Katragjini Prifti, K.; Vallada Regalado, E.; Ruiz García, R. (2013). Flow shop rescheduling under different types of disruption. International Journal of Production Research. 51(3):780-797. doi:10.1080/00207543.2012.666856.

Flowshop rescheduling under different types of disruptions

Ketrina Katragjini*, Eva Vallada, Rubén Ruiz

Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática,

Ciudad Politécnica de la Innovación, Edificio 8G. Acceso B. Universitat Politècnica de València,

Camino de Vera s/n, 46022 Valencia, Spain, ketrina@iti.upv.es, {evallada,rruiz}@eio.upv.es

November 29, 2011

Abstract

Almost all manufacturing facilities need to use production planning and scheduling systems to increase productivity and to reduce production costs. Real-life production operations are subject to a large number of unexpected disruptions that may invalidate the original schedules. In these cases, rescheduling is essential for minimizing the impact on the performance of the system. In this work we consider flowshop layouts that have been seldom studied in the rescheduling literature. We generate and employ three types of disruptions that interrupt the original schedules simultaneously. We develop rescheduling algorithms to finally accomplish the twofold objective of establishing a standard framework on the one hand and of proposing rescheduling methods that seek a good trade-off between schedule quality and stability on the other hand.

Keywords: Rescheduling, Uncertainty, Flowshop, Stability

1 Introduction

Although the first studies on quantitative scheduling started to appear in the 1950s (Salveson, 1952), from the early twentieth century, the work of Henry Gantt and other pioneers had started to introduce some formal methods into real manufacturing plant operations (Gantt, 1919). A vast

*Corresponding author. Tel: +34 96 387 99 52, ext: 79952. Fax: +34 96 387 74 99

body of research including a wide range of problem characteristics can be found in the scheduling literature. However, the practical use of scheduling techniques is still meager. King (1976) was one of the first who openly recognized the gap between theory and practice in production scheduling due to the simplification of complex real situations in order to construct mathematical models. The scheduling problem, as commonly defined in academic literature, emphasizes only a limited part of the scheduling task. Following the original schedule in real process facilities exactly as it was designed is practically impossible since it is very difficult to have accurate time estimations for the process data and also because of the unforeseen disturbances that affect the system. Pinedo (2008, pg. 430) enumerates 12 principal differences between the theoretical models and the real-life scheduling problems:

1) New jobs constantly arrive in the system. 2) The rescheduling problem is important. 3) The machine environments are more complicated. 4) The weights (priorities) of jobs fluctuate over time. 5) Preferences in the selection of machines are important. 6) There are restrictions in machine availability according to working shift patterns or timetables. 7) Non linear penalty functions. 8) More than one objective is often considered. 9) The available capacity (workload, work shifts) can be influenced. 10) Processing times do not follow statistical distributions. 11) Processing times on the same machine tend to be highly positively correlated. 12) Processing time distributions may be subject to change due to learning or deterioration.

He remarks that the rescheduling problem is not emphasized sufficiently in the literature of scheduling models. When designing a scheduling system, uncertainties need to be taken into account, since as time goes by, production schedules become inaccurate or infeasible and eventually a new schedule will be needed.

In recent years, more attention has been given to the consideration of uncertainties while designing scheduling models, since in real manufacturing systems it is impossible to know exactly a priori all parameters associated with scheduling decisions. Shop floor conditions, material availability, market demand and other parameters are highly likely to be affected by unexpected disturbances with respect to time. In the rescheduling literature, a plethora of approaches and techniques can be found to deal with uncertainties, focusing mainly on machine breakdowns, new job arrivals and stochastic process times. However, the great part of the existing work addresses these disruptions only independently (Church and Uzsoy, 1992; Allahverdi, 1996; Mehta and Uzsoy, 1999; Vieira et al., 2000; Hall and Potts, 2004; Rangsaritratsamee et al., 2004; Arnaut and Rabadi, 2008; Kopanos et al., 2008). These sources of disturbances and many others such as order cancelations, priority and release time changes due to material unavailability, may affect the current schedule simultaneously. Hence for real-life manufacturing scheduling it turns out to be crucial in coping with different types of unexpected disruptions at the same time.

With respect to the rescheduling approaches found in the literature, many authors employ mathematical models to generate optimal solutions for the considered rescheduling problem (Qi et al., 2006; Kopanos et al., 2008). However, these approaches and results remain well-grounded only

within the specific considered problem with its restrictions, and generally can not be extended to more complex and larger sized problems. Similarly, simulation is employed in several papers to validate the proposed rescheduling methods and algorithms (Sabuncuoglu and Kizilisik, 2003; Pfeiffer et al., 2007), and hence, results must be interpreted in the context of the simulated system. An existing benchmark of problems and disruptions for reproducing the results and comparing other methods with existing ones, does not exist.

Another gap in the scheduling literature is represented by reflecting the economic performance of the scheduling system only with classical efficiency performance measures such as makespan, maximum flow time, earliness, tardiness, etc. It is important to point out that as times goes by, the original schedule will become inaccurate and rescheduling actions will be needed to address new situations. Introducing frequent schedule changes can give rise to additional costs, like setup costs, material handling costs, storage costs, etc. Consequently, it is important to reduce schedule *nervousness*, a term coined by Steele (1975), that started being used in the context of Material Requirement Planning Systems (MRP) and meant significant changes in MRP plans (Vollmann et al., 2005). While efficiency measures are comprehensively studied and have appeared in academic research for decades, only a few studies address the drawbacks of continuously introducing changes in the schedule (Rangsaritratsamee et al., 2004).

In view of the aforementioned considerations, in this work our main objective is to define a rescheduling framework and methods in order to effectively address the rescheduling problem in more realistic manufacturing layouts, where the parameters of the scheduling system are always changing and the production schedules need to be updated continuously. In order to accomplish this goal, we consider flowshop layouts that have been seldom studied in the rescheduling literature, at least when compared to the wealth of literature on the single machine layout. We generate and employ three types of disruptions that simultaneously interrupt the original schedules and we develop rescheduling algorithms to cope with these uncertainties. Therefore, our research has a twofold aim: the first one is to establish a standard framework, that includes a rescheduling benchmark for ensuring reproducibility of the results, and the second one is to propose rescheduling methods that seek a good trade-off between schedule quality and nervousness.

The rest of the paper is organized as follows: In Section 2 we provide a brief review of the most relevant publications in the rescheduling literature. Sections 3 and 4 present a thorough description of the innovative rescheduling framework and the methods proposed in this paper. In Section 5, we present a complete comparative evaluation of the performance of the proposed rescheduling algorithms. We end with some concluding remarks in Section 6.

2 Existing terminology and rescheduling research

As previously discussed, a great deal of effort has been spent in generating optimal production schedules. Many scheduling studies employ a standard three field classification scheme for defin-

ing scheduling problems (Graham et al., 1979). This scheme describes a scheduling problem as a triplet $\alpha|\beta|\gamma$ where α represents the scheduling environment, β the job characteristics and restrictions and γ the objective function. Herrmann et al. (1993) and Pinedo (2008) present a comprehensive reference guide for defining and classifying static scheduling problems. On the contrary, there is not a standard classification scheme for dynamic and stochastic rescheduling problems. There is a variable use of concepts and terminology.

2.1 Terminology definition

The process of modifying an existing production schedule in response to disruptions or other changes is commonly known as *Rescheduling*. Vieira et al. (2003) make an effort to give a standard definition of the terms used in the existing rescheduling literature. They classify the *rescheduling environments* that define the set of jobs to be scheduled into two main groups: static and dynamic. In a static scheduling environment, there is a finite set of jobs to be scheduled, whereas in a dynamic one jobs arrive on a continuous basis. Moreover, a static environment can be broken down into a deterministic one, where all system parameters are exactly known and there is no uncertainty for the future, and stochastic environment, where there is a finite set of jobs, but some variables are uncertain such as the case in which the processing times of tasks are modeled as random variables. In dynamic rescheduling environments, jobs arrive continuously over time. Similarly, the existing *rescheduling strategies* are classified into three main categories: dynamic, robust and predictive-reactive. A dynamic scheduling strategy uses system information to dispatch jobs employing heuristics or priority rules when necessary (Church and Uzsoy, 1992). It is often referred to as online, real-time or completely reactive scheduling (Vieira et al., 2003). The robust scheduling approaches also known as proactive schemes (Sabuncuoglu and Goren, 2009), address the problem of creating a schedule which, when implemented, minimizes the effect of disruptions on the primary performance measure of the schedule. The predictive-reactive policy includes two main phases: generation and control. In the first step, an initial schedule that represents the desired behavior of the shop floor is generated. The second step updates the predictive schedule in response to unexpected system disruptions to minimize their effect on system performance (Akturk and Gorgulu, 1999).
In Vieira et al. (2003) a classification of the three main existing *rescheduling techniques* can also be found. Under the periodic technique, rescheduling actions are only taken periodically at the beginning of the defined rescheduling interval. Under the event driven rescheduling policy, rescheduling actions are taken every time the system is affected by unexpected events and a hybrid rescheduling policy reschedules the system periodically and also when special (or major) events take place. Hozak and Hill (2009) address the impact of replanning and rescheduling frequencies on the system performance and highlight the pros and cons of frequent rescheduling actions.

