeated again, until either the new job is placed on a position different from the left or the heuristic arrives at t_d. Algorithm 2 describes the proposed methodology.

Two starting positions for choosing the Nr jobs to reschedule are possible, either at the beginning or at the end of the sequence. Hence, we denoted by Heuristic 1 the method which traverses the positions from the end of the sequence to the beginning (as described in the algorithm), and Heuristic 2 the reverse, from the beginning to the end of the sequence.

4.2 Evaluation of the Heuristic Methods

As observed in Table 3, when p_θ tends towards 0.8, solutions are more hardly obtained in a reasonable time with the MILP software, in particular when p_β is small. For this special case, the proposed heuristics are tested and evaluated, both in terms of solution quality and computing time, with respect to the solutions obtained with the MILP.

The two heuristics are firstly compared with the MILP solution when p_θ=0.2 and p_θ=0.5. For an appropriate comparison, the original completion times CTo_{jm} obtained by the MILP solution, are used by the heuristics in constraint 15, in order to have at each rescheduling step, the same problem to solve. However, with the high disruptions' probabilities (p_θ=0.8 and p_θ=1), the MILP software is not always able to provide a solution in a reasonable time. *WSPT* (*Weighted Shortest Processing Time*) method consists in scheduling the jobs in ascending order of $\sum_{m=1}^{M} p_{jm}/w_j$, it is considered as an approximate method for this problem, it is used for obtaining the original completion times CTo_{jm}. The aim is to have, at each rescheduling step, the same problem to solve, for comparing the heuristics between them.

4.2.1 Variation of Appearance Frequency

The appearance frequency is varied for studying its impact on solution quality and computing time. Four values of p_θ (0.2; 0.5; 0.8; 1) and three values of α(0.5; 0.75; 1) are tested. The number of improvements established by each heuristic before providing the final solution have also been calculated. It is assumed in this subsection that Nr=5 and $Step$=4. 10 different instances are tested, and the averages are presented in Table 5.

In the low disruptions (p_θ = 0.2), the classical method and the proposed heuristics provide the same results in terms of time and solution quality. Thus, both heuristics do not need to make improvements for providing a solution.

With medium disruptions probability (p_θ = 0.5), the heuristics provide, in a short time, solutions close to the MILP. Heuristic 2 provides better solutions than Heuristic 1. Heuristic 2 acts from the left to the right position. The left positions contain high weight jobs, their rescheduling has a great impact on the solution efficiency. Hence, Heuristic 2 is more effective

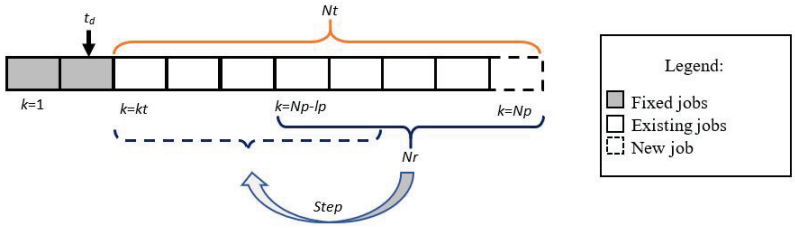

Figure 8 Illustration of the Used Parameters

Algorithm 2: The proposed methodology

Input data: Optimal schedule, New arrived job, Np: Total number of positions, t_d: disruption date, kt: position from which we reschedule, Nt: Total number of jobs that have to be rescheduled, Nr: Number of jobs to reschedule at each step, $Step$: The number of positions to skip.

