*"This is an Author's Accepted Manuscript of an article published in the* ***International Journal of Production Research*** *[*Published online: 03 Oct 2013*] [copyright Taylor & Francis], available online at: http://www.tandfonline.com/ [*http://dx.doi.org/10.1080/00207543.2013.835496*]."*

# Dynamic Setup Rules for Hybrid Flowshop Scheduling with Parallel Batching Machines

***Andrea Rossi, Andrea Pandolfi and Michele Lanzetta***
Department of Civil and Industrial Engineering, University of Pisa
Largo Lazzarino, 56122 Pisa, Italy
email: lanzetta@unipi.it; tel.: +39 050 22181.22 / .23; fax: +39 050 22180.65

## Abstract

An *S*-stage hybrid (or flexible) flowshop, with sequence-independent uniform setup times, parallel batching machines with compatible parallel batch families (like in casting or heat treatments in furnaces, chemical or galvanic baths, painting in autoclave etc.) has been analyzed with the purpose of reducing the number of tardy jobs (and the makespan); in Graham's notation: $FPB(m_1, m_2, \dots, m_S)|p\text{-}batch, ST_{si,b}|\Sigma U_i$. Jobs are sorted dynamically (at each new delivery); batches are closed within sliding (or rolling) time windows and processed in parallel by multiple identical machines. Computation experiments have shown the better performance on benchmarks of the two proposed heuristics based on new formulations of the critical ratio ($CR_{setup}$) considering the ratio of allowance setup and processing time in the scheduling horizon, which improves the weighted modified operation due date rule (WMOD).

**Keywords:** Batch Scheduling; Parallel Batches; Dispatching Rules; Critical Ratio, Time Window.

# 1. Introduction

This work considers a non-permutation hybrid (or flexible) flowshop, with sequence-independent uniform setup times, parallel batching machines with compatible parallel batch families.

A flowshop environment is similar to a job shop with unidirectional flow through production stages. Setup times can be sequence-dependent or sequence-independent respectively if jobs and machines or only jobs are considered (Allahverdi et al., 2008). Buffers between stages of the hybrid flowshop environment allow setup and processing of different jobs in parallel and different job sequences on subsequent machines (known as *non-permutation* flowshop scheduling, introduced by Tandon et al., 1991); buffers can be shared in the form of an automatic warehouse or an open space. When buffers between machines are present, non-permutation schedules are likely to outperform their permutation counterparts, where job sequences on each machine remain unchanged. Solutions overcoming the increasing complexity with respect to its permutation counterpart were proposed by Liao et al. (2006), Ying et al. (2010) and Rossi and Lanzetta (2013a, 2013b, 2013c). The impact of buffers, along with machines, products and operations, on reconfiguration and performance evaluation were examined by Colledani and Tolio (2005).

A *hybrid* flowshop is a flowshop with more than one machine in at least one stage. The hybrid flowshop scheduling problem is being extensively studied in manufacturing, logistics, computer sciences etc. with the purpose of assigning jobs to machines to make a better use of resources. A survey of scheduling literature on hybrid flowshop is available from Ribas et al. (2010) and Ruiz and Maroto (2006).

The model briefly recalled below is a two-stage non-permutation hybrid flowshop, with sequence-independent uniform setup times, parallel batching machines and compatible parallel batch families, which is extended to $S$ stages. It has been derived from a real case, the central sterile services of a large hospital, proposed by the authors (Rossi et al., 2013). The purpose is assuring the required quality level, by reducing the number of delayed (or tardy) jobs, virtually to zero, still keeping low staff costs, by reducing the total completion time (or makespan). Shorter schedules mean less idle times, better resources utilization and higher efficiency. Results can be extended to other sterilization plants, continuous casting (steel making), coating (heat and galvanic treatments, painting) etc. Investigating this scheduling problem is also important as it affects the logistics targets with due date reliability/no tardy jobs but also high capacity utilization and low inventory levels.

The layout and some notation for the examined case are represented in Figure 1. A mixed integer linear programming model (MILP) is available in Rossi et al. (2013), along with the related literature.

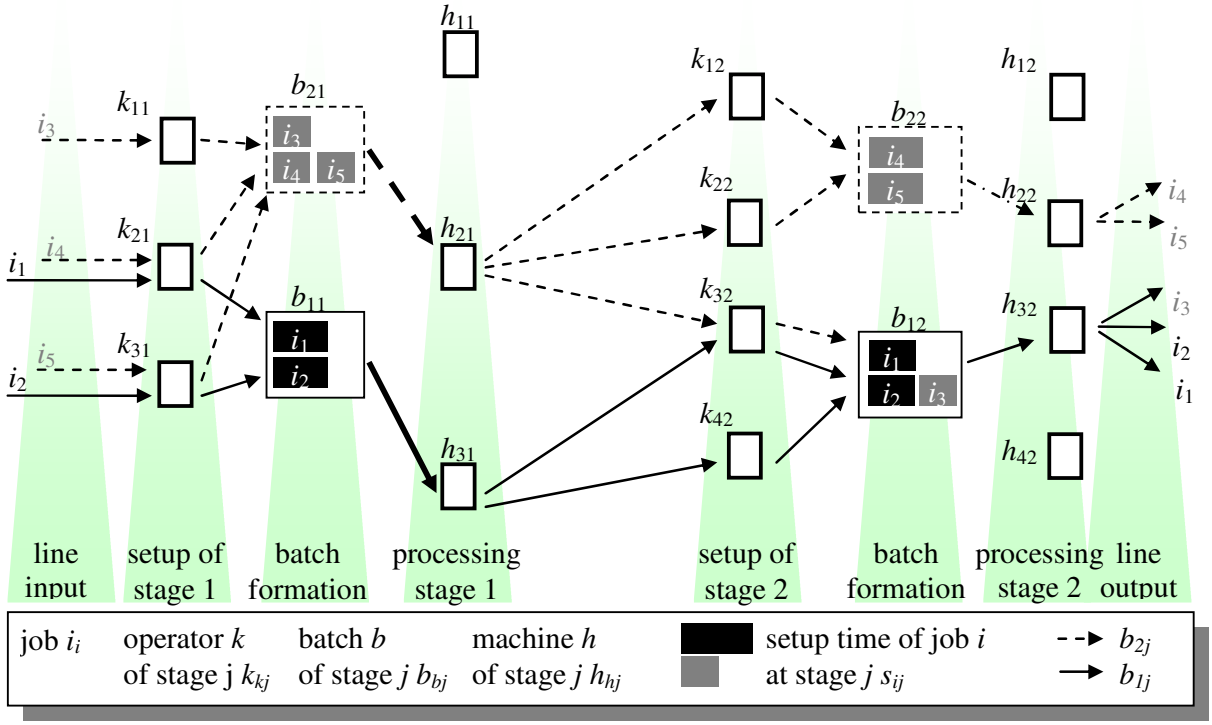


**Figure 1. The examined layout, the *FP2B*(3, 4)|*p-batch*, $ST_{si,b}$|$\Sigma U_i$ model and notation**

Figure 1 about here

The examined flow line includes four stages with manual operations preceding machine operations alternatively. It is modeled as two-stage flowshop by considering manual operations as machine setup. Setup times for each job type are standardized and sequence-independent, consequently the total setup time (and cost) for different schedules remains constant. Also, the total setup time for different job permutations within the same batch is constant. Permutations are possible, because buffers are present in machine loading/unloading (trays and carts). The batch forming time only depends on the current batch to be processed and not on the previous one. Operators work concurrently on the same batch thus affecting the *setup speed*. Setup can be considered *uniform* because jobs processed within different batches are affected by the different setup times of the other jobs belonging to the same batch. All jobs have the same routing through the stages. They are delivered (release date) at prefixed interval of times, and must be processed based on their subsequent planned utilization (due date); both times for each job are known in advance, before scheduling. After setup, jobs are processed by one of the *identical parallel* machines at each stage. The machine time within a stage is the same for all jobs and the same for both stages (70 minutes). The setup time could be added to the processing time. By this alternative two-stage flowshop model

without setup, each job type has a different processing time and processing is interrupted (*preemption*) during batch forming until the machine starts.

Each machine has a finite capacity, i.e. it is able to process one or more (usually in the order of 24-48) jobs of different type simultaneously, in *batches*. Two types of batch productions are possible, *serial batches* and *parallel batches*. In serial batches, jobs of the same batch are processed sequentially, while in parallel batches (current case) they are processed simultaneously, like in casting or heat treatments in furnaces, chemical or galvanic baths, painting in autoclave etc.

Scheduling problems can be described by a triplet $\alpha|\beta|\gamma$ according to the notation of Graham et al. (1979), where field $\alpha$ denotes the system layout and the production flow type (Figure 1), field $\beta$ indicates the operation characteristics and field $\gamma$ denotes the adopted performance indices.

The current problem can be formulated as

$$FPB(m_1, m_2, \dots, m_S)|p\text{-}batch, ST_{si,b}|\Sigma U_i \qquad (1)$$

where, an $S$-stage hybrid flowshop ($F$) with $m_1$, $m_2$, … , $m_S$ identical parallel ($P$) batching ($B$) machines per stage, processes parallel batches (*p-batch*) with sequence-independent setup times ($ST_{si,b}$) in order to minimize the number of tardy jobs $\Sigma U_i$.