As stated, the robust rescheduling strategy aims at producing schedules that are able to face disruptions without requiring new schedules. A solution for a scheduling problem is robust

if it can be adapted to external events/disruptions with little changes. An example is a flowshop schedule that has intended gaps where machines are left idle so that new jobs arriving into the system can be processed in those gaps without altering all other previously sequenced jobs. In the last years, schedule robustness has been widely studied. Some recent reviews considering robustness in scheduling can be found in Artigues et al. (2005), Herroelen and Leus (2005), Sabuncuoglu and Goren (2009) and Ghezail et al. (2010), among others.

Most of the published papers about schedule robustness consider proactive scheduling approaches, using measures to quantify the robustness of the solutions according to the capacity to maintain the schedule in a disturbed environment: Yang and Yu (2002), Jensen (2003), Pierreval and Durieux-Paris (2007), Al Kattan and Maragoud (2008), Goren and Sabuncuoglu (2008), among others. However, the majority of the papers about robustness are focused on one single type of disruption and/or simple machine environments, that is, in scheduling, one machine problems. When the number of different events/disruptions increases, the reactive approach seems to be more suitable as it is not possible to obtain a schedule that is robust against any possible type of event. Moreover, when a rescheduling method based on a reactive approach is applied, one of the possible objectives might be to maintain the stability of the original schedule, which usually is also the main objective in robust scheduling. Therefore, proactive and reactive approaches can be combined together: firstly a robust schedule is created in the predictive phase and then, the disruptions are handled at the moment they occur (O'Donovan et al., 1999). In this paper, several disruptions are considered at the same time for the flowshop scheduling problem, so rescheduling based on reactive approaches seem to be the most suitable scheme.

2.2 Disruptions classification

As previously mentioned, the actual performance of manufacturing settings often differs from the planned or scheduled one. The majority of the deviations are negative, i.e., they negatively affect system performance leading to deterioration or infeasibility. The unforseen disturbances, that affect the normal operations of real-life manufacturing settings have been classified into two big categories (Cowling and Johansson, 2002; Vieira et al., 2003):

1. Capacity disruptions: i.e., disturbances related to manufacturing resources like machine breakdowns, unavailability of tools, operators absence, etc.

2. Order disruptions: i.e., job related disturbances like rush jobs, job cancelation, raw material shortage, change in priority, rework, etc.

When disruptions upset system performance or lead to infeasibility, rescheduling is triggered to reduce the impact. Hence, these unexpected events are often defined as *rescheduling factors* (Dutta, 1990). Typical disruptions frequently encountered in manufacturing facilities are, amongst others: machine failures, rush orders, order cancelations, priority and due date modifications, workforce unavailability, material arrival delays, raw materials shortage, delay in transport,

rework, variation of process times, variation of set-up times, outsourcing, machine performance deterioration, etc.

2.3 Existing research

The aforementioned unexpected events have been analyzed in several research studies. Vieira et al. (2003) and Li and Ierapetritou (2008) review in detail rescheduling methods, politics and trends developed to address the problem of dealing with uncertainty in production scheduling. In this section we will present a summary of the research papers on rescheduling with uncertainties focusing especially on single machine, parallel machine, flowshop and jobshop environments and on the types of events considered in every one.

2.3.1 Single machine environment

The single machine scheduling problem is the process of assigning a group of tasks to a single machine or resource with the objective of optimizing one or many performance measures. Single machine models are important since practical scheduling problems with more complicated machine environments are often decomposed into subproblems that deal with single machines. For example a complicated machine environment with a single bottleneck may give rise to a single machine model (Pinedo, 2008). Bean et al. (1991) present a framework for rescheduling production facilities when disruptions, like machine breakdowns, tool unavailability, release or due date changes and order quantity increases, invalidate the original schedule. Their rescheduling strategy is based on matching up with the preschedule after every disruption occurrence. Adiri et al. (1991) consider the problem of single machine scheduling with a single breakdown to minimize stochastically the number of tardy jobs. Church and Uzsoy (1992) address the problem of rescheduling production systems in the face of dynamic job arrivals. Hall and Potts (2004) consider a single machine scheduling problem where a set of original jobs has already been scheduled to minimize some cost objective, and a new set of jobs arrives and creates a disruption. Liao and Chen (2004) address a single machine problem with sequence-dependent setup times under machine breakdowns. They propose a heuristic method to maximize the total setup time (or the total idle time) subject to due date constraints. Qi et al. (2006) analyze the problem of updating a machine schedule when random or anticipated disruptions occur. They focus on cases in which the SPT schedule is optimal for the original problem and analyze single and parallel two-machine problems.

2.4 Parallel machine environment

In the parallel machine scheduling problem, there is a set of n jobs that have to be scheduled on m parallel machines. A bank of machines in parallel is a generalization of a single machine model. Many production stages consist of several of machines in parallel (Pinedo, 2008).

Vieira et al. (2000) present analytical models that predict the performance of rescheduling strategies for parallel machine systems. They consider dynamic job arrivals and setups between job families. Azizoglu and Alagoz (2005) consider a rescheduling problem for parallel machines with breakdowns. They provide a polynomial-time algorithm that finds a set of efficient schedules with respect to two different criteria. Curry and Peters (2005) address the problem of nervousness reduction in parallel machine settings under dynamic job arrivals. Lee et al. (2006) address the problem of two machine scheduling under disruptions with transportation costs considerations. They propose polynomial and pseudo-polynomial algorithms for optimally solving the problem. Arnaut and Rabadi (2008) provide new repair and rescheduling algorithms for the unrelated parallel machine environment. They compare four repair rules and conclude that the FJR method (Fit Job Repair) performed best when schedule quality and stability were simultaneously optimized. Ozlen and Azizoglu (2009) provide a branch and bound algorithm to deal with the parallel machine scheduling problem subject to random machine disruptions. Huatuco et al. (2009) compare the impact of five rescheduling strategies on the performance of a parallel machine setting affected by stochastic machine breakdown arrivals.

2.4.1 Flowshop environment

In a flowshop scheduling problem there are m machines and n jobs that have to be processed in the same order on the m machines. An assembly line is an example of a flowshop. Allahverdi (1996) considers a two-machine proportionate flowshop scheduling problem with random breakdowns and the objective of minimizing the maximum lateness. He demonstrates that if breakdowns occur only in the first machine, the longest processing time policy obtains the best results and when they occur only in the second, the best policy is the shortest processing time. Akturk and Gorgulu (1999) suggest a strategy by which, after machine failures, part of the initial schedule is rescheduled to match up with the preschedule at some point in time. Caricato and Grieco (2008) address the problem of the insertion of new orders in production plans that have already been scheduled with the objective of controlling the number of changes with respect to the existing plan and minimizing the delays due to new job arrivals. They consider a hybrid flowshop setting and single job arrivals to be inserted in the current schedule. Zandieh and Gholami (2009) propose an immune algorithm for makespan minimization in a hybrid flowshop with sequence-dependent setups and machines affected by random breakdowns.

