Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 35 (2008) 1151 – 1175

computers & operations research

www.elsevier.com/locate/cor

Modeling realistic hybrid flexible flowshop scheduling problems

Rubén Ruiz[a,*], Funda Sivrikaya Şerifoğlu[b], Thijs Urlings[c]

[a]*Department of Applied Statistics and Operations Research, Polytechnic University of Valencia, Valencia, Spain*
[b]*Department of Management, Abant Izzet Baysal University, Bolu, Turkey*
[c]*Department of Quantitative Economics, Maastricht University, Maastricht, Netherlands*

Available online 18 September 2006

Abstract

This paper aims to contribute to the recent research efforts to bridge the gap between the theory and the practice of scheduling by modelizing a realistic manufacturing environment and analyzing the effect of the inclusion of several characteristics in the problem formulation. There are several constraints and characteristics that affect the scheduling operations at companies. While these constraints are many times tackled in the literature, they are seldom considered together inside the same problem formulation. We propose a formulation along with a mixed integer modelization and some heuristics for the problem of scheduling n jobs on m stages where at each stage we have a known number of unrelated machines. The jobs might skip stages and, therefore, we have what we call a hybrid flexible flowshop problem. We also consider per machine sequence-dependent setup times which can be anticipatory and non-anticipatory along with machine lags, release dates for machines, machine eligibility and precedence relationships among jobs. Manufacturing environments like this appear in sectors like food processing, ceramic tile manufacturing and several others. The optimization criterion considered is the minimization of the makespan. The MIP model and the heuristics proposed are tested against a comprehensive benchmark and the results evaluated by advanced statistical tools that make use of decision trees and experimental designs. The results allow us to identify the constraints that increase the difficulty.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Hybrid flexible flowshop; Realistic scheduling; Sequence-dependent setup times; Precedence constraints

1. Introduction

The first studies on scheduling appeared in the 1950s with the paper of Johnson [1]. Since then, scheduling has attracted a lot of interest and the prolific scientific literature in the field contains hundreds of papers dealing with the most diverse aspects and problem variations. However, there has always been a so-called “gap” between the theory and practice of scheduling. There are many studies where this separation is recognized. For example, in the work of Graves [2] some possible solutions so to aid in bridging this gap were given. Ledbetter and Cox [3] and Ford et al. [4] showed that the application of scheduling theory was almost non-existent in reality at the date of their study. McKay et al. [5] cited many practical situations that the theory of scheduling does not solve. Olhager and Rapp [6] stated that existing methods are difficult to use and omit important aspects frequently found in practice. Other authors have directly criticized scheduling research in general and the flowshop scheduling literature in particular (see [7,8] or more recently McKay et al. [9], where many pending issues in scheduling research are pointed out). Reviews of the state

* Corresponding author. Tel.: +34 96 387 70 07x74946; fax: +34 96 387 74 99.
E-mail addresses: rruiz@eio.upv.es (R. Ruiz), serifoglu_f@ibu.edu.tr (F.S. Şerifoğlu), T.Urlings@Alumni.Unimaas.NL (T. Urlings).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.07.014

of the art for different scheduling environments also allow us to draw similar conclusions (see [10,11] or [12]). The results of the statistical review of Reisman et al. [13] are also interesting, from a total of 184 reviewed papers, only five (less than a 3%) dealt with realistic production settings.

There is a recent trend where research work is targeting at providing solution approaches to more realistic problems. However, there has been little effort towards providing models to complex scheduling problems where many realistic situations are jointly considered. In this paper, we propose a new formulation along with a mixed integer mathematical model and some heuristics for a problem that tackles highly realistic flowshop scheduling environments where we find many simultaneous restrictions and characteristics. These include release dates for machines, unrelated parallel machines at each stage, sequence-dependent setups on machines, machine eligibility, time lags on operations and precedence constraints among jobs. While this set of characteristics might seem excessive, problems found in real life contain many of them. Some examples are the food processing industry, where one finds many canning lines with different unrelated machines, with setups and lags in order to ensure/avoid cooling of product before sealing cans. Other examples are the ceramic tile manufacturing or the processing of wood and the manufacture of furniture. We aim at investigating the effect of including such realistic considerations on problem difficulty. In order to test the performance of the proposed model and to characterize the factors that affect its performance we have solved a comprehensive benchmark and then analyzed the results by means of advanced statistical tools that, to the best of our knowledge, have not been applied in this context before. These tools are automatic interaction detection (AID) techniques and their derivatives along with decision trees. The results of the thorough analysis allow us to identify which characteristics of the complex problem considered result in harder to solve problems. For solving realistically sized problems, we also have employed various heuristic rules.