The case of only one stage with capacity of all machines equal to one can be reduced to $P||\Sigma U_i$. The case $P||C_{\max}$ is NP-hard according to Garey and Johnson (1979), therefore the $FPB(m_1, m_2, \dots, m_S)|p\text{-}batch, ST_{si,b}|\Sigma U_i$ problem is NP-hard. Finding an optimum in a reasonable time is unlikely, hence heuristics should be employed.

The main contribution of this work is to combine a priority rule and the *sliding* (or *rolling*) time window (or time slot) for application to the batch scheduling problem. Two proposed heuristics sort released jobs by ascending Setup Critical Ratio ($CR_{setup}$), an innovative formulation of the *Critical Ratio* (CR), which involves the setup times of jobs. The priority of jobs is determined dynamically at sliding time windows by evaluating the ratio of allowance setup and processing times in the scheduling horizon.

The proposed heuristics are *dynamic*, because they allow updating the schedule after new jobs enter the system.

### 1.1 Priority rule

In the literature, the dynamic priority has been applied to the sequencing rule of (individual) jobs and to batch forming.

Regarding individual jobs, a number of sequencing rules has been evaluated by Kapanoglu and Alikalfa (2011) and Shafaei and Brunn (1999). Kapanoglu and Alikalfa tested nine priority rules, particulary FIFO, EDD, MST, SOPN, SPT, SRPT, Critical Ratio (CR), COVERT, and Modified Due Date (MDD), in a job shop with time windows dynamic and reconfigurable over time. Dynamic priority rules such as the Critical Ratio and/or COVERT rules work generally better than static rules. The CR rule, originally proposed by Berry and Rao (1975), uses the ratio of the allowance time (time left to the due date) to the remaining processing time to determine the jobs priority. Shafaei and Brunn (1999) state that CR was the second best rule for a cost-based performance considering tardiness and holding costs. Abu-suleiman et al. (2005) introduced a modified CR rule, which yielded better performance than the original CR rule in terms of tardiness and earliness. Pfund et al. (2006) introduced a priority CR (PRCR), a critical ratio that tends to prioritize jobs that have less time available before they become late, but also acknowledges the need to finish tardy jobs that are close to their completion quickly. Chiang and Fu (2007) considered job shop scheduling and extended the concept of Enhanced Critical Ratio (ECR) introducing a degree of urgency based on due date. Chiang and Fu (2009) used a GA scheduler and took the ECR rule as the basis of genome encoding and of the refining mechanism for the local search procedure. According to the proposed dynamic priority rule, the priority value of a job depends on the influence upon all competing jobs caused by first processing it. This rule combines a concept of shorter processing time, earlier due date and longer remaining processing time.

Regarding dynamic priority in batch forming, the review by Perez et al. (2005) provided a detailed classification of papers dealing with deterministic scheduling problems related to batching. Mason et al. (2002) suggested a modification of the Apparent Tardiness Cost index for Sequence-dependent setup times (ATCS, Lee and Pinedo, 1997) to cover batching problems in order to minimize the total weighted tardiness in a hybrid job shop with sequence-dependent setup and ready times (BATCS). Pfund et al. (2008) minimized the total weighted tardiness with ready times on identical parallel machines scheduling with sequence-dependent setup times. They extended the ATCS heuristic by Lee and Pinedo considering a composite dispatching rule (called ATCSR) to allow non-ready jobs to be scheduled on a machine, which necessarily remains idle for a high priority job arriving at a later time. Recently, Pickardt et al. (2013) proposed a two-stage hyper-heuristic that combines a generator of composite rules with a GA to select the best possible combination of common and evolved rules for both the sequencing rule of individual jobs and for batch forming. Except for BATCS, all sequencing rules for batch forming are combined with a batching rule that selects the fullest batch, similarly to the principle of setup avoidance in its focus on maximizing the utilization of machine capacity. They examined a problem taken from a complex

dynamic job shop problem from semiconductor manufacturing. It results that the hyper-heuristic enhances all the common batch forming rules (BATCS, WMDD, PRCR, etc.) and the rules assigned most frequently to critical work centers by the hyper-heuristic is the batch forming rule of the PRCR rule.

Despite the sophisticated formulations of the hyper-heuristic proposed by Pickardt et al., a simple reformulation of the batch forming rule for the priority CR has been selected in this work for easier implementation and interpretation of experimental data; in the proposed heuristics, the novel priority rule has been integrated with a sliding time window technique.

### 1.2 Time window

In real-world situations, it is useful to consider the information on future job arrivals for making decisions in a batch-processing environment. The use of this information can lead to better decisions than those taken by considering only the current system status. Based on this intuitive idea, new heuristics can be devised where batches are formed and sequenced for a pre-specified time window (time slot) using a close-to-optimal procedure. The main objective of shop scheduling by time windows is to assign jobs to available time windows within the scheduling horizon in order to meet due dates. Time windows are used in combination with dispatching rules achieving a dynamic priority over time of jobs, addressed as sliding (or rolling) time windows. As for priority rules, in the literature sliding time windows have been applied to sequencing rules of (individual) jobs and to batch forming.

Regarding individual jobs, Garcia and Lozano (2005) used the time windows approach in a two-stage permutation flowshop with identical parallel machines, where each job would correspond with the production and delivery of each order. Among priority rules tested by Kapanoglu and Alikalfa (2011) in dynamic job shop with sliding time windows, a GA is developed for matching queue length intervals with appropriate priority rules during simulation.

Regarding batch forming, Chand et al. (1997) used time windows for the problem of minimizing total completion time on a single machine with release times. Mönch et al (2005) approached the batch forming problem of minimizing the total weighted tardiness on parallel batching machines (included in a single stage) with incompatible job families, $Pm|p\text{-}batch,incompatible|\Sigma_i w_i T_i$. They proposed a GA, which first assigns jobs to machines and then forms batches on each machine for the jobs assigned to it by a rule similar to BATCS (Mason et al., 2002). Computation experiments with 4-8 batches on 3-5 machines show that on average the GA is about 7% better than the considered dispatching rule used independently; however, the GA is time-consuming ranging between 10 minutes and almost 2 hours compared to the real-time decision of the dispatching rule.

A combination of dispatching rule and GA in which the GA assigns batches to the parallel machines of the single-stage system considered, reduces the computation time and improves the performance of the dispatching rule.

Rossi et al. (2013) considered time windows to assign job to batches and batches to available time windows for each of the two stages of the flowshop in order to minimize tardy jobs and makespan, similarly to Garcia and Lozano (2005), which considered sequencing rules of individual jobs. Rossi et al. considered a case study for problem (1) with $m_1$=3, $m_2$=4, i.e. the *FP2B*(3,4)|*p-batch*, $ST_{si,b}$|($\Sigma U_i$, $C_{\max}$), and developed two heuristics that combine minimum slack time and FCFS rules, based on

- constrained size of parallel batches and
- time windows.

By the first heuristic, a machine (on each of the two stages) starts when the belonging batch reaches a fixed fraction of the machine capacity ($\delta$). The second heuristic is an extension to batch forming of the sequencing rule proposed by Kanet and Li (2004), the weighted modified operation due date rule (WMOD), where different job weights are accounted to reduce the mean weighted tardiness. It considers a time window (a fixed slot of time, $T$), which acts like a system clock for forming batches, independently on the batch size, as shown in the two cases of Figure 2 also considered in current work.

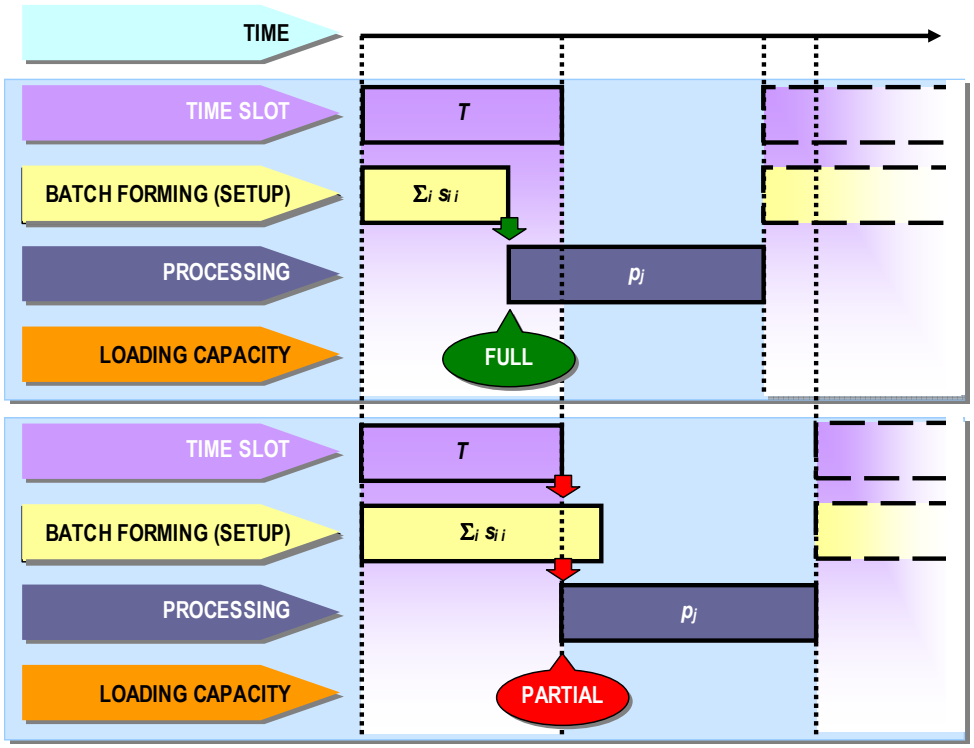


**Figure 2. The time slot (or time window) effect on the batch forming time and machine loading capacity**

Figure 2 about here

Computation experiments were performed on combinations of machines and operator numbers and suggested

- balancing the two stages by allocating operators proportionally to the setup time requirements;
- closing batches before completion, e.g. at fixed intervals of time or before a given machine capacity threshold is reached.