2.4.2 Jobshop environment

In a jobshop scheduling problem there are n jobs and each job visits a number of machines following a predefined route that depends on each job. Abumaizar and Svestka (1997) proposed the Affected Operations Algorithm (AOR) for the jobshop under random machine disruptions problem and showed that it outperformed the right shift procedure or the total rescheduling over all the scenarios. Muhlemann et al. (1982) consider dynamic jobshops affected by continuous job

arrivals in the shop floor, machine breakdowns and uncertainty in measuring the process times. They use simulation to compare the performance of different machine loading rules with respect to several classic jobshop performance measures. Rangsaritratsamee et al. (2004) address the problem of dynamic jobshop scheduling and propose a rescheduling methodology based on periodic rescheduling, in which a multicriteria objective function is used as the fitness function for a genetic local search procedure to generate schedules at each rescheduling point. Subramaniam et al. (2005) analyze the jobshop repairing problem subject to internal and external disturbances. After a careful analysis of seemingly complicated disruptions, they conclude that they can be broken down into a few simple basic steps: insert idle time, insert adjustment time and insert operation. Therefore, they employ the mAOR heuristic (modified affected operations algorithm) that repairs the complicated disruptions by successively performing the sequence of generic repair actions previously mentioned. Pfeiffer et al. (2007) propose a simulation-based evaluation technique for testing, validation and benchmarking of rescheduling methods.

2.5 Existing research conclusions

We can observe that the disruptions considered by the great majority of the current literature are principally machine breakdowns and new job arrivals. Most of the papers study the negative effects of only one type of disruption on system performance, differently from realistic situations in which systems may be affected by several types of events simultaneously. Diverse strategies and approaches have been developed to cope with several unexpected situations, but most of the work is simulation based, and hence must be interpreted in the context of the specific simulated system. There does not exist a body of standard practices, procedures, and rules when dealing with dynamic and stochastic manufacturing settings.
In this work we will present a novel approach for addressing the problem of the event driven rescheduling of a permutation flowshop subject to random simultaneous unexpected events, which has been seldom studied in the literature.

3 Our rescheduling framework

The objective of this work is to propose a rescheduling framework and methods for permutation flowshop environments subject to simultaneous random disruptions. Thus, according to the classification given in Section 2, we will consider a stochastic and dynamic environment, we will implement a predictive-reactive approach and will apply an event driven rescheduling policy. The deterministic flowshop problem (FSP) is one of the most exhaustively studied settings in the scheduling literature. In the FSP we have a set $N = \{1, \ldots, n\}$ of n jobs to be processed on a set $M = \{1, \ldots, m\}$ of m machines. Each job has to be processed on each one of the m machines. All jobs must follow the same route, i.e., they follow the same machine order in the shop, starting from machine 1 and finishing on machine m. The objective is to find a permutation

of jobs π that optimizes a certain criterion. The criterion that is most commonly studied in the literature is the minimization of the maximum completion time, also called makespan ($C_{\max}$), of the production sequence. Let p_{ij} denote the processing time of job j, $j \in N$ on machine i, $i \in M$. In this work processing times are deterministic and are known *a priori*. After completion on one machine, each job joins the queue at the next machine. If all queues are assumed to operate under the *First In First Out (FIFO)* discipline, i.e., the processing sequence on the first machine is maintained throughout the remaining machines, the flowshop is referred to as *permutation flowshop*. The resulting problem is called the permutation flowshop problem (PFSP) and with the makespan criterion it is denoted as $F/prmu/C_{\max}$ (Pinedo, 2008). A schedule S provides the start and completion times of all jobs on every machine. At every rescheduling point, our problem is to find a new schedule such that both efficiency and stability measures are minimized. In this work we propose a predictive-reactive approach made up by two steps: generation and control. In the first step we generate a schedule considering only the deterministic problem, i.e., solving the classical PFSP. We refer to this type of schedule as *predictive schedule* or *baseline* and denote it by B. We have used the Iterated Greedy algorithm (IG) of Ruiz and Stützle (2007) to solve a subset of the standard benchmark testbed of Taillard (1993) for the PFSP with makespan criterion. The IG method makes use of a destruction operator that randomly removes some jobs from the sequence and a construction operator that reinserts the previously removed jobs following the constructive heuristic of Nawaz et al. (1983). The predictive schedules represent the desired behavior of the shop floor but it is highly unlikely that they will be executed exactly as they were developed. Thus, the reactive step updates the predictive schedule in response to unexpected disruptions to minimize their effect on system performance.

3.1 Rescheduling factors generation

In many industrial processes machine failures continuously affect the planned activities. Preventive maintenance may reduce the breakdown rate, but **it** is almost impossible to eradicate this type of disruption from the system. Similarly, other parameters like material availability and market demand are highly likely to undergo modifications and hence, it is crucial to react rapidly and produce new schedules that take into account the new system variables. Three types of events that will disrupt the predictive schedules are generated for this work:

1. Machine breakdowns
2. New job arrivals
3. Job ready time variations

For every baseline B originated in the **predictive** step, we simulate its shop floor execution by generating disruptions randomly at time t, $0 \leq t \leq C_{\max}(B)$ where $C_{\max}(B)$ denotes the makespan of the baseline B. There are several reasons for generating disruptions until the end

of the baseline and not of the revised schedule: Firstly, jobs continuously arrive in the system postponing the completion time of the revised schedule and hence the process of disruption generation would be unending if halted at the completion of each revised schedule. Secondly, we want to generate a confined benchmark of disruptions and assure reproducibility of the results when comparing different rescheduling techniques. Since the revised schedule clearly depends on the algorithm providing the best solution, the disruptions generated after the completion time of the baseline are strongly related to the shopfloor status determined by this algorithm and hence we can not assure the reproducibility of the results, similarly to a simulation process. We try to to avoid lengthy and difficult-to-reproduce simulation processes. Moreover, unless the new job arrival rate is set to a very high level, as time goes by, the number of jobs to be scheduled decreases and the problems resolved at every rescheduling point tend to become trivial.

The reason for having initially considered only these three types of events relies fundamentally on the fact that the concern of the research is not to address all the types of disruptions that may affect manufacturing settings. The objective is to propose and validate a rescheduling framework that initially simultaneously considers three relevant types of events and then, in the future, to extend it including other types of disruptions and improved methods.

3.1.1 Machine disruptions generation

For this research we assume that the breakdown time and interval are not known a priori. We simulate the schedule disruption by generating random machine breakdowns at time t, $0 \leq t \leq C_{\max}(B)$. The downtime duration is determined immediately after the event occurs. Down times are generated using a uniform distribution in the range $U[1, \ldots, 99]$. A job that is preempted due to a machine breakdown, resumes its processing from the point at which the interruption occurred. At the beginning of every machine failure, corrective actions are triggered to cope with the event and to improve system performance.

3.1.2 Arrivals of new jobs

In our research, a dynamic problem scenario is considered by generating job arrivals randomly in the system. Specifically, there is a probability of generating one job arrival at every t, $0 \leq t \leq C_{\max}(B)$. All jobs are characterized by the arrival time, which is the time they enter the system, the ready times that identify the time at which they can be released to the shop floor and the processing times of operations on all shop floor machines. The distribution of the processing times for the new jobs is fixed to $U[1, \ldots, 99]$ following Taillard's processing times generation. When a new job arrives in the system, reactive procedures are prompted to find the best way to introduce the new job in the ongoing schedule.

3.1.3 Job ready times variation

Pinedo (2008) defines the ready or release date as the earliest time at which a job can start its processing. In our work, the release variation scenario is reproduced by simulating random ready time delays for the collection of jobs to be scheduled. According to a probability value PR that denotes the likelihood for jobs to undergo unexpected release variation, at every time t, $0 \leq t \leq C_{\max}(B)$, we generate ready times fixed to $U[1, \ldots, 99]$ for the job starting its processing at t. We will employ $r_j, j = (1, \ldots, n)$ to denote these release times.

3.2 Interaction between the events

The three types of events may occur together, i.e., we may have a machine breakdown, a new job arrival and a release time delay simultaneously. At the beginning of every disturbance, reactive actions are prompted to accommodate the disruptions and to pursue a balance between schedule performance on the one hand, and stability on the other hand. It is important to point out that at every time t at which an event occurs, we can only rearrange the part of the permutation that has not already started its processing on the first machine. Since we are considering a permutation flowshop environment, the order of the jobs in progress on the first machine, must be maintained throughout the remaining machines. With respect to the example of Figure 1, we can change the permutation order of only the jobs 2,4,1. We define the partial fixed sequence including the jobs that have already been executed or are in progress on the first machine at the moment of the disruption by π_{pf}. Similarly, we will denote by π_p the *permutable subsequence* containing the jobs which succession order can be modified. All the disruptions are saved as a rescheduling events benchmark to be used for algorithm comparison purposes and can be downloaded from `http://soa.iti.es/`. This benchmark can be used at any time to rapidly compare different rescheduling techniques and avoids launching long simulation experiments. As previously discussed, there does not exist any similar benchmark of disruptions, even for a single type of event and consequently, it is not possible to do comparisons with the results of other rescheduling methods from the literature in an easy and straightforward way.