Initialization: $p = 0, lp, Step, Nr$;
if $Nt \leq Nr$ **then**
  for $(j, k \; in \; 1 \; to \; Np - 1)$ **do**
    if $So(k, 1) \leq t$ **then**
      $X(j, k) = Xo(j, k)$ ▷ ! save the same position ;
    end
  end
  Reschedule the new problem;
  Print the solution;
else
  while $(Np - lp - p \geq kt)$ **do**
    for $(j, k \; in \; 1 \; to \; Np \mid k \; not \; in \; \{Np - lp - p..Np - p\})$ **do**
      $X(j, k) = Xo(j, k)$ ▷ ! save the same position ;
    end
    Reschedule the new problem;
    Print the solution;
    if $(X(newjob, Np - lp - p) = 0 \; or \; X(newjob, kt) = 1)$ **then**
      Break
    else
      Memorize the actual positions;
      if $(Np - lp - p - Step \geq kt)$ **then**
        $p = p + Step$;
      else
        $Step = Np - lp - p - kt$;
        $p = p + Step$;
      end
    end
  end
end

Table 5 Comparison of the Heuristics for Different Values of p_θ

(a) For p_θ=0.2 and p_θ=0.5

		p_θ=0.2(10jobs)			p_θ=0.5(24jobs)		
		*H*1	*H*2	*MILP*	*H*1	*H*2	*MILP*
α=1	Time (*s*)	2.47	2.46	2.53	98.56	61.12	348.09
	Solution	223	223	223	836.67	814.44	803.33
	N°Imprv	0	0	-	15.00	12.29	-
α=0.75	Time (*s*)	2.51	2.47	2.49	71.89	71.29	932.17
	Solution	175.54	175.54	175.54	672.12	655.91	653.95
	N°Imprv	0	0	-	13.89	13.67	-
α=0.5	Time (*s*)	2.51	2.49	2.55	44.82	71.47	387.89
	Solution	129.58	129.58	129.58	658.59	577.23	558.52
	N°Imprv	0	0	-	13.44	16.44	-

(b) For p_θ=0.8 and p_θ=1

		p_θ=0.8(39jobs)			p_θ=1(48jobs)		
		*H*1	*H*2	*WSPT*	*H*1	*H*2	*WSPT*
α=1	Time (*s*)	941	785	0.53	1954	2871	0.76
	Solution	2657.11	2566.89	2590.89	3988.67	3648.67	3931.56
	N°Imprv	45.67	51.00	-	58.56	69.78	-
α=0.75	Time (*s*)	706	948	0.56	1545	2956	0.90
	Solution	2374.03	2218.73	2221.46	3768.04	3187.79	3421.61
	N°Imprv	41.33	52.56	-	56.44	72.00	-
α=0.5	Time (*s*)	337	1082	0.52	750	4142	0.73
	Solution	2335.80	1893.93	1852.02	3839.41	2745.88	2911.66
	N°Imprv	34.78	55.78	-	47.44	72.25	-

*H*1: Heuristic 1, *H*2: Heuristic 2, *WSPT*: Weighted Shortest Processing Time, N°Imprv: Number of Improvements.

than Heuristic 1.

As observed in Table 3, With high disruptions probability, the MILP fails to solve the problem in a reasonable time. Meanwhile, *WSPT* method quickly provides a solution, and was used in Table 5 to generate the original completion times (CTo_{jm}), later used by the heuristics. The aim is to have, at each rescheduling step, the same problem to solve, to compare the heuristics between them. As *WSPT* is an approximate method for this problem, the heuristics can provide, in most of cases, better results than *WSPT*, but Heuristic 2 remains still better than Heuristic 1.

As explained in the heuristics' description, Heuristic 1 improves the obtained solution only if the new job is placed in the left position (1st position in *Nr*). Hence, when α decreases, the stability is more considered, and the left position will already be occupied by an existing job so as not to disturb the established sequence. Therefore, when α decreases, Heuristic 1 makes less improvements and takes less time. Contrarily, Heuristic 2 makes more improvements and takes more time when α decreases.

On the other hand, when p_θ increases, the number of improvements increases too, as there are more disruptions in the system.

It can also be observed that the positions of some existing jobs can switch after a disruption. In fact, when a new job is included in a sequence, the existing jobs will not only be shifted but they can also switch their positions.