The rest of the paper is organized as follows: Section 2 provides a literature review on realistic scheduling problems. In Section 3, we describe the problem considered in detail. We also give a detailed example in Section 4. Section 5 develops the mixed integer programming (MIP) mathematical model whereas Section 6 develops the heuristic methods. Section 7 gives a complete computational evaluation of the model and the heuristics along with a statistical testing. Finally, some conclusions on this study are given.

2. Literature review

Realistic settings have been studied by several authors. Odugawa et al. [14] provide a survey on evolutionary computation applications to real-world problems in metal forming industry, paper industry and chemical industry. Some researchers address real-world problems in their papers. Parthasarathy and Rajendran [15], for example, address the problem of scheduling in a real-life flowshop with jobs having sequence-dependent setup times and different relative weights (or importance). The objective is the minimization of the maximum weighted tardiness of a job and total weighted tardiness of jobs. A heuristic algorithm based on simulated annealing is proposed and evaluated. Pearn et al. [16] present a case study on the integrated circuit (IC) final testing scheduling problem with reentry, which is taken from a final testing shop floor in an IC manufacturing factory. The problem involves precedence constraints, serial-processing stage, batch-processing stage, job clusters, job-cluster dependent processing times, due dates, machine capacity constraints, and sequence-dependent setup times. Another scheduling problem from industry is presented by Bertel and Billaut [17]. They provide a MIP model for the three-stage hybrid flowshop (HFS) scheduling problem and heuristic approaches to solve it. Tanev et al. [18] solve a multi-objective, real-world, flexible job shop scheduling problem by hybridizing priority rules and GAs by incorporating several such rules in the chromosome representation. Andrés et al. [19] address the problem of products grouping in the tile industry. The production system is classified as a three-stage HFS with sequence-dependent and separable setup times. The basic concept of “exploiting similarities”, taken from the group technology philosophy, is used to identify a set of families integrated by products with common features.

Several researchers include realistic considerations and constraints into their problem definitions. Kochhar et al. [20] provide a local search approach to a realistic flexible flow line problem environment with setups, buffer capacities, blocking, starvation, breakdowns and downtimes. Schutten [21] provides some hints on how the shifting bottleneck procedure for the job shop problem can be modified so to consider transportation times, resource requirements as well as other constraints. Botta-Genoulaz [22] proposes several heuristics for a flowshop with multiple identical machines per stage, positive time lags, and out-tree precedence constraints as well as sequence-independent setup and removal times. Kurz and Askin [23] develop new or modified heuristics to solve a flexible flow line problem with identical parallel machines, non-anticipatory sequence-dependent setup times and the objective of makespan minimization. In a later paper (see [24]), the researchers develop a GA approach. Fondrevelle et al. [25] study permutation flowshop problems with minimal and/or maximal time lags, where the time lags are defined between couples of successive operations of jobs. They present theoretical results concerning two-machine cases and develop an optimal branch-and-bound procedure to solve the m machine case. Ruiz and Maroto [26] propose some genetic algorithms for a HFS problem with unrelated parallel machines per stage, sequence-dependent setup times and machine eligibility. The researchers conduct several experiments with a set of random instances as well as with real data taken from companies of the ceramic tile manufacturing sector. Although many realistic considerations and constraints are addressed in several papers in literature, very few papers consider such realistic constraints jointly. To the best of our knowledge, there is no attempt to jointly consider the set of realistic constraints included in the problem formulation of this paper.