3.3 Objective function

We propose a bi-objective performance measure as the objective function to be used for evaluating schedules in the reactive step. Our objective function consists of efficiency measured by makespan, and instability, measured by the number of tasks whose starting times have been altered in the new schedule. As previously pointed out, stability is an important measure for manufacturing settings. Applying standard scheduling methods that only consider shop efficiency (time-based measures such as makespan, maximum flow time etc.), may yield new schedules that significantly deviate from the preschedules, compromising other planning activities based on the baseline like materials management, manpower planning, etc. (Kopanos et al., 2008). Throughout this work,

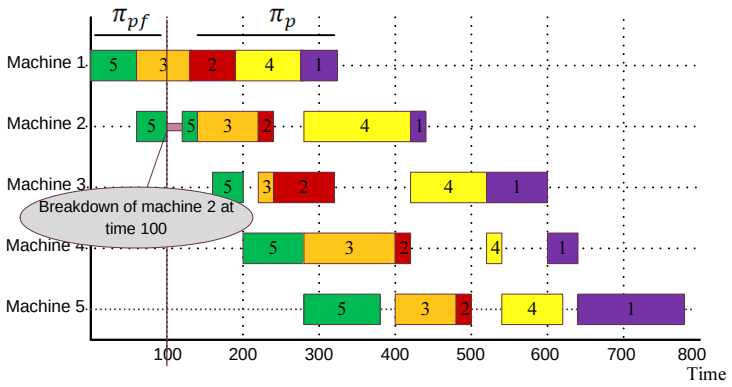

Figure 1: A simplified example of a machine breakdown affecting a flowshop with 5 jobs and 5 machines.

we refer to the schedule before the disruption as *ongoing schedule* and denote it by S and to the adapted schedule after the disruption as *new schedule* and denote it by S^*. Our goal is to simultaneously minimize makespan and instability at every rescheduling point. We apply the weighted sum method to cast the bi-objective problem into a single objective optimization problem by multiplying every single objective by a user-supplied weighting parameter α, whose value is to be chosen according to the relative importance of each objective in the problem. Furthermore, in view of the fact that makespan and instability values are measured in different units and may have different orders of magnitude, we apply a normalization process that scales their values so that they all fall in the range from 0 to 1. Therefore, the objective function to minimize at every rescheduling point, has the following structure:

$$Z = \alpha \cdot M_n(S^*) + (1 - \alpha) \cdot I_n(S^*) \tag{1}$$

where $M_n(S^*)$ and $I_n(S^*)$ represent the normalized makespan and instability, respectively. They are calculated as follows:

$$M_n(S^*) = \frac{C_{\max}(S^*) - \min(C_{\max})}{\max(C_{\max}) - \min(C_{\max})} \tag{2}$$

$$I_n(S^*) = \frac{Q(S^*) - \min(Q)}{\max(Q) - \min(Q)} \tag{3}$$

where

$$Q(S^*) = \sum_{i=1}^{m} \sum_{j=1}^{n} U_{ij} \tag{4}$$

$$U_{ij} = \begin{cases} 1, \text{ if } \mid O^*_{ij}s - O_{ij}s \mid > h \\ 0, \text{ otherwise} \end{cases}$$

In expression (2), $\min(C_{\max})$ and $\max(C_{\max})$ represent the lower and upper bounds for makespan at the moment t of the disruption, respectively. Similarly, in expression (3), $\min(Q)$

and $\max(Q)$ represent the lower and upper bounds for instability at the moment of the event. In expression (3), $Q(S^*)$ represents the instability calculated as the sum of operations whose starting times have been anticipated or delayed in the new schedule S^*. It is important to highlight that the number of jobs n and hence, the number of operations, is not constant, since new jobs arrive continuously in the system. In expression (4), $O_{ij}s^*$ denotes job j's starting time on machine i after the rescheduling action, and $O_{ij}s$ refers to the starting times of the same task before the disturbance. In real-life manufacturing settings jobs' starting times are continuously altered. We use the parameter h to indicate that altering operations starting times up to h time units does not affect schedule stability. Clearly the value of this parameter depends on the type of the environment and the desired level of accuracy for considering or not an operation as affected. By setting its value to 0, we consider the more general situation in which every single change contributes to the instability final value.

3.3.1 Makespan lower and upper bounds

At every rescheduling point, in order to measure the values of the objective function given by all methods, we have to calculate makespan lower and upper bounds identified by $\min(C_{\max})$ and $\max(C_{\max})$ in expression (2). **The makespan** lower bound is calculated as follows:

- Step 1: Determine π_{pf} and π_p.
- Step 2: Determine $C_{\max}(\pi_{pf})$, i.e., the makespan of π_{pf} given by the completion time of the last job of π_{pf} on the last machine. According to the example of Figure 1, the $C_{\max}(\pi_{pf})$ is equal to 480.
- Step 3: We calculate the total processing time of all jobs of π_p on the last machine, and sum it to $C_{\max}(\pi_{pf})$ of the previous step to obtain the makespan lower bound:

$$\min(C_{\max}) = C_{\max}(\pi_{pf}) + \sum_{j=1}^{n(\pi_p)} p_{mj} \tag{5}$$

Figure 2 shows how the makespan lower bound is obtained at the rescheduling point $t = 100$. The lower bound in the example is 720.

The makespan upper bound $\max(C_{\max})$ calculation is straightforward. We determine π_{pf} and π_p. For every job in π_p, we consider that it can not start before the termination of the previous job in the sequence. Figure 3 illustrates this calculation process. The first job in π_p, i.e., the job number 2 can only start after the termination of the job number 3, the job number 4 after the termination of the job number 3 and so on.

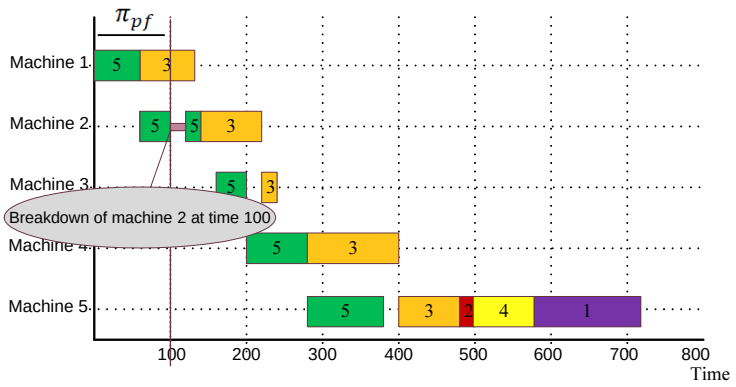

Figure 2: Example of makespan lower bound calculation.

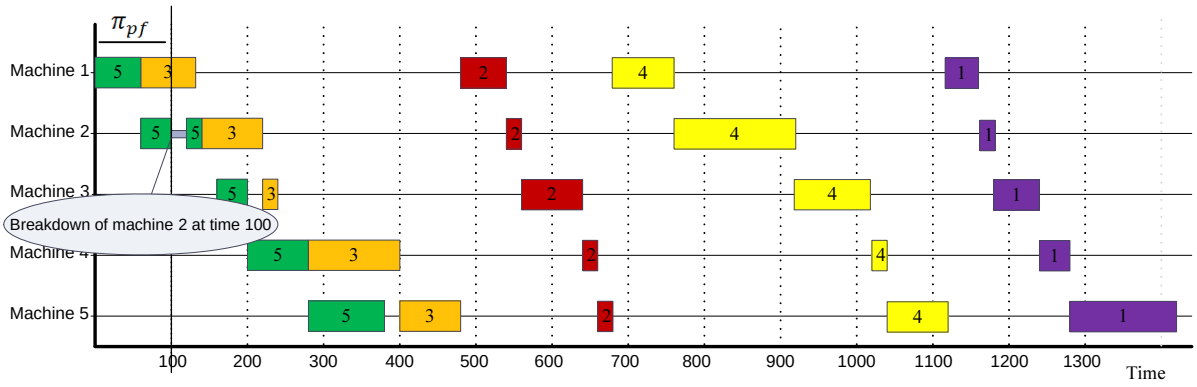

Figure 3: Example of makespan upper bound calculation.