To illustrate this phenomenon, an example is presented hereafter.

Example of jobs switching after a disruption

In this example, the processing times p_{jm}, the release dates r_j and the weights w_j for 8 jobs are given in Table 6. The first 5 jobs are considered in the initial schedule (in *offline*), for which all information is available, then jobs 6, 7 and 8 arrive dynamically during the execution (in *online*).

Table 6 Initial Problem Data, with Additional Jobs

		offline					*online*		
job$j$		1	2	3	4	5	6	7	8
p_{jm}	M_1	3	5	2	5	2	2	1	3
	M_2	2	2	4	2	3	1	1	1
	M_3	3	2	3	4	1	1	2	5
r_j		0	1	0	1	2	2	3	4
w_j		2	1	3	2	1	2	1	4

The optimal sequences obtained by the MILP are given in Table 7, it is assumed that $\alpha = 1$, to have more disturbance in the system.

In Step 3, the job 8 is appearing with a weight of 4. This high weight forces the MILP to insert the job 8 in an early date (between the jobs 6 and 1). However, the following jobs are switched. Thus, the optimal solution is given by the sequence 3-6-8-1-5-4-7-2 instead of 3-6-8-1-7-4-5-2, where the objective function value is 107. This is a result of switching Job 5 with Job 7. Figure 9 illustrates the Gantt chart for the optimal solutions in Steps 2 and 3.

As can be observed, it is possible for any job to move forward after a disruption, but not less than its original completion time. This is the case of job 5 in this example, which deviates from $CT_{51} = 15$ to $CT_{51} = 12$. The present phenomenon is a result of changing the flowshop structure after each disruption on account of the durations changes. This is also valid in the case of jobs cancellation.

Conclusion: The jobs switching phenomenon shows that the heuristics can, in some cases, miss optimal solutions by fixing a part of the jobs which must be rescheduled.

4.2.2 Variation of *Nr* and *Step*

In this subsection, the values of *Nr* and *Step* are varied to evaluate their impact on solution quality and computing time. Three values of α are tested (0.5; 0.75; 1). It is assumed that $p_\theta = 0.8$ in order to have more disruptions in the system. Thus, *WSPT* is used to get original completion times (CTo_{jm}) to compare the heuristics between them. 10 different instances are tested, and the averages are presented in Table 8. For each problem range, the average total time, the solution and the number of improvements proposed by each heuristic (*H1*, *H2* and *WSPT*) are given, in order to give an idea of the solutions quality, both in terms of accuracy, computing time and existing solution digging.

When *Nr* increases, the number of jobs to be rescheduled increases also. In this case, the system considers more jobs to reschedule at each improvement step. Consequently, the solution quality is better as the research space is larger, but the computing time increases. However, the heuristics use less improvements steps to give a solution. In addition, the solution values are improving for both heuristics. In this case, Heuristic 1 and Heuristic 2 converge towards the optimal solution.

When *Step* increases, the number of positions to skip increases. So, the system can find a solution with less iterations. However, the solution quality is improved as the research space is large, and the new arrived job has the possibility to be compared with more jobs. So, the more *Step* increases, the more we reach a better solution with less improvements. However, *Step* cannot be greater than or equal to *Nr*, because in this case, the new job will not be included in the set of *Nr* jobs. Hence, the optimal case will be when *Step*=*Nr*–1.