3. Problem formulation

In a HFS we have a set N of jobs, $N = \{1, \dots, n\}$, that have to be processed in a set M of stages, $M = \{1, \dots, m\}$. At every stage i, $i \in M$ we have a set $M_i = \{1, \dots, m_i\}$ of unrelated parallel machines that can process the jobs where $m_i \geqslant 1$. Every job has the same flow pattern and passes through all stages and is processed by exactly one machine at every stage. We consider the flexible case where stages might be skipped. The following additional definitions and notation will help in formulating the problem:

1. F_j: set of stages that job j visits, $1 \leqslant |F_j| \leqslant m$,
2. p_{ilj}: processing time of job j, $j \in N$ at machine l, $l \in M_i$, inside stage i. If a given job j skips stage i ($i \notin F_j$) then $p_{ilj} = 0, \forall l \in M_i$,
3. rm_{il}: release date of machine l inside stage i. This date indicates when the machine is released from previous work and can start processing. No operation on machine l can be started before rm_{il},
4. E_{ij}: set of eligible machines that can process job j at stage i. Any eligible machine might process job j while non-eligible machines cannot process job j. Clearly $1 \leqslant |E_{ij}| \leqslant m_i$ if job j is processed at stage i and $|E_{ij}| = 0$ if job j skips stage i. Also, $p_{ilj} = 0$ if $l \notin E_{ij}$,
5. P_j: set of predecessors of job j. Any job may have zero or more predecessors indicating that the first operation of job j should not start until all the last operations of jobs in P_j have finished. All sets P_j define a precedence graph among jobs which is directed and acyclic,
6. lag_{ilj}: time lag between the end of the processing of job j at machine l inside stage i and the beginning in the next stage in which job j is processed. We will simply refer to this next stage in the sequence as $i + 1$. We allow for positive or negative lags between successive operations in jobs. If $lag_{ilj} < 0$ operations overlap and if $lag_{ilj} > 0$ then a waiting time exists between successive operations. Only if $lag_{ilj} = 0$ we have the standard situation in which one operation might not start until the previous operation has finished. Note that for any given job j, $lag_{ilj} = 0, \forall l \in M_i$, if $i \notin F_j$ or if for a given $l \in M_i, l \notin E_{ij}$. Additionally, if i is the last stage for job j, then $lag_{ilj} = 0, \forall l \in M_i$. There are some additional considerations for negative lag times or overlaps: If ($lag_{ilj} < 0$) then this overlap has to be less than or equal to the processing time, i.e., $|lag_{ilj}| \leqslant p_{ilj}$ otherwise we could have a situation where successive operations start before a preceding operation starts. Similarly, $|lag_{ilj}| \leqslant p_{i+1,l',j}, \forall l' \in E_{i+1,j}$, i.e., the overlap should be smaller than or equal to the processing time on any machine at the next stage, otherwise successive operations might finish before preceding operations,
7. S_{iljk}: machine-based sequence-dependent setup time on machine l at stage i when processing job k, $k \in N$, after having processed job j. There is an associated binary value, A_{iljk}, that when one indicates that the setup time is anticipatory, i.e., the setup might be done before job k is released at the previous stage. If $A_{iljk} = 0$ then the setup is non-anticipatory and job k should be on machine l for setting up. It should be noted that $S_{iljk} = 0$ if $j = k$ and that $S_{iljk} = 0, \forall j, k \in N, l \in M_i$ if one or more of the following conditions are satisfied: $i \notin F_j, i \notin F_k, l \notin E_{ij}, l \notin E_{ik}$ or $j \in P_k$. The value of A_{iljk} in these cases is irrelevant and will be assumed 0.

According to points 1 and 2, the machines available at each stage are unrelated and also jobs might skip some stages. We will refer to this problem as the hybrid flexible flow line problem or HFFL. This situation can be seen as a generalization of the hybrid flowshop or HFS (also many times referred to as flowshop with multiple machines or FSMP) where there are usually several stages with identical machines and the flexible flow line problem or FFL where there is one machine per stage but machines can be skipped. Point 3 modelizes a very common situation found in industry: when scheduling one has to consider that the shop is currently in operation and production environments are seldom found "empty". Additionally, point 4 modelizes another common situation found in practice where special jobs can only be processed on certain machines or specialized machines that can only process some given families of jobs. Point 5 models the existence of auxiliary products in the set of jobs, i.e., some jobs might require as inputs other jobs that must be finished first, as well as many other realistic situations. Point 6 allows the consideration of several practical cases. A positive lag between successive operations of the same job is useful when for example drying or cooling of products must occur before further operations can be carried out. Negative lags model other situations where for example large batches of small products have been modelized as jobs and not all products need to be finished on a stage before operations start on the following stage. Finally, point 7 considers the frequent existence of setup times in industry. Setup operations include cleaning, adjusting or changing the configuration of machines between the processing of successive jobs. The most general and complex case is when setup times are sequence dependent. In this work, we also consider the existence of both anticipatory and non-anticipatory setups. Some setups on machines might require the jobs to be already present on the machine (i.e., fixing the product to be processed on the machine) and are therefore non-anticipatory.