3.3.2 Instability lower and upper bounds

Similarly to makespan, the instability objective has been normalized to limit all values between 0 and 1. In expression (3), $\min(Q)$ and $\max(Q)$ represent the lower and upper bounds for this second objective. Clearly, the trivial lower bound is accounting for not moving any operation, hence, $\min(Q)$ is 0 at every rescheduling point, whereas the upper bound is calculated considering as altered all operations starting times, therefore is calculated as $m \times n$.

It is important to remark that makespan and instability lower and upper bounds are needed for normalizing the objective values in the weighted bi-objective function. Therefore, there is no need for particularly strong lower or upper bounds.

4 Rescheduling methods

One of the objectives of this work is to propose simple rescheduling methods that outperform the typical repair actions taken in manufacturing plants, when faced with unexpected system disturbances. The principal concern of this paper is not to propose state-of-the-art methods for the considered setting but rather to compare reasonable approaches for a complex setting with multiple machines and several types of simultaneous disruptions. In the first part of our work we came up with a testbed of events of three categories: machine breakdowns, new job arrivals and release time delays. Given Taillard's instances and baselines, the predictive-reactive scenario in this work is reproduced by computing the following principal steps:

- Step 1: Load the baseline schedule previously generated and its makespan $C_{\max}(B)$.
- Step 2: At every time t, $0 \leq t \leq C_{\max}(B)$ we examine if there is any event in the testbed that affects the ongoing schedule. If so, the rescheduling methods are triggered to accommodate the disruption. In this work we propose a novel scheme by which all the proposed methods are launched together at every rescheduling point. The main motivation is to make fair comparison of their results, considering that every method will face the same problem difficulty, i.e., the same disruption affecting the same system state. The objective function presented in Section 3.3 is used to evaluate the schedules produced by every method. The best schedule becomes the solution S^* of the rescheduling point and will be used as an ongoing schedule until the next rescheduling point. Therefore, our rescheduling approach has a twofold aim: to accommodate the disruption and, at the same time, to generate a predictive schedule for the future that achieves a good tradeoff between schedule quality and instability.

4.1 Schedule repair

The first method implemented, defined as schedule repair, derives from the companies' common practices to cope with unexpected disruptions. We have implemented three types of disruption

recovery routines, each one depending on the type of event that disturbs the system. The schedule repair routine for machine breakdowns basically performs a right shift on the operations affected by the event. If the machine failure disrupts any operation, its execution is continued after the breakdown interval from the point it was interrupted. The schedule repair routine for a new job arrival, introduces the job at the end of the ongoing schedule. When a release time delay occurs, the repair routine sets the starting time of the affected job to the new release time value. As a result, all affected operations are delayed.

4.2 Local search one single pass

The second method implemented, defined as LS, is a local search algorithm based on the insertion neighborhood. This neighborhood is defined as the set of solutions that can be reached by extracting one job from the sequence π_p and inserting it into all $n(\pi_p)$ positions of the partial permutation consisting of $n(\pi_p) - 1$ jobs. Algorithm 1 presents the pseudocode for this method. The bi-objective function Z is evaluated for all neighbors and the permutation with the lowest value is chosen as the rescheduling method solution. Local search algorithms typically allow one to shift from solution to solution in the space of candidate solutions (the search space) until a local optimal solution is found or a time limit is elapsed. According to Hoos and Stützle (2004), "...LS methods are surprisingly simple, and the respective algorithms are rather easy to understand, communicate and implement. Yet, these algorithms can often solve computationally hard problems very efficiently and robustly." A condition for their good performance, however, is the use of speedups or accelerations, that greatly depend on the problem. Therefore, in order to obtain a faster algorithm, we have implemented some accelerations. The objective function Z consists of the weighted sum of the normalized makespan and instability and hence we need to evaluate jobs starting and completion times in π_p in accordance with the following formulae:

$$
\begin{aligned}
O_{1,\pi_p(1)}s &= \max\left\{O_{1,\pi_{pf}(k)}f; r_1\right\} \\
O_{1,\pi_p(j)}s &= \max\left\{O_{1,\pi_p(j-1)}f; r_j\right\}, \quad j = (2,\ldots,n(\pi_p)) \\
O_{i,\pi_p(j)}s &= \max\left\{O_{i,\pi_p(j-1)}f; O_{i-1,\pi_p(j)}f\right\}, \quad i = (2,\ldots,m), \\
&\qquad j = (2,\ldots,n(\pi_p)) \\
O_{i,\pi_p(j)}f &= O_{i,\pi_p(j)}s + p_{i,\pi_p(j)}, \quad i = (1,\ldots,m), j = (1,\ldots,n(\pi_p))
\end{aligned}
$$

where $O_{ij}f$ is the completion time of job j at machine i. The completion times are evaluated as follows:

$$C_{\pi_p(j)} = O_{m,\pi_p(j)}f, \quad j = (1,\ldots,n(\pi_p))$$

And $C_{\max}$:

$$C_{\max} = \max\left\{C_{\pi_p(1)}, C_{\pi_p(2)}, \ldots, C_{\pi_p(n(\pi_p))}\right\}$$

As algorithm 1 shows, without considering the first evaluation of the first step, Z is evaluated

$n(n-1)$ times. Thus, in order to obtain a faster algorithm, we have implemented some accelerations, taking into account the fact that, for every job, we have to calculate the complete schedule only for the first two neighbors. The third neighbor has the same job in the first position as the second neighbor and can use its starting and finishing times. The fourth neighbor has the same jobs in the first and the second position as the third neighbor and, hence, can use their starting and finishing times, and so on. Note that these accelerations in flowshop problems have been known for some time already, as they where initially proposed by Taillard (1990) for the makespan criterion. However, for other criteria, "partial" accelerations are the only possibility, as explained in detail in Vallada and Ruiz (2010).

Algorithm 1: Local Search one single pass

```
Input: Instance data, makespan and instability upper and lower bounds, π_pf, π_p
Output: permutation π_p^*
begin
    Set Z^* to the current value of Z ;
    Set π_p^* to π_p;
    for j = 1 to n(π_p) do
        for i = 1 to n(π_p) do
            if i ≠ j then
                insert job π_p(j) in position i;
                if new objective value Z < Z^* then
                    set Z^* to Z;
                    set j^* to j;
                    set i^* to i;
    insert job π_p(j^*) in position i^*;
    return π_p^*;
```

4.3 Complete local search

The third method implemented is very similar to the first one: we iterate the single pass local search of the previous section until a local optimal solution with respect to the insertion neighborhood is encountered. Similarly to the single pass procedure, the calculation of the objective function has been accelerated taking into account the features of the insertion neighborhood.

4.4 Iterated Greedy

The Iterated Greedy algorithm (IG), proposed by Ruiz and Stützle (2007), iterates between two phases: the destruction and the construction phase. In the former, some jobs are randomly eliminated from the solution, whereas in the latter, the eliminated jobs are reinserted into the sequence using the NEH construction heuristic of Nawaz et al. (1983). The main loop that incudes these two phases continues until a stopping criterion is met. In our problem, at every rescheduling

point, IG is applied to the partial sequence π_p, that, as already pointed out, is the only part of the affected permutation that we can optimize. Therefore, the destruction procedure is applied to the permutation π_p of $n(\pi_p)$ jobs. At the beginning of the destruction step, d jobs are chosen randomly and are then removed from π_p. As a result of this procedure, we have two subsequences, the first being the partial sequence π_d with $n(\pi_p) - d$ jobs and the second being a sequence of d jobs denoted as π_r. π_r contains the jobs that have to be reinserted into π_d in the opposite order in which they were removed from π_p. The number of jobs d to be extracted in the destruction phase is fixed to 4 for permutations π_p with more than 5 jobs, following Ruiz and Stützle (2007) and some preliminary tests. The construction phase consists in the application of step 3 of the NEH heuristic until obtaining a complete sequence of all $n(\pi_p)$ jobs. Note that in our implementation of IG, we have to deal with the permutation π_{pf} of jobs already scheduled, π_d and π_r. Every job j of π_r is inserted in the position k of π_d that minimizes Z and therefore, the objective function Z is evaluated at each insertion step. Similarly to the two previous rescheduling routines, the construction step has been accelerated considering that when inserting the job j in position k of π_d, all $O_{i,h}s$ and $O_{i,h}f$, $h = \{k-1, k-2, \ldots, 1\}$ were already calculated in the previous insertion step.