The best solutions written in bold are obtained with Heuristic 2 when *Nr*=7, which con-

Table 7 The Optimal Solutions Given by the MILP

	Sequence	Objective function
Step 0: Initial optimal sequence	3-1-4-5-2	36
Step 1: Optimal sequence after occurrence of job 6	3-6-1-4-5-2	54
Step 2: Optimal sequence after occurrence of job 7	3-6-1-7-4-5-2	66
Step 3: Optimal sequence after occurrence of job 8	3-6-8-1-5-4-7-2	104

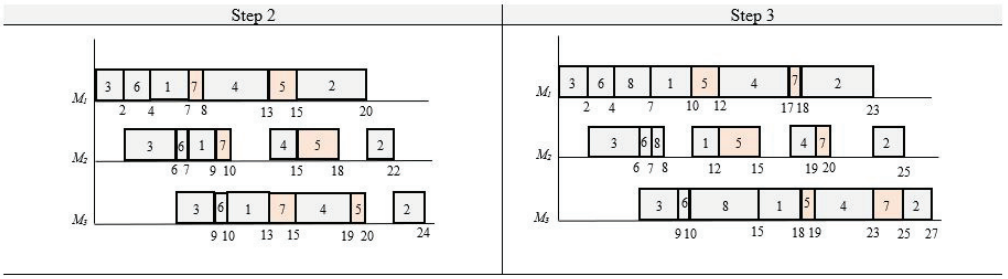

Figure 9 Gantt Chart for the Optimal Solution in $Step$ 2 and 3

Table 8 Comparison of the Heuristics According to the Variation of Nr and $Step$

(a)

		Nr= 5, Step= 3			*Nr= 5, Step= 4*			*Nr= 6, Step= 3*		
		H1	*H2*	*WSPT*	*H1*	*H2*	*WSPT*	*H1*	*H2*	*WSPT*
	Time (s)	847	856	0.56	984	825	0.56	933	1011	0.56
α=1	Solution	2581	2469	2497	2573	2463	2497	2491	2490	2497
	N°Imprv	48.57	45.71	-	46.43	52.43	-	43.57	38.29	-
	Time (s)	814	1230	0.59	724	1019	0.59	987.23	1265	0.59
α=0.75	Solution	2324	2134	2133	2310	2132	2133	2266	2113	2133
	N°Imprv	45.86	57.29	-	42.71	54.29	-	31.57	38.71	-
	Time (s)	697	1580	0.54	345	1124	0.54	756	1883	0.54
α=0.5	Solution	2292	1835	1768	2275	1792	1768	2213	1789	1768
	N°Imprv	36.71	60.29	-	35.43	57.00	-	16.51	52.43	-

(b)

		Nr= 6, Step= 4			*Nr= 6, Step= 5*			*Nr= 7, Step= 3*		
		H1	*H2*	*WSPT*	*H1*	*H2*	*WSPT*	*H1*	*H2*	*WSPT*
	Time (s)	865	649	0.56	720	593	0.56	1350	1070	0.56
α=1	Solution	2479	2471	2497	2471	2468	2497	2468	**2440**	2497
	N°Imprv	32.00	31.10	-	28.00	25.86	-	33.43	30.29	-
	Time (s)	620	857	0.59	517	681	0.59	1246	1308	0.59
α=0.75	Solution	2217	2107	2133	2214	2106	2133	2254	**2104**	2133
	N°Imprv	28.00	31.14	-	23.86	28.00	-	22.29	31.29	-
	Time (s)	443	1293	0.54	436	981	0.54	804	1898	0.54
α=0.5	Solution	2099	1780	1768	2094	1778	1768	2098	**1771**	1768
	N°Imprv	14.71	39.43	-	12.00	35.86	-	8.71	40.86	-

H1: Heuristic 1, *H2*: Heuristic 2, *WSPT*: Weighted Shortest Processing Time, N°Imprv: Number of Improvements.

firms that Heuristic 2 is still better than Heuristic 1, and the increase of Nr allows improving the solution quality. However, as *WSPT* is an approximate method for this problem, in most of cases, Heuristic 2 gives better solutions than *WSPT*.

In this study, we also concluded that Heuristic 2 is still better than Heuristic 1. As well, α still impacts the computing times of the heuristics, as explained in the last subsection.