The first conclusion also allows assigning resources (operators on the different stages) and designing a plant layout, not only job scheduling. Both criteria suggest keeping a constant flow of jobs, to prevent accumulation, when all machines are busy, which could delay jobs with short deadline. These criteria are intuitive and results left room for further improvement. The machine capacity threshold was 80% of the total; the fixed interval of time corresponded to the mentioned machine processing time (70 minutes).

As a possible improvement, these two heuristics can be parameterized, by changing the batch-closing interval or the machine capacity threshold. The search of optima for these two heuristics parameters could be carried out

- off-line, however there is no guarantee that they will remain optimal for different manufacturing plans or for different applications;

- on-line, by automatically and exhaustively searching parameters by the scheduling algorithm, thus increasing the scheduling computation time.

As explained below, the proposed heuristics require a single parameter to be set, which is updated automatically, which characterizes the sliding time window.

## 2. The proposed heuristics

The developed scheduling method performs batch forming considering the $CR_{setup}$ rule and a sliding time window technique, applied to the MILP model in Rossi et al. (2013).

**Objective function**

The objective function is minimizing the number of tardy jobs

$$\text{Min} \sum_{i=1}^{N} U_i$$

**Notation**

| | |
|---|---|
| $j$ | stage index, $j$=1,...,$S$ |
| $h$ | machine index, $h$=1,...,$M_j$, where $M_j$ is the total number of parallel machine at stage $j$ |
| $i$, $i'$, $i^*$ | job indices, $i$=1,...,$N$, where $N$ is the total number of jobs |
| $b$ | batch index, $b$=1,...,$B$, where $B$ is the total number of batches |
| $k$ | operator index, $k$=1,.., $v_j$, $v_j$+1,.., $v_{j+1}$,…,$V$ where $V$ is the total number of operators and $v_j$ is the number of operators at stage $j$ (also representing the setup speed at stage $j$) |
| $u_j$ | machine capacity at stage $j$ (i.e. all the machines at stage $j$ have the same capacity) |
| $r_i$ | release date (delivery) of job $i$; also used to update the job available time in the system |
| $d_i$ | due date of the job $i$ |
| $z_i$ | size of job $i$ |
| $s_{i\,j}$ | setup time of job $i$ at stage $j$ |
| $p_j$ | processing time of parallel machines at stage $j$ |
| $C_{i\,j}$ | completion time of job $i$ at stage $j$ |
| $SC_{j\,i\,k}$ | completion time of operator $k$ for the setup of job $i$ at stage $j$ |
| $BC_{j\,b\,h}$ | completion time of batch $b$ of machine $h$ at stage $j$ |
| $t$ | current time |

| | |
|---|---|
| $t^*$ | closing time of batch |
| $T$ | sliding time window span |
| $L^w$ | $w^{\text{th}}$ queue of jobs included in the sliding time window, sorted by the $CR_{setup}$ rule |
| $CR_{setup}$ | Setup Critical Ratio defined in expressions (2) and (7) below. |

**Decision variable**

| | |
|---|---|
| $Z_{i\,j\,b\,h}$ | 1, if job $i$ is assigned to batch $b$ of machine $h$ at stage $j$ |
| | 0, otherwise |

**$CR_{setup}$ rule**

The proposed priority rule is based on the setup critical ratio, $CR_{setup}$, which considers setup time and due date of each job. This rule allows selecting jobs according to the ratio between the setup time of the current operation and its deadline by the expression

$$CR_{setup} = \frac{\max\{d_i - t, 0\}}{s_{i\,j} + t_{i\,j}},\ i \in L,\ j=1,..,S \qquad (2)$$

The $CR_{setup}$ rule considers the time elapsed from the earliest to the latest setup time of jobs ready to be processed. Jobs with lower $CR_{setup}$, corresponding to closer due date $d_i$ and higher setup time $s_{i\,j}$, are scheduled first at each stage $j$. Below, a more sophisticated $CR_{setup}$ rule that considers the allowance setup times of the scheduling horizon (all the release and due dates are known a priori) is also proposed.

If a new job enters the system, the whole schedule can be changed by applying the $CR_{setup}$ rule on all the jobs except those already being processed by an operator. This makes the algorithm dynamic. The proposed implementation is reported in concise form for readability in ***Listing 1***. The same steps are reported in bold and further detailed in ***Listing 2*** and ***Listing 3*** for completeness.

***Listing 1: The proposed heuristic (concise)***

**Select a sliding time window by the lowest ready time**

**Stage Loop ($j$=1,…,$S$)**

1. **Sort jobs of the sliding time window by increasing $CR_{setup}$ and update the sliding time window**
2. **Open a batch for each machine $h$ of stage $j$**

3. **Assign jobs to operators**
    - 3.1. **Sort jobs included in the sliding time window by $CR_{setup}$**
    - 3.2. **Assign the highest priority job to the available operator**
    - 3.3. **Update the sliding time window with the lowest setup time**
4. **Assign jobs to batches**
    - 4.1. **Sort jobs included in the sliding time window by $CR_{setup}$**
    - 4.2. **Assign the highest priority job to the available batch until the machine capacity is reached**
    - 4.3. **Machine capacity reached ?: close the batch and open a new batch for the next run**
    - 4.4. **Update the sliding time window with the maximum between the next time window parameter and the lowest batch processing time**
5. **Performance index evaluation**, $\sum_{i=1}^{N} U_i$ (and $C_{max}$)

As the scheduling horizon is known, the proposed heuristics schedule batches stage by stage. For each stage, jobs whose release date is included in the sliding time window $\left[\min_{i=1,\ldots,N}\{r_i, t^*\}, t^*\right]$ are sorted by increasing $CR_{setup}$ and assigned to the first available operator. If no release date of jobs is included in the sliding time window, i.e. $t < \min_{i=1,..,N} r_i$, *all* the jobs are sorted by the $CR_{setup}$. Hence, the sliding time window is updated to the lowest setup completion time

$$\min_{i=1,\ldots,N,\, i^*=v_{j-1}+1,\ldots,v_j}\{r_i, SC_{ji^*k}\} \qquad (3)$$

where $i^*$ is the job assigned for setup to operator $k$ of stage $j$.

Initially, for each machine of the stage, a batch is opened (to allow forming a lot of jobs that are processed together, i.e. a parallel batch). Next, the two steps of job assignment to, respectively, operators (in list $L^1$) and batches (in list $L^2$), are performed. Both steps include the same three phases (detailed in ***Listing 3***) of: i) sort by increasing $CR_{setup}$ the jobs included in the time window, ii) assign the highest priority job to, respectively, the available operator or the available batch

(machine) and iii) update the sliding time window with, respectively, the lowest setup or batch processing time. Moreover, in the case of batch assignment, the heuristics check the machine capacity, with expression

$$z_{i^*} \cdot Z_{i^* j\,(b+h^*)\,h^*} + \sum_{i=1}^{N} z_i \cdot Z_{i\;j\,(b+h^*)\,h^*} \leq \delta \cdot u_j \qquad (4)$$

with $\delta = 1$, whereas in Rossi et al. (2013) this was an additional algorithm parameter to be set. If the batch size exceeds the machine capacity, the batch is closed (and the machine starts); the end time of the sliding time window is updated considering the time window parameter, $T$, by the expression

$$t = \min\left\{\min_{h=1,..,M_j}\{BC_{jbh}\}, r_{i^*} + T\right\} \qquad (5)$$

Expression (5) slides the time window parameter from the previous time to a new time ($r_{i^*}$+$T$). If a machine completes within this time window, the end time of the time window rolls to the machine completion time and hence a new batch can be formed Figure 2.

***Listing 2: The proposed heuristic (complete)***

**Initialization**: set $t$=0, $b$=0, $v_0$=0, $C_{i\,0} = r_i$ for all jobs, close_batch$_h$=0 for each machine $h$ = 1,.., $M_j$; set the time window parameter ($T$). Set all the binary digits $Z_{i\,j\,b\,h'}$ and all the completion setup and batch processing times to zero.