As it can be seen, many realistic situations arising in practice can be modeled with the aforementioned characteristics that we include in this work. As a matter of fact, this work stems from observation of real life cases. For example, points 1 and 2 modelize a very common case in industries. If one stage in a production process is a bottleneck, managers usually end up adding parallel machines at that stage in order to lessen the bottleneck. Commonly, new machines will have greater speeds or special characteristics so that they will have different processing speeds compared to existing machines. This is very common in filling and canning stages on food processing industries as well as in the kiln firing stage in ceramic tile manufacturing. Wood polishing stages in the manufacturing of furniture is another common example where unrelated parallel machines do exist. Similarly, some special products might skip stages as for example unpolished vintage furniture, or the optional polish for mirror-finish ceramic tiles.

Similarly, point 4 reflects the special machines purchased for processing some special products or conversely, some multipurpose machines that can process a wide variety of products but the special ones. Multi format/special format molding machines in ceramic tile industries are a clear example.

Point 5 is also very common. Some jobs in wood processing might consist in preparing the wood boards that need to be cut (used as an input) for the manufacture of furniture. Also, in the ceramic tile industry, when producing a stair step, both the upper and the lower part of the step need to be fully produced prior to the assembly of the final step. These are just some examples.

Point 6 is also frequent. Examples are the cooling time needed in prepared food prior to canning or the drying of the glaze on ceramic tile manufacturing before kiln firing. Overlaps also exist in the production of ceramic tiles or in the semiconductor industry where a job is actually a batch of many small ceramic tiles or ICs, respectively.

Finally, point 7 modelizes the almost ever present setup times. In ceramic tile industries, a very large setup needs to be carried out in the molding stage when a different format tile is to be produced. This setup is not needed if the next type of tile has the same format and hence the sequence-dependent setups. In wood processing, prior to polishing a board of wood, the board itself needs to be attached to the machine and this attachment is in itself a setup where clamps must be added/removed. Therefore, it is a non-anticipatory setup time.

It is easy to see that all of the aforementioned characteristics in our proposed model appear simultaneously in the industrial problems cited and probably in many more.

This HFFL problem is significantly more complex than other flowshops, where machines are identical, able to process all jobs and ready at time 0. Gourgand et al. [27] showed the total number of possible solutions for a HFS to be $n!(\prod_{i=1}^{m} m_i)^n$ which is much larger than the number of possible solutions in the regular flowshop scheduling problem. In this HFFL problem, due to the possibility of skipping stages, the machine eligibility and precedence constraints among jobs, the number of feasible solutions depends on the instance and might be smaller. However, many simplifications of the proposed problem are $\mathcal{NP}$-Hard. Gupta [28] showed the flowshop with multiple processors or FSMP problem with only two stages to be $\mathcal{NP}$-Hard even when one of the two stages contains a single machine. A more general result on the $\mathcal{NP}$-Hardness of HFS problems is provided by Lee and Vairaktarakis [29]. The addition of precedence constraints does not simplify the problem since the two parallel machine problem with makespan objective and general precedence constraints ($P/prec/C_{\max}$) is strongly $\mathcal{NP}$-Hard according to Pinedo [30]. The same applies to the other characteristics considered as for example the regular flowshop with sequence-dependent setup times ($F/S_{ijk}/C_{\max}$) that was shown to be $\mathcal{NP}$-Complete by Gupta [31]. So we can conclude that the considered HFFL problem is also $\mathcal{NP}$-Hard.

We will work with a popular optimization criterion in scheduling which is the minimization of the maximum completion time, commonly known as makespan or $C_{\max}$. Considering the well-known three field notation for scheduling problems, the extension for HFS proposed by Vignier et al. [12], and the aforementioned characteristics and constraints, the HFFL problem considered here can be denoted as: $HFFLm, ((RM^{(i)})_{i=1}^{(m)})/rm, lag, S_{ijk}, M_j, prec/C_{\max}$.