5 Computational and statistical analysis

All the proposed methods have been implemented in Delphi 2007 and run on a Dual Core PC with a 2.4 GHz processor and 2 GB of main memory. The stopping criterion for IG is given by a CPU time limit depending on the size of the permutation π_p, since IG needs more time for larger instances. The time limit TL is calculated as follows:

$$TL = t \cdot n(\pi_p) \cdot \frac{m}{2} \tag{6}$$

where t is a time input parameter. For our experiments, t has been fixed to 150 milliseconds as a good compromise between quality and CPU time. To test our rescheduling framework and methods we have employed the well known set of problems of Taillard (1993) that is composed of 120 instances ranging from 20 jobs and five machines to 500 jobs and 20 machines. For our experiments we included the instances with up to 100 jobs and 20 machines. The reason for limiting the Taillard's set of problems considering only the instances with up to 100 jobs relies fundamentally on the considerable computational time required for the larger of the problems. For each of the considered problems we generated the baseline in the predictive phase and performed $R = 5$ independent runs applying the proposed methods at every rescheduling point. The experimental tests were carried out using, in each of Taillard's instances, the events benchmark and three values for the parameter α that represent the relative importance of each objective in the problem. More precisely, the experiments were executed setting α to 0.1, 0.5 and to 0.9. As a measure for the results, we calculate the relative deviation over the best found solution value

in percent, and take the average over the set of events for instances with the same number of jobs and machines. At every rescheduling point i, we evaluate the objective value given by each one of the four methods. Once the best value is known, i.e., the lowest Z value, we calculate the relative percentage deviation over the best solution for each method as follows:

$$RPD_i = \frac{Heu_{sol_i} - Best_{sol_i}}{Best_{sol_i}} \cdot 100$$

Where Heu_{sol_i} is the solution obtained by the method and $Best_{sol_i}$ is the best solution value at rescheduling point i. Obviously, the method or methods that give the best solution will have a RPD of 0. Our objective is to evaluate the average relative percentage deviation ($\overline{RPD}$) for every method, taken over all the set of events grouped by the initial Taillard's problem size.

$$\overline{RPD} = \frac{\sum_{i=1}^{T} RPD_i}{T \cdot R}$$

Where T denotes the total number of rescheduling points within the group of instances with the same initial problem size and R the number of independent runs for each instance. Tables 1, 2 and 3 report the $\overline{RPD}$ for the three analyzed values of α with respect to the initial problem size.

$\alpha = 0.1$	$\overline{RPD}$			
Problem	**REPAIR** [a]	**LS** [b]	**LSLO** [c]	**IG** [d]
Ta20 × 5	46.65	4.56	2.48	**0.00**
Ta20 × 10	35.87	12.32	8.17	**0.00**
Ta20 × 20	38.88	10.36	7.11	**0.00**
Ta50 × 5	129.82	17.83	8.17	**0.00**
Ta50 × 10	104.56	22.35	7.61	**0.00**
Ta50 × 20	57.94	12.13	6.05	**0.00**
Ta100 × 5	320.06	16.84	7.83	**0.00**
Ta100 × 10	217.36	21.45	8.83	**0.00**
Ta100 × 20	131.99	15.93	4.72	**0.00**
Average	**120.35**	**14.86**	**6.77**	**0.00**

[a] Generic repair action [b] Local search one single pass (LS).
[c] LS iterated until local optimum [d] Iterated Greedy of Ruiz and Stützle (2007)

Table 1: $\overline{RPD}$ Average relative percentage deviation over the best solution, $\alpha = 0.1$.

$\alpha = 0.5$	$\overline{RPD}$			
Problem	**REPAIR** [a]	**LS** [b]	**LSLO** [c]	**IG** [d]
Ta20 × 5	42.28	4.02	2.16	**0.00**
Ta20 × 10	30.18	10.39	6.73	**0.00**
Ta20 × 20	35.31	9.38	6.23	**0.00**
Ta50 × 5	121.31	15.02	7.78	**0.00**
Ta50 × 10	81.42	21.77	8.26	**0.00**
Ta50 × 20	57.55	10.88	4.91	**0.00**
Ta100 × 5	232.28	16.56	6.72	**0.00**
Ta100 × 10	186.57	18.88	7.28	**0.00**
Ta100 × 20	107.06	15.59	4.45	**0.00**
Average	**99.33**	**13.61**	**6.06**	**0.00**

[a] Generic repair action [b] Local search one single pass (LS)
[c] LS iterated until local optimum [d] Iterated Greedy of Ruiz and Stützle (2007)

Table 2: $\overline{RPD}$ Average relative percentage deviation over the best solution, $\alpha = 0.5$.

$\alpha = 0.9$	$\overline{RPD}$			
Problem	**REPAIR** [a]	**LS** [b]	**LSLO** [c]	**IG** [d]
Ta20 × 5	59.43	3.56	1.47	**0.00**
Ta20 × 10	30.21	6.86	3.11	**0.00**
Ta20 × 20	28.32	5.12	2.60	**0.00**
Ta50 × 5	113.37	19.27	8.42	**0.00**
Ta50 × 10	100.49	17.71	6.83	**0.00**
Ta50 × 20	41.40	9.62	3.92	**0.00**
Ta100 × 5	187.38	15.27	6.37	**0.00**
Ta100 × 10	127.85	16.49	4.73	**0.00**
Ta100 × 20	75.94	12.22	4.16	**0.00**
Average	**84.93**	**11.79**	**4.62**	**0.00**

[a] Generic repair action [b] Local search one single pass (LS)
[c] LS iterated until local optimum [d] Iterated Greedy of Ruiz and Stützle (2007)

Table 3: $\overline{RPD}$ Average relative percentage deviation over the best solution, $\alpha = 0.9$.

In the first set of experiments, we set the value of α to 0.1 to analyze the case in which more importance is given to the normalized instability, and hence, less weight is given to makespan. As the experimental results show, IG always obtains the lowest value of the objective function Z, across all rescheduling problems and, therefore, its excellent performance does not depend on the type of the disruption that affects the system. On the contrary, the repair mechanism yields the highest values of $\overline{RPD}$ for all rescheduling problems. The other two local search methods based on the insertion neighborhood, obtain solutions with $\overline{RPD}$ in the range of 2.48% and 22.35%. The LSLO procedure, when compared to the schedule repair results, shows an excellent performance. This is a good result considering the simplicity and the speed of the algorithm. In the second and the third set of experiments, the results are very similar to the previous case. As Tables 2 and 3 show, IG yields the best results across all events and instances, demonstrating

that, with respect to the other three methods, IG is able to find the lowest value of the objective function Z, no matter the type of event or the value of α. These attributes are very significant, since we are strongly interested in designing and implementing algorithms that always obtain excellent results, no matter the performance measures or the features of the rescheduling environment. The simple schedule repair is the lowest performing method that yields the highest RPD for all values of α. This is an important result for practical manufacturing settings, where schedule repair is one of the most usual ways to deal with unexpected disruptions. In Table 4, the results are statistically tested by means of an Analysis of Variance (ANOVA, Montgomery, 2007). **The multifactor ANOVA procedure is used to describe the impact of the two factors Algorithm and Objective Weight (Alpha), on the dependent variable RPD. We consider a fixed-effects model since we are interested in only three values (levels) of Alpha:** 0.1, 0.5 **and** 0.9. **For our** 2**-way ANOVA procedure, the possible null hypothesis are: 1) There is no difference in the means of factor Algorithm. 2) There is no difference in the means of factor Alpha. 3) There is no interaction between factors Algorithm and Alpha.**