**Stage loop** ($j$=1,…,$S$)

1. **Assign jobs to operators**
   - **1.1. Sort jobs of the sliding time window (list $L^1$) by increasing CR$_{setup}$ and update the sliding time window**
   - **1.2. Assign job to available operator**
     - **1.2.1.** Assign job $i^*$ to the first available operator $k^* \in \{v_{j-1}+1,.., v_j\}$
     - **1.2.2.** Update the current completion time $C_{j\,i^*\,k^*}$ of job $i^*$ increasing the available time of operator $k^*$: $r_{i^*} \leftarrow \max\{C_{i^*0}, SC_{j\,i'\,k^*}\} + s_{i^*\,j}$, $SC_{j\,i^*\,k^*} \leftarrow r_{i^*,}$ where $i'$ is the previous job processed by operator $k^*$

**1.2.3.** Insert job $i^*$ in list $L^2$ and remove job $i^*$ from $L^1$

**1.3. Update the sliding time window with the lowest setup time**

**1.3.1.** If $L^1$ is empty go to **Assign Job to Batches**, else

$t \leftarrow \min_{k=v_{j-1}+1,..,v_j}(SC_{ji^*k})$ and go to **Sort jobs of the sliding time window ($L^1$) by increasing $CR_{setup}$ and update the sliding time window**

**2. Assign jobs to batches**

**2.1. Sort jobs of the sliding time window ($L^2$) by increasing $CR_{setup}$ and update the sliding time window**

**2.2. Select a machine with an open batch**

2.2.1. If the set $M = \{h \mid \text{close\_batch}_h = 0\}$ is not empty

2.2.1.1. Select a machine $h^* \in M$

2.2.1.2. **Sort jobs of the sliding time window ($L^2$) by increasing $CR_{setup}$ and update the sliding time window**

2.2.1.3. Apply the close batch rule for machine $h^*$

2.2.1.3.1. If job $i^*$ can be included in machine $h^*$ according to expression (4)

2.2.1.3.1.1. $Z_{i^* j\,(b+h^*)\,h^*} \leftarrow 1$

2.2.1.3.1.2. remove job $i^*$ from $L^2$

2.2.1.3.2. Else (i.e. job $i^*$ exceeds the size of machine $h^*$)

2.2.1.3.2.1. $\text{close\_batch}_h \leftarrow 1$

2.2.1.3.2.2. evaluate $C_{ij} \leftarrow C_{ij} + p_j$ and $BC_{j\,b\,h} = C_{ij}$ $\forall i = 1,..,N \mid Z_{i\,j\,(b+h^*)\,h^*} = 1$

2.2.1.3.2.3. go to **Sort jobs of the sliding time window ($L^2$) by increasing $CR_{setup}$ and update the sliding time window** (at step 2.i)

2.2.2. Else synchronize the new current time, with time window parameter and first completed batch: $t = \max\left\{\min_{h=1,..,M_j}\{BC_{jbh}\}, r_{i^*} + T\right\}$

2.3. If $L^2$ is empty

2.3.1. If $j=S$ go to **Performance index evaluation**

2.3.2. Else If $j=1$ go to **Stage loop** ($j=1,\ldots,S$)

2.4. Else $b \leftarrow b+ M_j$ and go to **Assign jobs to batches**

**Performance index evaluation**

$$\sum_{I=1}^{N} U_i = \{|i| \mid C_{i\,S} - d_i > 0\} \text{ and } C_{max} = \max_{i=1,..,N} C_{i\,S}$$

Two heuristics, $H^{❶}$ and $H^{❷}$, are proposed in order to test the impact of different batch forming criteria on performance targets according two different expressions of the $CR_{setup}$ rule. Heuristic $H^{❶}$ considers stages separately as already defined by expression (2)

$$CR_{setup}(H^{❶}) = CR_{setup} \qquad (6)$$

evaluated at each stage individually.

Heuristic $H^{❷}$ evaluates the critical ratio by the expression

$$CR_{setup}(H^{❷}) = \frac{\max\{d_i - t, 0\}}{\sum_{j=j^*}^{S}\left(s_{i\,j} + t_{i\,j}\right)},\ i=1,\ldots N,\ j=j^*,..,S \qquad (7)$$

which takes into account the allowance setup times of the *remaining* scheduling horizon (all the release and due dates are known a priori), i.e. the total job setup time over all the remaining stages.

The main difference between heuristic $H^{❶}$ and $H^{❷}$ is the (jobs) precedence criterion. The former considers stages separately, while the latter considers all the setup times in advance.

According to these two heuristics, batches are *closed* when full capacity is reached or at dynamic sliding times window also without reaching the full capacity.

To achieve better schedules, the machine idle time needs to be minimized and can be achieved by starting a machine as soon a batch is closed (as described below), i.e. *non-delay batch schedule* is considered. Nevertheless, non-delay schedule not necessarily implies optima schedules.

These heuristics do not include a capacity threshold limit $\delta$ to close a batch as in the authors' previous work, where it was fixed at 80%; one parameter less, $\delta = 1$, and only one to set, the sliding time window $T$, reduces the number of computation experiments and makes the application of the proposed heuristics to real cases straightforward; the machine capacity is automatically adjusted by the limited waiting time (time window) before the batch close and the machine start.
The differentiation of the two heuristics according to expressions (2) and (7) and the integration with the sliding time window is detailed in ***Listing 3***. The priority lists $L^1$ for the assignment of jobs to operators and $L^2$ for the assignment of jobs to batches are parameterized in ***Listing 3*** as $L^w$.

***Listing 3:*** **Sort jobs of the sliding time window ($L^w$) by the $CR_{setup}$ rule and update the sliding time window**

1. **Sort all the jobs in list $L^w$ by $CR_{setup}$**
   - **1.1.** Sort jobs in a list $L^w$ according to decreasing $\max\{r_i - t, 0\}$, $i=1,..,N$ and, in case of ties, arrange jobs in decreasing order according to $CR_{setup}$ by expression (2) for $H^{❶}$ and by expression (7) for $H^{❷}$
2. **Update the sliding time window:** if $t \le r_{i*}$ $t \leftarrow r_{i*}$
3. **Select the highest priority job $i^*$ from $L^w$**

A pictorial view of the different cases available for combinations of release and due date of jobs is shown in Figure 3 for the simpler heuristic $H^{❶}$ for clarity.

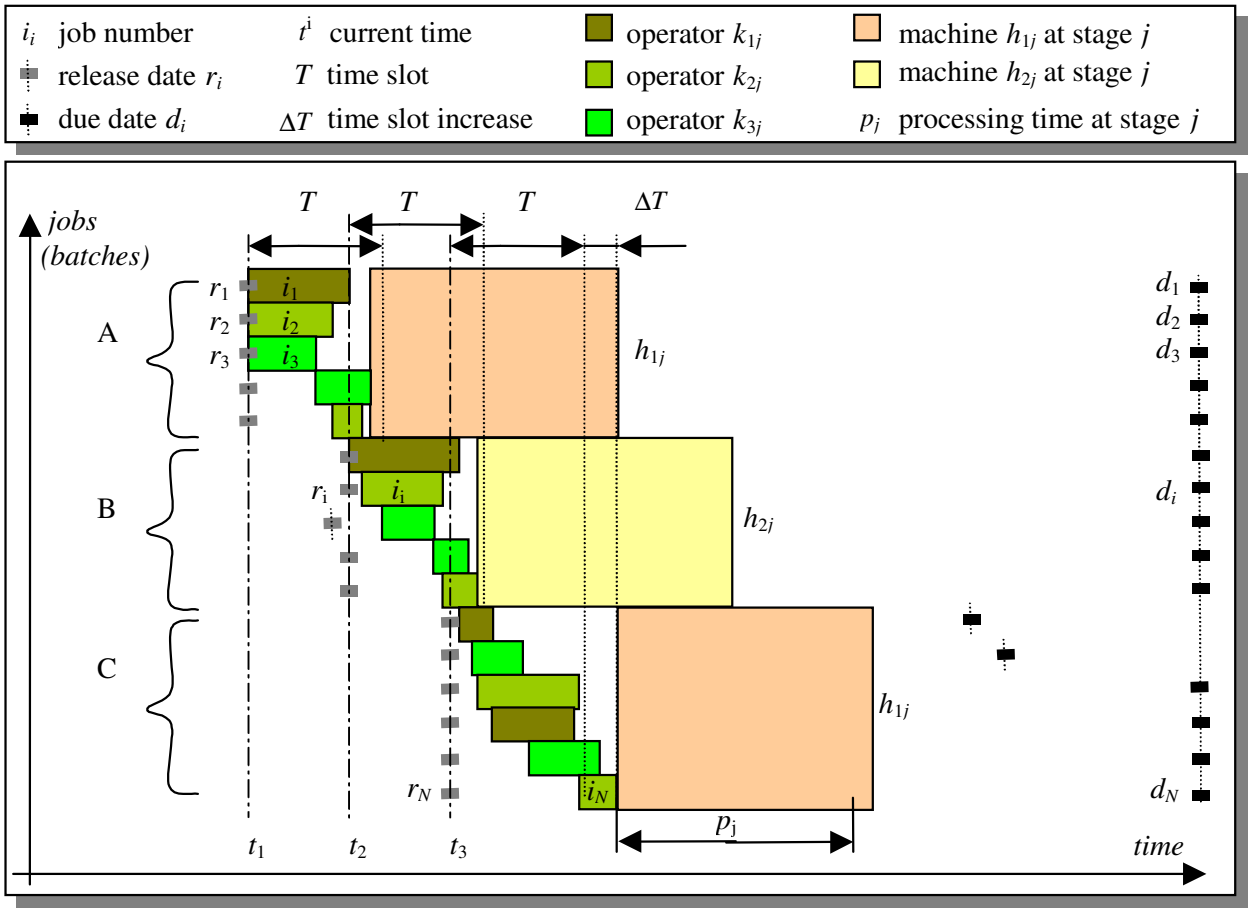


**Figure 3. The proposed heuristic for the formation of batches A, B and C: dynamic priority rule and time window selection**