4. Example problem

The following example will help in illustrating this complex HFFL problem. We have an instance with five jobs and three stages with two machines in the first two stages and only one machine in the third. Therefore $n=5$, $m=3$, $m_1=m_2=2$ and $m_3=1$. Jobs 1 and 5 visit all three stages whereas job 2 skips stage 3, job 3 skips stage 1 and job 4 skips stage 2. So we have that $F_1=F_5=\{1,2,3\}$, $F_2=\{1,2\}$, $F_3=\{2,3\}$ and $F_4=\{1,3\}$. Furthermore, jobs 4 and 5 are preceded by jobs 2 and 3, i.e., $P_4=P_5=\{2,3\}$. Table 1 gives the eligibility for the jobs on the machines. Table 2 shows the processing times of jobs and the release dates for machines.

Tables 3 and 4 show the lag values and sequence-dependent setup times, respectively. In the cells of Table 4, the setup time and, between parentheses, the anticipatory flag for every machine at each stage are given. Note that "–" means that both the setup time and the anticipatory value are 0.

Table 1
Eligibility (E_{ij}) of jobs on machines for the example

	i	1	2	3
j	1	{1, 2}	{2}	{1}
	2	{1, 2}	{1, 2}	–
	3	–	{1}	{1}
	4	{2}	–	{1}
	5	{1, 2}	{1, 2}	{1}

Table 2
Processing times (p_{ilj}) and release dates for machines (rm_{il}) for the example

	i	1		2		3
	l	1	2	1	2	1
	rm_{il}	4	3	8	16	23
	p_{ilj}					
j	1	10	15	0	8	6
	2	6	9	11	4	0
	3	0	0	9	0	8
	4	0	10	0	0	6
	5	11	14	6	12	3

Table 3
Lag times (lag_{ilj}) of jobs for the example

	i	1		2	
	l	1	2	1	2
j	1	10	2	0	−4
	2	2	−2	0	0
	3	0	0	3	0
	4	0	1	0	0
	5	−5	−6	−3	8

Table 4
Sequence-dependent setup times and anticipatory flags (S_{iljk} and A_{iljk}) for the example

j	$i=1$					$i=2$					$i=3$				
	k					k					k				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	–,–	3(1),6(1)	–,–	–,8(1)	4(1),2(1)	–,–	–,6(1)	–,–	–,–	–,6(1)	–	–	6(1)	3(1)	9(1)
2	4(0),5(0)	–,–	–,–	–,6(1)	1(1),4(1)	–,5(1)	–,–	6(0),–	–,–	4(1),2(0)	–	–	–	–	–
3	–,–	–,–	–,–	–,–	–,–	–,–	8(0),–	–,–	–,–	5(1),–	4(0)	–	–	1(0)	8(1)
4	–,8(0)	–,–	–,–	–,–	–,2(1)	–,–	–,–	–,–	–,–	–,–	5(0)	–	–	–	2(1)
5	6(0),10(0)	–,–	–,–	–,4(0)	–,–	-,4(1)	–,–	–,–	–,–	–,–	2(0)	–	–	6(0)	–