5. Conclusion and Perspectives

This paper proposed a new optimization criterion to evaluate the performance of a flowshop rescheduling problem. The proposed measure combines simultaneously the schedule efficiency represented by the total weighted waiting times, and the schedule stability by the total weighted completion times deviation. This association of criteria has never been studied in the flowshop rescheduling literature before, and it could be a very helpful and significant measure for real life industrial systems. A MILP model is implemented, as well as an iterative predictive-reactive strategy for dealing with the online part, under two types of disruption: jobs arrival and jobs cancellation. Numerical results show that the model resolution is limited when the set of jobs to reschedule is large. Accordingly, two heuristic methods have been developed to improve the computing time. The heuristics evaluation allows us to observe that:

- Heuristic 2 divides the part to reschedule into subparts. Then, it starts the improvements from the left of the sequence to the right. This one provides better solutions in terms of efficiency than its twin, starting from the right of the schedule back to the left.
- The smaller the part to reschedule, the faster computing time. However, the solution quality reduces.
- The efficiency-stability coefficient α impacts the computing time of the heuristic's resolution. As the stability consists in fixing the already existing jobs, Heuristic 1 makes less improvements when α decreases, contrary to Heuristic 2 that makes more improvements in the same case.
- The jobs switching phenomenon shows that when a new job is included inside a sequence, the existing jobs will not only be shifted, but they can also switch their positions. This is a result of changing the flowshop structure after each disruption on account of the durations changes.

This research can be of great interest, not only for the researchers working in the field of production and operations management facing with flowshop rescheduling problems, but also for a broader audience, such as industrial actors or hospital decision makers. In the further works, we intend to improve the proposed method, by combining both heuristics, choosing at each disruption which heuristics to apply depending on the weight and the job duration. We will also consider a flowshop rescheduling problem with blocking constraints.

Acknowledgments

The authors greatly appreciate the comments of the editor and the referees, which were very helpful for improving the quality of the paper. This work is financially supported by Communauté d'Agglomération Sarreguemines Confluences, France and Région Grand Est, France.

References

Anand R, Aggarwal D, Kumar V (2017). A comparative analysis of optimization solvers. *Journal of Statistics and Management Systems* 20(4): 623-635.

Bakker M, Tsui K L (2017). Dynamic resource allocation for efficient patient scheduling: A data-driven approach. *Journal of Systems Science and Systems Engineering* 26(4): 448-462.

Bierwirth Christian, Mattfeld D C (1999). Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation* 7(1): 1-17.

Chaari T , Chaabane S, Loukil T, Trentesaux D (2011). A genetic algorithm for robust hybrid flow shop scheduling. *International Journal of Computer Integrated Manufacturing* 24(9): 821-833.

Chiu Y, Shih C J (2012). Rescheduling strategies for integrating rush orders with preventive maintenance in a two-machine flow shop. *International Journal of Production Research* 50(20): 5783-5794.

Curry J and Peters B (2005).Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives. *International Journal of Production Research* 43(15): 3231-3246.

Duenas A and Petrovic D (2008). An approach to predictive-reactive scheduling of parallel machines subject to disruptions. *Annals of Operations Research* 159(1): 65-82.

El-Bouri A (2012). A cooperative dispatching approach for minimizing mean tardiness in a dynamic flowshop. *Computers* & *Operations Research* 39(7): 1305-1314.

Foumani M, Jenab K (2013). Analysis of flexible robotic cells with improved pure cycle. *International Journal of Computer Integrated Manufacturing* 26(3): 201-215.

Foumani M, Razeghi A, Smith-Miles K (2020). Stochastic optimization of two-machine flow shop robotic cells with controllable inspection times: From theory toward practice. *Robotics and Computer-Integrated Manufacturing* 61: 101822.

Guo Y, Huang M, Wang Q, Leon V J (2016). Single-machine rework rescheduling to minimize maximum waiting-times with fixed sequence of jobs and ready times. *Computers* & *Industrial Engineering* 91: 262-273.