Figure 3 about here

The following cases are possible:

- batch A: at instant $t_1$, jobs $i_1$, $i_2$ and $i_3$ included in batch A have identical release ($r_1$=$r_2$=$r_3$) and due dates ($d_1$=$d_2$=$d_3$). According to heuristic H$^{\bullet}$, jobs with higher setup time are processed first;
- batch B: jobs included in batch B have identical due dates ($d_{i-1}$=$d_i$=$d_{i+1}$) and one job has earlier release date ($r_{i-1}$=$r_i$>$r_{i+1}$). At current time $t_2$, all jobs are available and hence their release dates are not considered by heuristic H$^{\bullet}$.
- batch C: jobs included in batch C have identical release dates. At current time $t_3$, all jobs are available and hence according to $CR_{setup}$, jobs with lower due dates, i.e. $d_i - t$ in expression (2), are processed first.

Figure 3 also shows a typical behavior of the batch formation mechanism. Batches A and B complete exactly within the interval of time $T$, respectively at times ($t_1$+$T$) and ($t_2$+$T$). On the opposite, batch C is not closed within the sliding time window $T$, because all the machines are busy.

In this case the sliding time window is expanded by $\Delta T$, which corresponds to the time before a machine is available again (i.e. $t_2+T+\Delta T=\min_h\{C_{j\,b\,h}\}$).

A sliding time window has the purpose of reducing the operator idle time, based on the machines availability:

- if there is at least a machine waiting, the time window is kept constant, whilst
- if all machines are busy, the sliding time window is increased.

The performance of the two formulations of $CR_{setup}$ has been assessed by computation experiments.

## 3. Computation experiments

The performance of the proposed heuristics, implemented with Visual Basic for Applications in Microsoft Excel 2007, has been assessed with the heuristic parameters in Table 1 on the six benchmark problems by Rossi et al. (2013) generated with the parameters in Table 5 below.

**Table 1. Heuristic parameters**

| Parameter | Symbol | Proposed | Selected |
|---|---|---|---|
| Sliding time window span | $T$ | $7 \cdot \text{Uniform}[1,8]$ | 19 for H❶<br>18 for H❷ |
| Heuristic H❶ | $CR_{setup}(H❶)$ | $\frac{\max\{d_i - t,0\}}{s_{i\,j}+t_{i\,j}}$<br>$i\in L, j=1,..,S$ | no |
| Heuristic H❷ | $CR_{setup}(H❷)$ | $\frac{\max\{d_i - t,0\}}{\sum_{j=j^*}^{S}\left(s_{i\,j}+t_{i\,j}\right)}$<br>$i=1,\ldots N, j=j^*,..,S$ | yes |

In the cited paper, a lower bound (recalled in Table 3) is given for each problem for tardy jobs and for makespan, consequently the total completion time is also considered in experiments. The results of the proposed heuristics are compared with the best of the four heuristics proposed by Rossi et al. in all possible layout configurations of operators assignment ($v_1$, $v_2$), with a total of six and seven operators ($V=\{6,7\}$) and minimum two operators on the two stages ($v_1=(V\text{-}i,\ i=2,..,V\text{-}2)$, $v_2=V\text{-}v_1$).

### 3.1 Optimal time window

A preliminary set of tests has been carried out to evaluate the basic algorithm parameter $T$ providing the minimum makespan. In particular, a *Set* of suitable $T$ values ($Set=\{7 \cdot l, l=1,..,8\}$) has been used for each layout configuration. The operators' assignment to the two stages has been exhaustively tested, considering their low number (seven combinations).

In conclusion, each of the two heuristics has run 48 times (considering the six benchmark problems for each occurrence of $T$), totaling 672 runs considering both heuristics and the combinations of ($v_1$, $v_2$).

Table 2 shows the values of the optimal time windows determined by the heuristics $H^{❶}$ and $H^{❷}$ with operators' assignment ($v_1$, $v_2$). The main statistical parameters (median, mean, mode and standard deviation) for the optimal makespan occur when the time window parameter ranges in [7, 28] minutes. From this preliminary analysis the average values for $T$ are selected, particularly $T$=19 for $H^{❶}$ and $T$=18 for $H^{❷}$.

**Table 2. Exhaustive evaluation of time windows $T$ from the $Set=\{7 \cdot l, l=1,..,8\}$ providing the minimum makespan with heuristics $H^{❶}$ and $H^{❷}$ (last two columns) with operators assignment, ($v_1$, $v_2$), where $v_1=(V-i, i=2,..,V-2)$, $v_2=V-v_1$ and $V=\{6, 7\}$**

| *instance no.* | *benchmark no.* | *operators no. (stage 1)* $v_1$ | *operators no. (stage 2)* $v_2$ | *best T heuristic* $H^{❶}$ | *best T heuristic* $H^{❷}$ |
|---|---|---|---|---|---|
| *1* | *1* | 5 | 2 | 21 | 21 |
| *2* | *2* | 5 | 2 | 21 | 21 |
| *3* | *3* | 5 | 2 | 7 | 7 |
| *4* | *4* | 5 | 2 | 7 | 14 |
| *5* | *5* | 5 | 2 | 14 | 21 |
| *6* | *6* | 5 | 2 | 14 | 7 |
| *7* | *1* | 4 | 3 | 7 | 21 |
| *8* | *2* | 4 | 3 | 21 | 28 |
| *9* | *3* | 4 | 3 | 7 | 35 |
| *10* | *4* | 4 | 3 | 21 | 14 |
| *11* | *5* | 4 | 3 | 14 | 14 |
| *12* | *6* | 4 | 3 | 21 | 7 |
| *13* | *1* | 3 | 4 | 14 | 21 |
| *14* | *2* | 3 | 4 | 21 | 35 |
| *15* | *3* | 3 | 4 | 49 | 28 |
| *16* | *4* | 3 | 4 | 21 | 7 |
| *17* | *5* | 3 | 4 | 7 | 14 |

| | | | | | |
|---|---|---|---|---|---|
| *18* | *6* | 3 | 4 | 7 | 14 |
| *19* | *1* | 2 | 5 | 21 | 14 |
| *20* | *2* | 2 | 5 | 28 | 35 |
| *21* | *3* | 2 | 5 | 21 | 28 |
| *22* | *4* | 2 | 5 | 7 | 28 |
| *23* | *5* | 2 | 5 | 35 | 14 |
| *24* | *6* | 2 | 5 | 14 | 7 |
| *25* | *1* | 4 | 2 | 21 | 21 |
| *26* | *2* | 4 | 2 | 7 | 7 |
| *27* | *3* | 4 | 2 | 7 | 7 |
| *28* | *4* | 4 | 2 | 14 | 7 |
| *29* | *5* | 4 | 2 | 21 | 21 |
| *30* | *6* | 4 | 2 | 21 | 14 |
| *31* | *1* | 3 | 3 | 7 | 7 |
| *32* | *2* | 3 | 3 | 35 | 7 |
| *33* | *3* | 3 | 3 | 21 | 21 |
| *34* | *4* | 3 | 3 | 21 | 21 |
| *35* | *5* | 3 | 3 | 28 | 14 |
| *36* | *6* | 3 | 3 | 21 | 14 |
| *37* | *1* | 2 | 4 | 21 | 21 |
| *38* | *2* | 2 | 4 | 7 | 7 |
| *39* | *3* | 2 | 4 | 49 | 28 |
| *40* | *4* | 2 | 4 | 21 | 35 |
| *41* | *5* | 2 | 4 | 14 | 35 |
| *42* | *6* | 2 | 4 | 49 | 7 |
| | ***Median*** | | | 21 | 14 |
| | ***Mode*** | | | 21 | 7 |
| | ***Mean*** | | | 19 | 18 |
| | ***Standard deviation*** | | | 11 | 9 |

From the statistics in Table 2, it can be noticed that the best sliding time window values with heuristic $H^{❷}$ are slightly lower than with $H^{❶}$. This shows that $CR_{setup}(H^{❷})$ produces a higher fragmentation of batches.