The ANOVA procedure divides the overall variability observed among all measurements of RPD into several components: a component which measures the variability explained by the Algorithm, a component which measures the variability explained by Alpha, a component attributable to the interactions between the two factors and a residual component, which measures the variability amongst subjects at identical levels of the factors. These components are estimated by the mean square calculated for all effects (simple factors and interactions). When the null hypothesis is true, the effects mean squares and the error mean squares (also known as residual mean square) estimate the same quantity (error variance), and should be of approximately equal magnitude. If the mean square of an effect is significantly larger than the residual mean square, it implies that there is a real effect at a population level and hence the effect is considered to be statistically significant. The F-ratio, that is the ratio between the mean square of an effect and the residual mean square, is used to determine the statistically significance of the factors and their interactions. Alternatively, the p-value can be used to determine if the observed variability in the RPD values may have occurred by chance. The smaller the p-values and the stronger the evidence of the existence of the correspondent effect. In our test, we analyze the following factors: Algorithm with four categorical variants, Alpha with three fixed numerical levels and their interaction. The F-ratios and the p-values are reported in Table 4. Since the 3 **p-values are smaller than** $0,01$**, that is the significance level for this test, these factors have a statistically significant effect on RPD at a** 99% **confidence level. The** 99% **Tukey confidence intervals serve to determine which of the differences in the mean RPD values are statistically significant, since the ANOVA**

Variability source	F-ratio	P-value
A: Algorithm	2364.74	0.0000
B: Alpha	49.48	0.0000
AB: Interaction	38.79	0.0000

Table 4: Analysis of variance for RPD.

F-test only permits to reject the null hypothesis, but does not determine which are the groups with different mean values. The Tukey's HSD (honestly significative difference) intervals are designed for comparing all pair of means. The interval limits are calculated using Student's t distribution and the residual mean square from the ANOVA table. Since the interaction between the factors is statistically significant, it means that the effect of one factor depends on the levels of the other factor. Figure 4 shows the interaction plot with the RPD mean values at all combinations of the two factors and the 99% Tukey's confidence intervals. The analysis confirms that the repair mechanism yields the highest values of $\overline{RPD}$ for all rescheduling problems. The mean RPD values of the repair mechanism depend clearly on the value of the objective weighting factor. The repair routine performance worsens, when greater importance is given to stability. In order to zoom the results of the other three methods, we present another interaction plot in Figure 5, eliminating the repair algorithm from the study. Since the 99% Tukey's confidence intervals do not overlap, we can state, with a 99% confidence level, that the means are statistically different from one another and IG is the best performing method and its results do not depend on the alpha value.

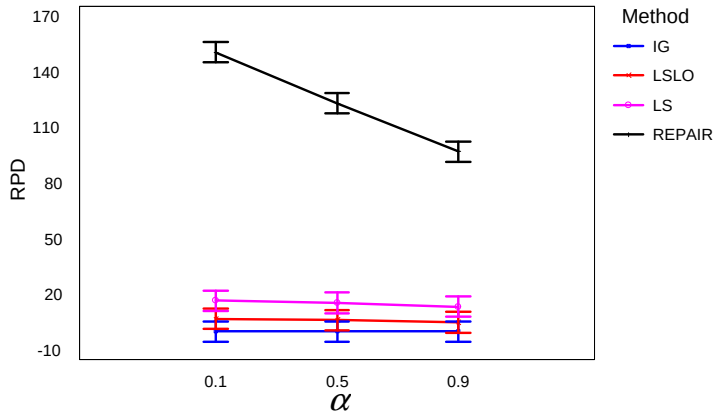

Figure 4: Interactions and 99% Tukey confidence intervals for all rescheduling methods

Table 5 reports the average execution times for all methods taken over all rescheduling problems. The number of jobs to be scheduled varies at every point within every Taillard's instance and the values of α do not affect the methods execution times. As expected, the Iterated Greedy algorithm is the slowest method

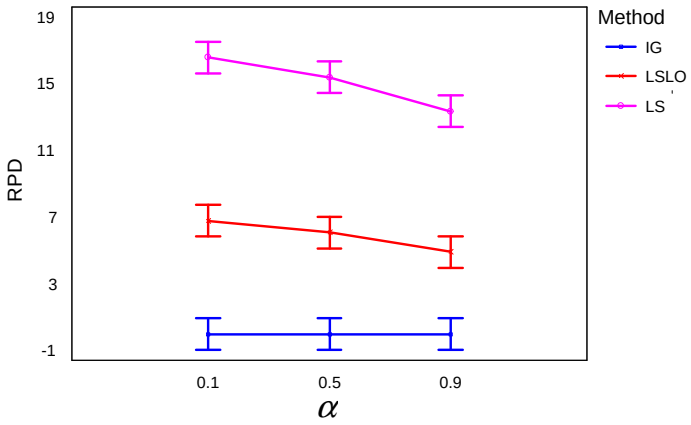

Figure 5: Interactions and 99% Tukey confidence intervals for IG, LSLO and LS methods

among all the rescheduling algorithms implemented, since its stopping criterion is fixed and depends on the number of machines and jobs that can be rescheduled. Its execution times vary from 0.75 **seconds for problems with a small number of jobs, to** 150 **for the largest problems with** 100 **jobs. The IG time limit calculation is given by expression (6) presented in the beginning of this section. The remaining methods are almost immediate.**

Method	Alpha	Minimum	Maximum	Average
IG[a]	0.1	0.75	150.00	62.93
	0.5	0.75	150.00	62.93
	0.9	0.75	150.00	62.93
LSLO[b]	0.1	<0.01	3.13	0.06
	0.5	<0.01	3.03	0.05
	0.9	<0.01	3.05	0.08
LS[c]	0.1	<0.01	0.39	0.02
	0.5	<0.01	0.39	0.02
	0.9	<0.01	0.39	0.02
REPAIR[d]	0.1	<0.01	0.02	<0.01
	0.5	<0.01	0.02	<0.01
	0.9	<0.01	0.02	<0.01

[a] Iterated Greedy of Ruiz and Stützle (2007) [b] LS iterated until local optimum [c] Local search one single pass [d] Generic repair action

Table 5: Minimum, maximum and average execution times (seconds) for all instances and the three tested α values.

6 Conclusions and future research

In the first part of this work we presented a comprehensive review of the rescheduling literature, evidencing the lack of a standard methodology when dealing with dynamic and stochastic manu-

facturing settings and the existence of a gap between theory and practice in production scheduling. In this work we have addressed the problem of flowshop rescheduling under three types of simultaneous random disruptions: machine breakdowns, new job arrivals and release time delays. These disturbances are very common in every day manufacturing operations and negatively affect the overall system performance. Hence, the first achievement of this work is the generation of a new disruption benchmark to be used for comparing the proposed rescheduling methods. We employed Taillard's set of flowshop instances (Taillard, 1993) to implement a predictive-reactive approach and to simulate random schedule disruptions. Note that this work deals with flowshop scheduling problems. However, interesting venues of research open for other flowshop problems (Samarghandi and ElMekkawy, 2011), parallel machine problems (Purushothaman et al., 2009) or even single machine scheduling problems (Valente and Schaller, 2010).

Our next objective has been to find a rescheduling method able to obtain a good trade-off between schedule quality and instability, since the two objectives together reflect the economic performance of the scheduling method. For this purpose, we implemented four simple disruption recovery routines: a simple repair mechanism, a local search procedure of one single pass based on the insertion neighborhood, a local search procedure iterated until a local optimum and the Iterated Greedy algorithm (IG) of Ruiz and Stützle (2007). In this work, we bring forward a novel approach to deal with disruptions: we trigger the four methods together at every rescheduling point. The objective function presented is used to evaluate the schedules produced by every method. The best found solution becomes the solution of the rescheduling point and will be used as the ongoing schedule until the next rescheduling point. Therefore, our rescheduling approach accomplishes the twofold objective of accommodating the disruption and, at the same time, generating a predictive schedule for the future that achieves a good balance between schedule quality and stability. The results of the experiments carried out demonstrate that IG outperforms the rest of the methods, no matter the type of disruption or the relative importance of makespan and instability in the objective function, whereas the repair routine presents the poorest performance for all values of α. The local search procedures, when compared to the schedule repair results, show an excellent performance, considering their simplicity and speed. Therefore, the use of the repair actions is not justified and should be replaced by other rescheduling procedures such as the LS methods when very fast execution times are required and by the IG method, when time is not a concern and higher quality solutions are necessary.