Guo Y, Wang Q, Huang M (2013). Rescheduling with release time to minimize sum of waiting time considering waiting constraint of original loads. *Acta Automatica Sinica* 39(12): 2100-2110.

Guo Y, Xie X (2017). Two mixed integer programming formulations on single machine to reschedule repaired jobs for minimizing the total waiting-time. In *2017 Chinese Automation Congress (CAC)*:2129-2133. IEEE, 2017.

Gürel S, Körpeoğlu E, Aktürk M S (2010). An anticipative scheduling approach with controllable processing times. *Computers* & *Operations Research* 37(6): 1002-1013, 2010.

Hall N G, Liu Z, Potts C N (2017). Rescheduling for multiple new orders. *INFORMS Journal on Computing* 19(4): 633-645.

Hall N G and Potts C N (2004). Rescheduling for new orders. *Operations Research* 52(3): 440-453.

Han Y, Gong D, Jin Y, Pan Q (2019). Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. *IEEE Transactions on Cybernetics* 49(1): 184-197.

Han Y, Gong D, Jin Y, Pan Q (2017). Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. *IEEE Transactions on Cybernetics* 49(1): 184-197.

He X, Dong S, Zhao N (2020). Research on rush order insertion rescheduling problem under hybrid flow shop based on nsga-iii. *International Journal of Production Research* 58(4): 1161-1177.

Herrmann J W (2006). *Handbook of Production Scheduling* Volume 89. Springer Science & Business Media, 2006.

Hoogeveen H, Lenté C, T'kindt V (2012). Rescheduling for new orders on a single machine with setup times. *European Journal of Operational Research* 223(1): 40-46.

Ivanov D, Dolgui A, Sokolov B, Werner F, Ivanova M (2016). A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0. *International Journal of Production Research* 54(2): 386-402.

Jun S, Lee S, Chun H (2019). Learning dispatching rules using random forest in flexible job shop scheduling problems. *International Journal of Production Research* 57(10): 3290-3310.

Jurcisin R, Sebo J (2015). Basic production scheduling concept software application in a deterministic mechanical production environment. *Acta Simulatio* 1(4): 1-4.

Kan A R(2012). *Machine Scheduling Problems: Classification, Complexity and Computations*. Springer Science & Business Media, 2012.

Katragjini K,Vallada E, Ruiz R(2013). Flow shop rescheduling under different types of disruption. *International Journal of Production Research* 51(3): 780-797.

Katragjini K, Vallada E, Ruiz R (2015). Rescheduling flowshops under simultaneous disruptions. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*: 84-91. IEEE, 2015.

Khorsi M, Chaharsooghi S K, Ali B A, and Ali H K (2020). A multi-objective multi-period model for humanitarian relief logistics with split delivery and multiple uses of vehicles. *Journal of Systems Science and Systems Engineering* 29: 360-378.

Kianfar K, Ghomi SMT F, Jadid A O (2012). Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid ga. *Engineering Applications of Artificial Intelligence* 25(3): 494-506.

Kovalyov M Y, Kress D, Meiswinkel S, Pesch E (2016). A parallel machine schedule updating game with compensations and zero risk. Technical report, working paper, National Academy of Sciences of Belarus and University of Siegen, 2016.

Li J Q, Pan Q K, Mao K (2015). A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Engineering Applications of Artificial Intelligence* 37: 279-292.

Li J Q, Pan Q K, Mao K (2015). A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems. *IEEE Transactions on Automation Science and Engineering* 13(2): 932-949.

Li R K, Shyu Y T, Adiga S (1993). A heuristic rescheduling algorithm for computer-based production scheduling systems. *The International Journal of Production Research* 31(8): 1815-1826.

Li Z, Ierapetritou M(2008). Process scheduling under uncertainty: Review and challenges. *Computers* & *Chemical Engineering* 32(4-5): 715-727.

Liu F, Wang S, Hong Y, Yue X (2017). On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. *IEEE Transactions on Engineering Management* 64(4): 539-553.