The computation time for each run is in the order of 4-6 minutes, consequently the optimal time window parameter *T* can be updated often to take into account changing production plans. The computation time is significantly lower than manual scheduling in the original hospital case. This computation time is lower than 0.5% of the considered horizon (in the original time unit: minutes). This processing time can be easily reduced by at least one order of magnitude by software engineering and new processors, considering that high-level programming language has been used.

# 4. Results

## 4.1 Operators' assignment

The seven configurations of operators' assignment are listed in Table 3 along with the two performance indices calculated (the number of tardy jobs and the makespan) and are compared on benchmark no. 1 (worst case) from Rossi et al. (2013). The grayed rows in Table 3 list the assignment of $V$=7 operators, at stages 1 (2): $v_1$=2 (5), 3 (4), 4 (3) and 5 (2). Similarly in the white rows for one operator less: $V$=6 and $v_1$={2, 3, 4}.

The configuration parameters can be expressed in compact form by the tern $H^p|V^q|v_1{}^r$ with the quotes of p={1,2}, q={6,7} and r={2,..,$V$-2}. The short version $H^p||$ indicates each set of seven configurations where the first two columns take the values H=p.

Results have been evaluated for the two heuristics by comparison with the LB according to

$$\%\text{gap}_{H\text{-}LB} = [C_{\max}(H^p) - \text{LB}] / \text{LB} \qquad (8)$$

Analogously the relative distance between the results of different heuristics versus Rossi et al. (2013) is evaluated with reference to the LB, e.g.

$$\%\text{gap}_{RPL\text{-}H} = [C_{\max}(\text{RPL}) - C_{\max}(H^p)] / \text{LB} \qquad (9)$$

where $C_{\max}$(RPL) is the best heuristic and operator configuration from Rossi et al. (2013).

**Table 3. Benchmark no. 1 (worst case), the lower bound (LB) is 624.3 minutes: result of simulations for different operators' assignment for heuristics $H^{❶}$ and $H^{❷}$ with selected time windows, respectively, of $T$=19 and $T$=18. In bold new upper bounds**

| | Operators | | | Rossi et al. (2013) | | Proposed heuristic $H^{❶}$ | | Proposed heuristic $H^{❷}$ | | $\%gap_{RPL\text{-}H}$ INSERIRE (9) | | $\%gap_{H\text{-}LB}$ INSERIRE (8) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Case no.** | $V$ | $v_1$ | $v_2$ | $\sum U_i$ | $C_{max}$ | $\sum U_i$ | $C_{max}$ | $\sum U_i$ | $C_{max}$ | $H^{❶}$ | $H^{❷}$ | $H^{❶}$ | $H^{❷}$ |
| ***1*** | 7 | 5 | 2 | 3 | 867 | **0** | 785 | **0** | **784** | 13.1 | 13.3 | 25.7 | 25.6 |
| ***2*** | 7 | 4 | 3 | 0 | 745 | 0 | **689** | 0 | 700 | 9.0 | 7.2 | 10.4 | 12.1 |
| ***3*** | 7 | 3 | 4 | 0 | 690 | 0 | 667 | 0 | **666** | 3.7 | 3.8 | 6.8 | 6.7 |
| ***4*** | 7 | 2 | 5 | 0 | 676 | 0 | **659** | 0 | 661 | 2.7 | 2.4 | 5.6 | 5.9 |
| ***5*** | 6 | 4 | 2 | 3 | 867 | **0** | **789** | **0** | 806 | 12.5 | 9.8 | 26.4 | 29.1 |
| ***6*** | 6 | 3 | 3 | 6 | 749 | **0** | 695 | **0** | **694** | 8.6 | 8.8 | 11.3 | 11.2 |
| ***7*** | 6 | 2 | 4 | 6 | 700 | **0** | 678 | **0** | **672** | 3.5 | 4.5 | 8.6 | 7.6 |

Both heuristics $H^{❶}$ and $H^{❷}$ reach the optimum schedule with 0 tardy jobs for all instances.

This optima performance was not reached by Rossi et al. in some configurations (particularly with the minimum number of operators, $V$=6), with up to six tardy jobs.

The best makespan for each heuristic are respectively 659 and 661 minutes with 7 operators (case $|V^7|v_1{}^2$ no. 4) and 678 and 672 with 6 operators (case $|V^6|v_1{}^2$ no. 7). With both $H^{❶}$ and $H^{❷}$, the worst results are achieved when the assignment of operators between the two stages is strongly unbalanced inversely to the respective setup speeds (case $|V^7|v_1{}^5$ no. 1), particularly with one operator less (case $|V^6|v_1{}^4$ no. 5) with a makespan of respectively 789 and 806 for the two heuristics.

The makespan $C_{max}$ ranges from 659 (case $H^{❶}|V^7|v_1{}^2$ no. 4) to 806 minutes (case $H^{❷}|V^6|v_1{}^5$ no. 5), with respect to the range [676.2, 889.8] achieved by the competitor heuristics. This also shows that a wrong selection of the total number of operators $V$ and assignment on the two stages $v_i$ causes a total batch processing time increase of 147 minutes (+22%).

A wrong operator assignment may not produce an increase of the number of tardy jobs but it clearly produces a makespan increase.

The minimum makespan with the new heuristic case $H^{❶}|V^7|v_1{}^2$ no. 4 is only +5.6% above the lower bound with respect to +8.3% of the competitor heuristics.

As for the operators' assignment, a wrong assignment ($H^{❶}|V^7|v_1{}^5$ no. 1 vs. case $H^{❶}|V^7|v_1{}^2$ no. 4) produces a makespan increase of 126 minutes (+20.1%) within the same heuristic. Also within heuristic $H^{❷}$ ($H^{❷}|V^7|v_1{}^5$ no. 1 vs. case $H^{❷}|V^7|v_1{}^2$ no. 4) a wrong assignment provides an increase of 123 minutes (+19.7%). A reduction of one operator (case $V^7|v_1{}^2$ no. 4 vs. $V^6|v_1{}^2$ no. 7) produces no

effect on the number of tardy jobs and a slight makespan increase: 19 minutes (+3.0%) for $H^{❶}$ and 11 minutes (+1.8%) for $H^{❷}$. This performance decrease is most probably tolerable compared to the relevant economic impact of reducing the operator number.

The case of one machine less ($m_2$=3 vs. $m_2$=4) and one operator less in the latter stage ($|V^6|v_1{}^2$) has also been simulated. Also machines seem redundant, considering the slight makespan increase of 19 minutes (+3.0% above 659 minutes of case $|V^7|v_1{}^2$ no. 4) for $H^{❶}$ and of 11 minutes (+1.8% above 661 minutes of case $|V^7|v_1{}^2$ no. 4) for $H^{❷}$.

### 4.2 Validation

In the first set of tests, the preferred layout configuration is $|V^6|v_1{}^2$, because it involves fewer operators as anticipated, and they are assigned to the two stages inversely to the respective setup speeds.

The two proposed heuristics have been tested in this optimal layout configuration on five benchmark problems proposed by Rossi et al. (2013), which were obtained by randomization of real data for validation purposes.

Table 4 shows the results achieved by heuristics $H^{❶}$ and $H^{❷}$ with time windows parameter $T$=19 minutes for $H^{❶}$ and $T$=18 minutes for $H^{❷}$ on benchmarks no. 2 to no. 6.

The performance of the best heuristic and operator configuration from Rossi et al. (2013) is evaluated as

$$\%\text{gap}_{RPL\text{-}LB} = [C_{\max}(\text{RPL}) - \text{LB}] / \text{LB} \qquad (10)$$

**Table 4. Benchmarks no. 2 to no. 6 (generated with the parameters in Table 5 below). Tardy jobs and makespan and respective lower bounds (LB) from Rossi et al. (2013) and the best results (case $|V^6|v_1^2$ and their best heuristic). In bold new upper bounds**

| | | Rossi et al. (2013) | | | | $H^{❶}$ | | | $H^{❷}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Benckmark no.** | **LB for** $\sum_{i=1}^{N} U_i$ | **LB for** $C_{\max}$ | $\sum_{i=1}^{N} U_i$ | $C_{\max}$ | **%gap$_{RPL}$ (10)** | $\sum_{i=1}^{N} U_i$ | $C_{\max}$ | **%gap$_{H\text{-}LB}$ (8)** | $\sum_{i=1}^{N} U_i$ | $C_{\max}$ | **%gap$_{H\text{-}LB}$ (8)** |
| 1 | 0 | 624.3 | 6 | 700 | 12.1 | **0** | 678 | 8.6 | **0** | **672** | 7.6 |
| 2 | 7 | 620.6 | 7 | 689 | 11.0 | **7** | **685** | 10.4 | **7** | 689 | 11.0 |
| 3 | 6 | 633.5 | 7 | 709 | 11.9 | **6** | 694 | 9.6 | **6** | **676** | 6.7 |
| 4 | 6 | 607.9 | 6 | 671 | 10.4 | **6** | 673 | 10.7 | **6** | **662** | 8.9 |
| 5 | 5 | 629.0 | 6 | 719 | 14.3 | **5** | 723 | 14.9 | **5** | **684** | 8.7 |
| 6 | 4 | 633.5 | 5 | 676 | 6.7 | **4** | 673 | 6.2 | **4** | **662** | 4.5 |

The number of tardy jobs is minimum, equal to the lower bound, for both the heuristics and for all the benchmark problems, whereas Rossi et al. (2013) reached no tardy jobs only in two benchmarks (no. 2 and no. 4).