In the future we intend to detect stagnation situations in the IG, in order to reduce its execution times without significantly reducing the quality of the solutions. In this work we considered only three types of disruptions that affect dynamically the shop floor layout, but real-life manufacturing operations are affected by other types of events that need to be accommodated. Therefore, looking forward, we intend to extend the types of events considered, taking into account other critical rescheduling factors. The shop floor layout considered in this work is a permutation flowshop, yielding to very stiff permutations that can be reoptimized only partially at every

rescheduling point. In the future our intention is to consider more realistic situations including other shop floor layouts and consider the bi-objective problem of improving schedules quality and stability by means of Pareto-based multiobjective optimization methods.

Acknowledgments

The authors would like to thank the anonymous referees for their careful and detailed comments which have helped improve this manuscript considerably. This work is partially funded by the Spanish Ministry of Science and Innovation, under the project "SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances" with reference DPI2008-03511/DPI. The authors should also thank the IMPIVA - Institute for the Small and Medium Valencian Enterprise, under the project "OSC" with references IMIDIC/2008/137, IMIDIC/2009/198 and IMIDIC/2010/175.

References

Abumaizar, R. J. and Svestka, J. A. (1997). Rescheduling job shops under random disruptions. *International Journal of Production Research*, 35(7):2065–2082.

Adiri, I., Frostig, E., and Kan, A. H. G. R. (1991). Scheduling on a single-machine with a single breakdown to minimize stochastically the number of tardy jobs. *Naval Research Logistics*, 38(2):261–271.

Akturk, M. S. and Gorgulu, E. (1999). Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, 112(1):81–97.

Al Kattan, I. and Maragoud, R. (2008). Performance analysis of flowshop scheduling using genetic algorithm enhanced with simulation. *International Journal of Industrial Engineering-Theory, Applications and Practice*, 15(1):62–72.

Allahverdi, A. (1996). Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness. *Computers and Operations Research*, 23(10):909–916.

Arnaut, J. P. and Rabadi, G. (2008). Rescheduling of unrelated parallel machines under machine breakdowns. *International Journal of Applied Management Science*, 1(1):75–89.

Artigues, C., Billaut, J., and Esswein, C. (2005). Maximization of solution flexibility for robust shop scheduling. *European Journal Of Operational Research*, 165(2):314–328.

Azizoglu, M. and Alagoz, O. (2005). Parallel-machine rescheduling with machine disruptions. *IIE Transactions*, 37(12):1113–1118.

Bean, J. C., Birge, J. R., Mittenthal, J., and Noon, C. E. (1991). Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39(3):470–483.

Caricato, P. and Grieco, A. (2008). An online approach to dynamic rescheduling for production planning applications. *International Journal of Production Research*, 46(16):4597–4617.

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

Cowling, P. and Johansson, M. (2002). Using real time information for effective dynamic scheduling. *European Journal of Operational Research*, 139(2):230–244.

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.

Dutta, A. (1990). Reacting to scheduling exceptions in FMS environments. *IIE Transactions*, 22(4):300–314.

Gantt, H. L. (1919). *Work, wages, and profits*. The Engineering Magazine Co, New York, second edition.

Ghezail, F., Pierreval, H., and Hajri-Gabouj, S. (2010). Analysis of robustness in proactive scheduling: A graphical approach. *Computers & Industrial Engineering*, 58(2, Sp. Iss. SI):193–198.

Goren, S. and Sabuncuoglu, I. (2008). Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions*, 40(1):66–83.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. Elsevier.

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

Herrmann, J. V., Lee, C., and Snowdown, J. L. (1993). A classification of static scheduling problems. pages 203–253. World Scientific Publishing Co., Singapore.

Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal Of Operational Research*, 165(2):289–306.

Hoos, H. H. and Stützle, T. (2004). *Stochastic Local Search: Foundations and Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, USA.

Hozak, K. and Hill, J. (2009). Issues and opportunities regarding replanning and rescheduling frequencies. *International Journal of Production Research*, 47(18):4955–4970.

Huatuco, L. H., Efstathiou, J., Calinescu, A., Sivadasan, S., and Kariuki, S. (2009). Comparing the impact of different rescheduling strategies on the entropic-related complexity of manufacturing systems. *International Journal of Production Research*, 47(15):4305–4325.

Jensen, M. (2003). Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 7(3):275–288.

King, J. R. (1976). The theory-practice gap in job-shop scheduling. *Production Engineer*, 55(3):137–143.

Kopanos, G. M., Capon-Garcia, E., Espuna, A., and Puigjaner, L. (2008). Costs for rescheduling actions: A critical issue for reducing the gap between scheduling theory and practice. *Industrial and Engineering Chemistry Research*, 47(22):8785–8795.

Lee, C. Y., Leung, J. Y. T., and Yu, G. (2006). Two machine scheduling under disruptions with transportation considerations. *Journal of Scheduling*, 9(1):35–48.

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

Liao, C. J. and Chen, W. J. (2004). Scheduling under machine breakdown in a continuous process industry. *Computers and Operations Research*, 31(3):415–428.

Mehta, S. V. and Uzsoy, R. (1999). Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing*, 12(1):15–38.

Montgomery, D. (2007). *Design and Analysis of Experiments*. John Wiley & Sons, New York, fifth edition.

Muhlemann, A. P., Lockett, A. G., and Farns, C. K. (1982). Job shop scheduling heuristics and frecuency of scheduling. *International Journal of Production Research*, 20(2):227–241.

Nawaz, M., Enscore, E. E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *Omega-International Journal of Management Science*, 11(1):91–95.

O'Donovan, R., Uzsoy, R., and Mckay, K. N. (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 37(18):4217–4233.

Ozlen, M. and Azizoglu, M. (2009). Generating all efficient solutions of a rescheduling problem on unrelated parallel machines. *International Journal of Production Research*, 47(19):5245–5270.

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

Pierreval, H. and Durieux-Paris, S. (2007). Robust simulation with a base environmental scenario. *European Journal Of Operational Research*, 182(2):783–793.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms and Systems*. Springer, New York, third edition.

Purushothaman, D., Hirani, N. S., and Velez-Gallego, C. M. (2009). Scheduling identical parallel batch processing machines to minimise makespan using genetic algorithms. *European Journal of Industrial Engineering*, 3(2):187–206.

Qi, X. T., Bard, J. F., and Yu, G. (2006). Disruption management for machine scheduling: The case of spt schedules. *International Journal of Production Economics*, 103(1):166–184.

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

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Sabuncuoglu, I. and Goren, S. (2009). Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing*, 22(2):138–157.

Sabuncuoglu, I. and Kizilisik, O. B. (2003). Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research*, 41(17):4211–4231.

Salveson, M. E. (1952). On a quantitative method in production planning and scheduling. *Econometrica*, 20(4):554–590.

Samarghandi, H. and ElMekkawy, T. Y. (2011). An efficient hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server. *European Journal of Industrial Engineering*, 5(2):111–131.

Steele, D. C. (1975). The nervous MRP system: how to do battle. *Production and Inventory Management*, 16(4):83–89.

Subramaniam, V., Raheja, A. S., and Reddy, K. R. B. (2005). Reactive repair tool for job shop schedules. *International Journal of Production Research*, 43(1):1–23.

Taillard, E. (1990). Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

Valente, J. M. S. and Schaller, J. E. (2010). Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. *European Journal of Industrial Engineering*, 4(1):99–129.

Vallada, E. and Ruiz, R. (2010). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega-International Journal of Management Science*, 38(1-2):57–67.

Vieira, G. E., Herrmann, J. W., and Lin, E. (2000). Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of Manufacturing Systems*, 19(4):256–266.

Vieira, G. E., Herrmann, J. W., and Lin, E. (2003). Rescheduling manufacturing systems: A

framework of strategies, policies, and methods. *Journal of Scheduling*, 6(1):39–62.

Vollmann, T. E., Berry, W. L., Whybark, D. C., and Jacobs, F. R. (2005). *Manufacturing Planning and Control for Supply Chain Management*. Irwin/McGraw Hill, New York, fifth edition.

Yang, J. and Yu, G. (2002). On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6(1):17–33.

Zandieh, M. and Gholami, M. (2009). An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *International Journal of Production Research*, 47(24):6999–7027.