Liu L (2019). Outsourcing and rescheduling for a two-machine flow shop with the disruption of new arriving jobs: A hybrid variable neighborhood search algorithm. *Computers* & *Industrial Engineering* 130: 198-221.

Liu L, Zhou H (2013). Open shop rescheduling under singular machine disruption. *Computer Integrated Manufacturing Systems* 10(10): 2467-2480.

Liu W, Jin Y, Price M (2017). New scheduling algorithms and digital tool for dynamic permutation flowshop with newly arrived order. *International Journal of Production Research* 55(11): 3234-3248.

Liu Z, Ro Y K (2014). Rescheduling for machine disruption to minimize makespan and maximum lateness. *Journal of Scheduling* 17(4): 339-352.

Lodree E J, Jang W, Klein C M (2004). A new rule for minimizing the number of tardy jobs in dynamic flow shops. *European Journal of Operational Research* 159(1): 258-263.

Mason S J, Jin S, Wessels C M (2004). Rescheduling strategies for minimizing total weighted tardiness in complex job shops. *International Journal of Production Research* 42(3): 613-628.

Moeuf A, Pellerin R, Lamouri S, Tamayo G S, Barbaray R (2018). The industrial management of smes in the era of industry 4.0. *International Journal of Production Research* 56(3): 1118-1136.

Muluk A, Akpolat H, Xu J (2003). Scheduling problems - An overview. *Journal of Systems Science and Systems Engineering* 12(4): 481-492.

Nagasawa K, Ikeda Y, Irohara T (2015). Robust flow shop scheduling with random processing times for reduction of peak power consumption. *Simulation Modelling Practice and Theory* 59: 102-113.

Nouiri M, Bekrar A, Trentesaux D (2020). An energy-efficient scheduling and rescheduling method for production and logistics systems. *International Journal of Production Research* 58(11): 3263-3283.

Nurre S G, Sharkey T C (2018). Online scheduling problems with flexible release dates: Applications to infrastructure restoration. *Computers* & *Operations Research* 92: 1-16.

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

Peng K, Pan Q K, Gao L, Li X, Das S, Zhang B (2019). A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm and Evolutionary Computation* 45: 92-112.

Peng K, Pan Q K, Gao L, Zhang B, Pang X (2018). An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process. *Computers* & *Industrial Engineering* 122: 235-250.

Pfeiffer A, Kádár B, Monostori L (2007). Stability-oriented evaluation of rescheduling strategies, by using simulation. *Computers in Industry* 58(7): 630-643.

Pinedo M, Zacharias C, Zhu N (2015). Scheduling in the service industries: An overview. *Journal of Systems Science and Systems Engineering* 24(1): 1-48.

Qin W, Zhang J, Song D (2018). An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time. *Journal of Intelligent Manufacturing* 29(4):891-904.

Rahmani D, Heydari M (2014). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems* 33(1): 84-92.

Rahmani D, Heydari M, Makui A, Zandieh M (2013). A new approach to reducing the effects of stochastic disruptions in flexible flow shop problems with stability and nervousness. *International Journal of Management Science and Engineering Management* 8(3): 173-178.

Rahmani D, Ramezanian R (2016). A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers* & *Industrial Engineering* 98: 360-372.

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

Sabuncuoglu I, Karabuk S (1999). Rescheduling frequency in an fms with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems* 18(4): 268-283.

Salido M A, Escamilla J,Barber F, Giret A (2017). Rescheduling in job-shop problems for sustainable manufacturing systems. *Journal of Cleaner Production* 162: S121-S132.

Tang D, Dai M, Salido M A, Giret A (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an

improved particle swarm optimization. *Computers in Industry* 81: 82-95.

Tighazoui A, Sauvey C, Sauer N (2020). New efficiency-stability criterion in a rescheduling problem with dynamic jobs weights.In *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*, volume 1:475-480. IEEE, 2020.