Heuristic $H^{②}$ based on the new expression (7) of the critical ratio performs better than $H^{①}$ in five out of six benchmark problems. The $\%gap_{H\text{-}LB}$ reduction of $H^{②}$ with respect to $H^{①}$ ranges between 1.0% and 6.2% except in benchmark no. 2. The $\%gap_{H\text{-}LB}$ of the best heuristic $H^{②}$ is 4.5% allowing a further makespan reduction by future work. Both proposed heuristics perform better than the best heuristics from Rossi et al. (2013) in all 7 operator configurations for benchmark no. 1 and on the 5 random benchmarks no. 2 to no. 6, except for $H^{①}$ in benchmark no. 3 and no. 4. The total number of tardy jobs for all six benchmarks in Table 4 has been reduced by 32% with respect to the competitor heuristics. Additionally, the $\%gap_{RPL\text{-}H}$ reduction of $H^{②}$ for all six benchmarks in Table 4 is 3.2%.

## 5. Discussion

The two performance indices (number of tardy jobs and makespan) versus operator number, assignment and time window have been assessed. Tardy jobs are the dominating criterion (efficacy) for the examined problem. The makespan is an efficiency criterion. The minimum $C_{\max}$ achieved in simulations is less than half of the observation period (24 hours).

From preliminary computation experiments, an optimal time window parameter $T$ between 7 and 28 minutes, further restricted to 18-19 minutes has been found, as opposed to 70 minutes of previous work. By such low time allowed for setup, machines start operation more frequently and consequently with smaller batches. For this reason, there has been no need to test the system with an additional parameter to limit the machine capacity as in previous work.

Computation experiments with the two new proposed heuristics on the six benchmarks have shown similar or better performance in all cases with respect to the best result among the best of the four heuristics in Rossi et al. (2013). Both the machine capacity threshold and the time window parameter have been explored here, yielding lower optimal values than those that were fixed in Rossi et al. The better performance seems a consequence of the increased batch fragmentation by the lower time window. Batch fragmentation (machine loaded with fewer jobs and starting more frequently) is possible by the excess of machine capacity with respect to the number of jobs to be processed. The excessive machine capacity has also been shown by the slightly worst performance achieved with one machine less in the second stage.

The optimal time window and capacity provided are not absolute values and will probably vary on a case basis; however these two parameters are automatically set by the $CR_{setup}$ rule.

Heuristic $H^{❷}$ based on the new proposed expression (7) of critical ratio, which compares the available time with the total remaining processing (setup) time is better than $H^{❶}$, which only considers the processing time on each stage individually, as in the authors' previous work.

A difference in performance of the two proposed heuristics and regarding tardy jobs and makespan has been noticed between the first benchmark obtained from a real case (Table 3) and the five benchmarks obtained by randomization (Table 4).

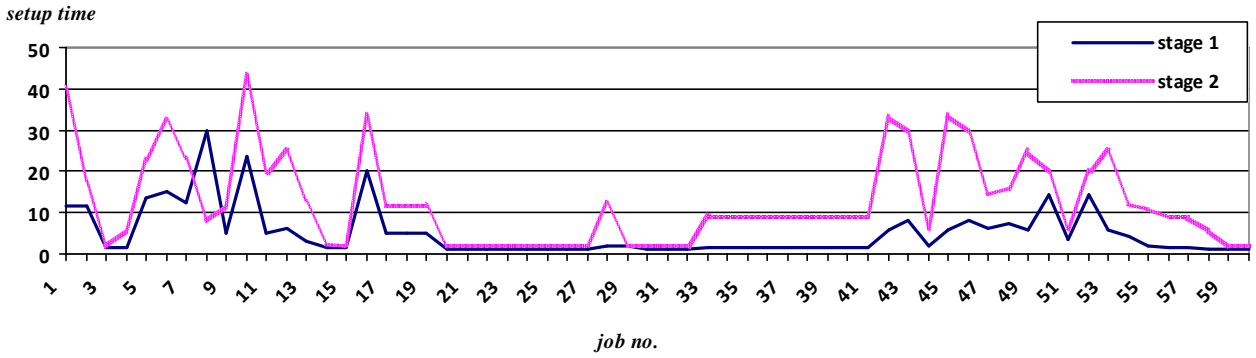


**Figure 4. Benchmark no. 1 (worst case), distributions of setup times on stage 1 and on stage 2 are *similar***

Figure 4 about here

From Table 3, with benchmark no. 1, heuristic $H^{❶}$ does not differ significantly from $H^{❷}$ in the case of makespan minimization. Figure 4 shows that for benchmark no. 1, the setup times have a similar distribution between stage 1 and stage 2. Consequently, the adopted critical ratio $CR_{setup}$ calculated respectively with expression (2) and (7) produces the effect of sorting jobs in a similar order.

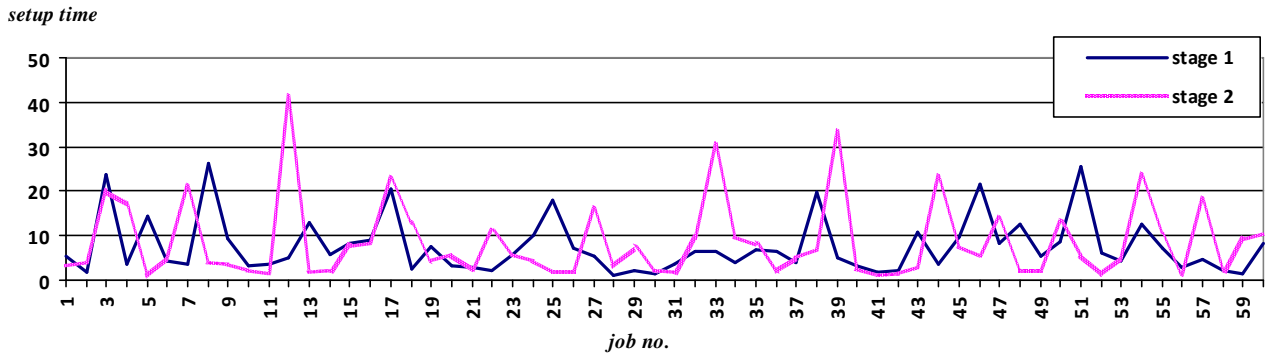


**Figure 5. Benchmark no. 2 (generated with the parameters in Table 5 below), distributions of setup times on stage 1 and on stage 2 are *uniform***

Figure 5 about here

From Table 4 with benchmarks no. 2 to no. 6, heuristic $H^{❶}$ does differ from $H^{❷}$ in the case of makespan minimization. Figure 5 shows that setup times of benchmark no. 2 have a uniform distribution (by construction, according to the parameters in Table 5) and that they do not have a similar distribution between stage 1 and stage 2 (as opposed to benchmark no. 1 in Figure 4). The same occurs for the other randomly generated benchmarks.

From computation experiments overall it can be observed that heuristic $H^{❷}$ based on the new proposed expression of the critical ratio, which considers the processing (setup) times on all the remaining stages is to be preferred, particularly for the more general case where setup times on different stages are *not correlated*.

## 5. 1 Extension to *S* stages

To show the applicability of the proposed heuristics to *S* stages, five new benchmarks have been generated with the parameters in Table 5 and can be downloaded as supplementary material. To compare the performance of the proposed approach on two-stage benchmarks with the new 5-stage benchmarks the following criteria has been adopted: the selected heuristic parameters from Table 1 ($T$=18 and $H^{❷}$) and the worst conditions denoted by * in Table 5 have been selected.

**Table 5. Benchmarks parameters**

| Parameter | Symbol | Benchmarks no. 2 to no. 6 | Benchmarks no. 7 to no. 11 |
|---|---|---|---|
| Number of jobs | $N$ | 60 | 60 |
| Job size | $z_i$ | Uniform[1,4] | Uniform[1,4] |
| Job setup time | $s_{ij}$ | Uniform[1,30], $j$=1<br>Uniform[1,45], $j$=2 | Uniform[1,12] |
| Number of stages | $S$ | 2 | 5 |
| Number of machines per stage | $M_j$ | 3, $j$=1<br>4, $j$=2 | 3* |
| Machine capacity | $u_j$ | 30, $j$=1<br>24, $j$=2 | 30 |
| Release date | $r_i$ | Uniform[0,480] | Uniform[0,3600]* |
| Due date | $d_i$ | Uniform[360,1440] | Uniform[900,3600] |
| Number of operators per stage | $v_j$ | 2, $j$=1<br>4, $j$=2 | 2* |
| Machine processing time | $p_j$ | 70 | 70 |

Results of computation experiments on the new benchmarks no. 7 to no. 11 are listed in Table 6. The LB for tardy jobs has been met only in two cases (compared to five cases for benchmarks no. 2 to no. 6 in Table 4), whereas the efficiency parameter (the makespan) has decreased approximately by one order of magnitude and one benchmark (no. 11) is solved to the optimality and has met the LB.