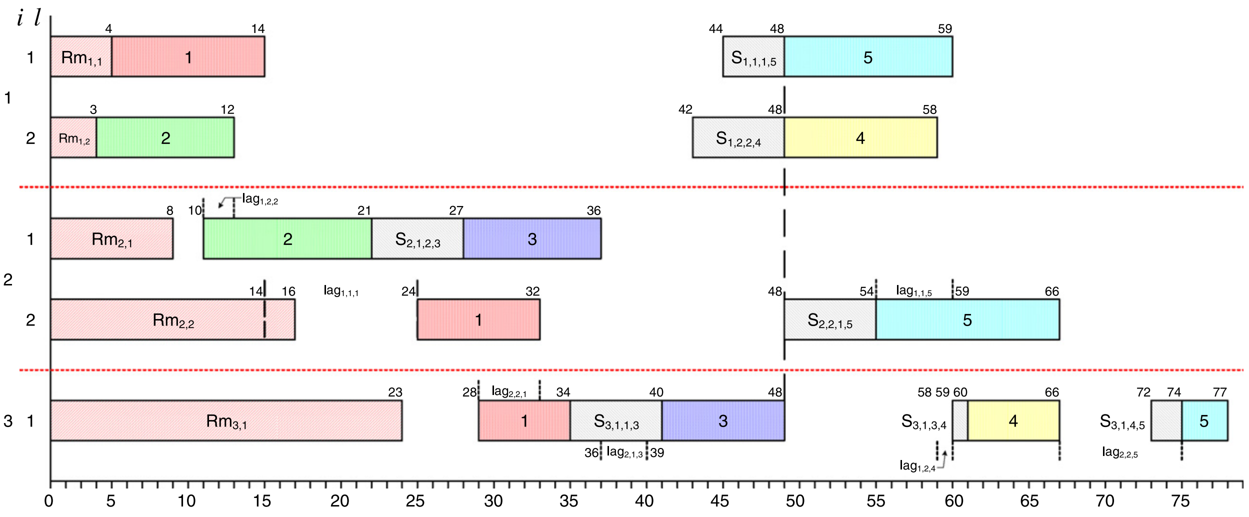


Fig. 1. Gantt chart for the example problem. $C_{\max} = 77$.

Now let us suppose that we have a permutation of jobs $\pi = \{2, 1, 3, 4, 5\}$ for which we want to find the makespan. The assignment of jobs to machines at each stage, is as follows:

$$\begin{pmatrix} 2 & 1 & 0 & 2 & 1 \\ 1 & 2 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

where each column represents the machines assigned at each stage to the job at the corresponding position in the permutation. It should be clear that π is feasible from the perspective of the precedence constraints among jobs and that the assignment of jobs to machines at each stage is also feasible with respect to the eligibility constraints. The resulting Gantt chart is shown in Fig. 1.

It can be observed that jobs 4 and 5 must wait for their predecessors before starting. Also, sometimes the lag times allow for an overlap of the operations whereas in other situations these lags force a waiting period between successive operations of the same job. A closer look at job 4 reveals that, even with a missing stage 2, the lag still holds for the third stage. Some setups are anticipatory (see for example, the setup between jobs 1 and 5 on machine 1 in the first stage), whereas other setups are non-anticipatory (see for example, the setup between jobs 3 and 4 on machine 1 in the last stage). This means that the setup between jobs 1 and 5 in the first stage, for example, could be started anytime between 14 and 44, whereas the setup between jobs 3 and 4 in the last stage can start only after job 4 arrives at that stage (and, in this case, after lag_{124} is also considered).

5. MIP model formulation

There are several well-known branch-and-bound approaches developed for the relatively easier problem of HFS scheduling [32–34]. Although, to the best of our knowledge, there is not any branch-and-bound approach developed for the hybrid flexible flow line problem with the same or similar characteristics as considered here yet, some researchers provide MIP formulations for simpler problems. Sawik [35] presents MIP formulations for scheduling of a flexible flow line with blocking. The machines are assumed to be identical. The basic MIP formulation is enhanced to model reentrant shops, where jobs visit a set of stages more than once, and to incorporate alternative processing routes for jobs. Kurz and Askin [23] consider a hybrid flexible flow line environment with identical parallel machines and non-anticipatory sequence-dependent setup times. Their objective is to minimize the makespan. They provide a MIP formulation for the problem and propose some lower bounds.

In the following, we provide a MIP formulation for the HFFL problem defined in Section 3. We first need some additional notation in order to simplify the exposition of the model:

- G_i is the set of jobs that visit stage i, ($G_i \subseteq N$ and $G_i = \{j|i \in F_j\}$),
- $G_{il} \subseteq G_i$ is the set of jobs that can be processed on machine l inside stage i, i.e., $G_{il} = \{j|i \in F_j \wedge l \in E_{ij}\}$,
- S_k gives the complete and unchained set of successors of job k, i.e., $S_k = \{j|k \in P_j\}$,
- FS_k (LS_k) is the first (last) stage that job k visits.