Tokola H, Ahlroth L, Niemi E (2014). A comparison of rescheduling policies for online flow shops to minimize tardiness. *Engineering Optimization* 46(2): 165-180.

Trabelsi W, Sauvey C, Sauer N (2012). Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems. *Computers* & *Operations Research* 39(11): 2520-2527.

Uhlmann I R, Frazzon E M (2018). Production rescheduling review: Opportunities for industrial integration and practical applications. *Journal of Manufacturing Systems* 49: 186-193.

Unal A T, Uzsoy R, Kiran A S (1997). Rescheduling on a single machine with part-type dependent setup times and deadlines. *Annals of Operations Research* 70: 93-113.

Valledor P, Gomez A, Priore P, Puente J (2018). Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions. *International Journal of Production Research* 56(19): 6363-6377.

Vieira G E, Herrmann J W, Lin E (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling* 6(1): 39-62.

Wang D, Yin Y, Cheng TCE (2018). Parallel-machine rescheduling with job unavailability and rejection. *Omega* 81: 246-260.

Weng W, Wei X, Fujimura S (2012). Dynamic routing strategies for JIT production in hybrid flow shops. *Computers* & *Operations Research* 39(12): 3316-33242.

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

Hu Y H, Yan J Q, Ye F F, Yu J H (2005). Flow shop rescheduling problem under rush orders. *Journal of Zhejiang University-SCIENCE A* 6(10): 1040-1046.

Yang B, Geunes J (2008). Predictive-reactive scheduling on a single resource with uncertain future jobs. *European Journal of Operational Research* 189(3): 1267-1283.

Yuan J, Mu Y (2007). Rescheduling with release dates to minimize makespan under a limit on the maximum sequence disruption. *European Journal of Operational Research* 182(2): 936-944.

Zhang G, Cai X, Wong C K (2003). Scheduling two groups of jobs with incomplete information. *Journal of Systems Science and Systems Engineering* 12(1): 73-81.

Zhang L, Gao L, Li X(2013). A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem.*International Journal of Production Research* 51(12): 3516-3531.

Zhao X, Liu N, Zhao S, Wu J, Zhang K, Zhang R (2019). Research on the work-rest scheduling in the manual order picking systems to consider human factors. *Journal of Systems Science and Systems Engineering* 28(3): 344-355.

Ayoub Tighazoui is a Ph.D. student at the LGIPM (Laboratory of Informatics Engineering, Production and Maintenance) of the "Université de Lorraine" in Metz, France, from which he received his Master's degree in industrial systems engineering in 2018. He also received his engineer degree in industrial engineering in 2014 from ENSA (National School of Applied Sciences) in Safi, Morocco. His areas of expertise include: mathematical modelling, optimisation methods, operations research, scheduling and applications.

Christophe Sauvey is an assistant professor at the LGIPM (Laboratory of Informatics Engineering, Production and Maintenance) of the "Université de Lorraine", in Metz, France. He received his engineer degree in electrical engineering from the Polytechnic National Institute of Grenoble (France) in 1997 and his PhD in 2000, on the modelling and optimisation of switched reluctance motors. He received his accreditation to supervise research on modelling and optimisation methods in 2021. His areas of expertise include mathematical modelling, optimisation methods, supply chain management, healthcare systems, logistics, scheduling and applications.

Nathalie Sauer is full professor at the "Université de Lorraine", Metz, France. She received the M.S. degree in applied mathematics, University of Paris 6, in 1991, the PhD degree in industrial engineering from the University of Metz, France, in 1994, and the HDR (accreditation to supervise research) from the University of Nantes, France, in 2004. In 2005, she joined LGIPM (Laboratory of Informatics Engineering, Production and Maintenance), Metz, and she is currently deputy director of this laboratory. Her research interests include scheduling problems, operations research, modeling, analysis and logistic and production systems optimization.