**Table 6. Benchmarks no. 7 to no. 11 (generated with the parameters in Table 5). Tardy jobs and makespan and respective lower bounds (LB) for the best heuristic parameters $T$=18 and $H^{\bullet}|V^{10}|v_1{}^2|v_2{}^2|v_3{}^2|v_4{}^2$. In bold LBs that have been met**

| Benchmark no. | LB for $\sum_{i=1}^{N} U_i$ | LB for $C_{\max}$ | $C_{\max}$ | $\sum_{i=1}^{N} U_i$ | %gap$_{H\text{-}LB}$ (8) |
|---|---|---|---|---|---|
| 7 | 5 | 1555 | 1565 | **5** | 0.6 |
| 8 | 4 | 1571 | 1589 | 5 | 1.1 |
| 9 | 4 | 1557 | 1573 | 5 | 1.0 |
| 10 | 4 | 1557 | 1561 | 5 | 0.3 |
| 11 | 5 | 1573 | **1573** | **5** | 0.0 |

## 6. Conclusions

Two new heuristics have been proposed for the non-permutation hybrid flowshop scheduling problem, with parallel batching machines, compatible parallel batch families and manual machine loading modeled as sequence-independent uniform setup times, with the purpose of reducing the number of tardy jobs (and the makespan).

Jobs are sorted to form batches according to a dynamic priority, updated at each new event, e.g. the release of a new job. The job priority depends on an innovative formulation of the critical ratio ($CR_{setup}$) of the available time between current time and due date versus the remaining processing (setup) time, according to two heuristics which consider respectively individual stages or all stages. In this latter formulation the algorithms has been extended to $S$ stages and it improves the weighted modified operation due date rule (WMOD) for two-stage flowshop, which only considers imminent setup.

Batches are closed (machine setup) and machines start within a time limit (sliding time window) determined by preliminary tests and updated automatically.

Computation experiments have shown that the combination of the $CR_{setup}$ rule and the sliding time window mechanism has produced an improvement of tardy jobs and makespan.

This work has the following highlights:

- The benefit of *fragmenting batches* to achieve better schedules instead of loading machines to full capacity, found in previous work has been confirmed with both the new proposed heuristics.
- A machine *capacity limit* in the heuristic is *not necessary* because the optimal time window is much lower than the one used in previous work (less than 20 minutes compared to 70). This result further stresses the benefit of fragmenting batches until all available machines are loaded. The result is that machines run more often and with lower loads. This should be taken into account where energy requirements are more relevant (like in the case of furnaces running at high temperature) than in the examined case. A general recommendation seems reducing the machine capacity and increasing their number, ideally to the limit of single-job batches as in flexible (or hybrid) flowshop scheduling. However a constraint to reducing the machine capacity is given by the job size: the minimum machine capacity is given by the maximum job size.
- Regarding the two-stage case, operators should be assigned to the two stages proportionally to the respective setup times of scheduled jobs (*constant setup speed* at different stages). This is confirmed from previous work, showing that a constant flow of jobs as opposed to waiting to

achieve full machine capacity is beneficial. By introducing the sliding time window parameter, the system achieves some kind of cyclic behavior with synchronized stages. A reduction of one operator (economic impact) is more tolerable than a wrong (*unbalanced*) assignment on the two stages (organization impact), producing no effect on the number of tardy jobs and slight makespan increase.

The given problem and the provided approach offer further investigation opportunities, including:

- finding possible relationships between time window and release/due date and setup times, as an alternative to the presented exhaustive search;
- optimizing "fragmentation", e.g. batch sizing and machine capacity/number;
- correlating setup time distribution on the *S s*tages and operators' assignment;
- optimization algorithms, such as metaheuristics, may increase the performance.

## Acknowledgement

The authors would like to thank the anonymous referees for the in-depth observations.

## References

Abu-suleiman A., Prattz, D. and Boardman, B. (2005). The modified critical ratio: towards sequencing with a continuous decision domain. *International Journal of Production Research*. 43(15),3287-3296.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*. 187(3), 985-1032.

Berry, W. L. and Rao, V. (1975). Critical ratio scheduling: an experimental analysis. *Management Science*, 22, 192-201.

Chand S., Traub R. and Uzsoy R. (1997). Rolling horizon procedures for the single machine deterministic total completion time scheduling problem with release dates. *Annals of Operations Research*. 70, 115–25.

Chiang, T.-C. and Fu, L.-C. (2007). Using dispatching rules for job shop scheduling with due date-based objectives. *International Journal of Production Research*. 45(14):3245–3262.

Chiang, T.-C. and Fu, L.-C. (2009). Using a family of critical ratio-based approaches to minimize the number of tardy jobs in the job shop with sequence dependent setup times. *European Journal of Operational Research*. 196:78–92.

Colledani, M., and Tolio, T. (2005). A decomposition method to support the configuration/reconfiguration of production systems. *CIRP Annals - Manufacturing Technology*. 54(1), 441-444.

Garcia, J.M. and Lozano, S. (2005). Production and delivery scheduling problem with time windows, *Computers & Industrial Engineering*. 48(4), 733-742

Garey, M. R. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. San Francisco: W.H. Freeman and Company.

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

Kanet, J. J., and Li, X. (2004). A weighted modified due date rule for sequencing to minimize weighted tardiness. *Journal of scheduling*. 7(4), 261-276.

Kapanoglu, M. and Alikalfa, M. (2011). Learning IF–THEN priority rules for dynamic job shops using genetic algorithms. *Robotics and Computer-Integrated Manufacturing*. 27:47–55.

Lee, Y. H., and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*. 100(3), 464-474.

Liao, C.J., Liao, L.M. and C.T. (2006). Tseng, A performance evaluation of permutation vs. non-permutation schedules in a flowshop. *International Journal of Production Research*. 44, 4297–4309.

Mason, S. J., Fowler, J. W., and Matthew Carlyle, W. (2002). A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*. 5(3), 247-262.

Mönch, L., Balasubramanian, H., Fowler, J. W., and Pfund, M. E. (2005). Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers & Operations Research*. 32(11), 2731-2750.

Perez, I. C., Fowler, J. W., and Carlyle, W. M. (2005). Minimizing total weighted tardiness on a single batch process machine with incompatible job families. *Computers & operations research*. 32(2), 327-341.

Pfund, M. E., Mason, S. J., and Fowler, J. W. (2006). Semiconductor manufacturing scheduling and dispatching. *Handbook of Production Scheduling*. 213-241.

Pfund, M., Fowler, J.W., Gadkan, A., and Chen, Y. (2008) Scheduling jobs on parallel machines with setup times and ready times, *Computers & Industrial Engineering*. 54(4), 764-782.

Pickardt, C. W., Hildebrandt, T., Branke, J., Heger, J. and Scholz-Reiter, B. (2013). Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics*. doi:10.1016/j.ijpe.2012.10.016.

Ribas, I., Leisten, R. and Framinan, J.M. (2010). Review and classification of hybrid flowshop scheduling problems from a production system and a solutions procedure perspective, *Computers & Operations Research*. 37:1439–1454.

Rossi, A. and Lanzetta, M. (2013a). Scheduling Flow Lines with Buffers by Ant Colony Digraph, *Expert Systems with Applications*. 40(9), 3328-3340, doi:10.1016/j.eswa.2012.12.041.

Rossi, A. and Lanzetta, M. (2013b). Native Metaheuristics for Non-Permutation Flowshop Scheduling. *Journal of Intelligent Manufacturing (JIMS)*, doi:10.1007/s10845-012-0724-8.

Rossi, A. and Lanzetta, M. (2013c). Non-permutation Flow Line Scheduling by Ant Colony Optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, doi:10.1017/S0890060413000176.

Rossi, A., Puppato, A. and Lanzetta, M. (2013). Heuristics for Scheduling a Two-stage Hybrid Flowshop with Parallel Batching Machines: an Application on Hospital Sterilization Plant. *International Journal of Production Research*. 51(8), 2363-2376, doi:10.1080/00207543.2012.737942.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operartion Research*. 169(3):781–800.

Shafaei, R. and Brunn, P. (1999). Workshop scheduling using practical (inaccurate) data Part 1: The performance of heuristic scheduling rules in a dynamic job shop environment using a rolling time horizon approach. *International Journal of Production Research*. 37(17):3913-3925.

Tandon, M., Cummings, P.T., and LeVan, M.D. (1991). Flowshop sequencing with non-permutation schedules. *Computers and Industrial Engineering*, 15(8), 601–607.

Ying, K.-C., Gupta, J. N.D., Lin S.-W. and Lee, Z.-J. (2010). Permutation and non-permutation schedules for the flowline manufacturing cell with sequence dependent family setups. *International Journal of Production Research*. 48(8), 2169–2184.