The model involves the following decision variables:

$$X_{iljk} = \begin{cases} 1 & \text{if job } j \text{ precedes job } k \text{ on machine } l \text{ at stage } i, \\ 0 & \text{otherwise,} \end{cases}$$

$$C_{ij} = \text{Completion time of job } j \text{ at stage } i,$$

$$C_{\max} = \text{ Maximum completion time.}$$

The objective function is

$$\min C_{\max}. \tag{1}$$

And the constraints are:

$$\sum_{\substack{j \in \{G_i, 0\} \\ j \neq k, j \notin S_k}} \sum_{l \in E_{ij} \cap E_{ik}} X_{iljk} = 1, \quad k \in N, \ i \in F_k, \tag{2}$$

$$\sum_{\substack{j \in G_i \\ j \neq k, j \notin P_k}} \sum_{l \in E_{ij} \cap E_{ik}} X_{ilkj} \leqslant 1, \quad k \in N, \ i \in F_k, \tag{3}$$

$$\sum_{\substack{h \in \{G_{il}, 0\} \\ h \neq k, h \neq j \\ h \notin S_j}} X_{ilhj} \geqslant X_{iljk}, \quad j, k \in N, \ j \neq k, \ j \notin S_k, \ i \in F_j \cap F_k, \ l \in E_{ij} \cap E_{ik}, \tag{4}$$

$$\sum_{l \in E_{ij} \cap E_{ik}} (X_{iljk} + X_{ilkj}) \leqslant 1, \quad j \in N, \quad k = j+1, \ldots, n, \ j \neq k, \ j \notin P_k, \ k \notin P_j, \ i \in F_j \cap F_k, \tag{5}$$

$$\sum_{k\in G_{il}} X_{il0k} \leqslant 1, \quad i \in M,\ l \in M_i, \tag{6}$$

$$C_{i0} = 0, \quad i \in M, \tag{7}$$

$$C_{ik} + V(1 - X_{iljk}) \geqslant \max\left\{\max_{p\in P_k} C_{LS_p,p}, rm_{il}, C_{ij} + A_{iljk}\cdot S_{iljk}\right\} + (1 - A_{iljk})\cdot S_{iljk} + p_{ilk},$$
$$k \in N,\ i = FS_k,\ l \in E_{ik},\ j \in \{G_{il}, 0\},\ j \neq k,\ j \notin S_k, \tag{8}$$

$$C_{ik} + V(1 - X_{iljk}) \geqslant \max\left\{C_{i-1,k} + \sum_{\substack{h\in\{G_{i-1},0\}\\ h\neq k, h\notin S_k}} \sum_{l'\in E_{i-1,h}\cap E_{i-1,k}} (lag_{i-1,l',k}\cdot X_{i-1,l',h,k}),\right.$$
$$\left. rm_{il}, C_{ij} + A_{iljk}\cdot S_{iljk}\right\} + (1 - A_{iljk})\cdot S_{iljk} + p_{ilk},$$
$$k \in N,\quad i \in \{F_k \backslash FS_k\},\ l \in E_{ik},\ j \in \{G_{il}, 0\},\ j \neq k,\ j \notin S_k, \tag{9}$$

$$C_{\max} \geqslant C_{LS_j,j}, \quad j \in N, \tag{10}$$

$$X_{iljk} \in \{0, 1\}, \quad j \in \{N, 0\},\ k \in N,\ j \neq k,\ k \notin P_j,\ i \in F_j \cap F_k,\ l \in E_{ij} \cap E_{ik}, \tag{11}$$

$$C_{ij} \geqslant 0, \quad j \in N,\ i \in F_j. \tag{12}$$

The set of constraints (2) assures that every job should be preceded by exactly one job on only one machine at each stage. Here only the possible variables are considered. Note that for every stage and machine we introduce a dummy job 0, which precedes the first job at each machine. This also allows for the consideration of initial setup times. Constraint set (3) is similar in the way that every job should have at most one successor. Constraint set (4) forces that if a job is processed on a given machine at a stage, then it should have a predecessor on the same machine. This is a way of forcing that assignments are consistent in the machines. Constraint set (5) avoids the occurrence of cross-precedences. Note again that only the possible alternatives are considered. With constraint set (6) we enforce that dummy job 0 can only be predecessor of at most one job on each machine at each stage. Constraint set (7) simply ensures that dummy job 0 is completed at time 0 in all stages. Constraint set (8) controls the completion time of jobs at the first stage they start processing by considering all eligible machines. The value *V* represents a big number so to make the constraint redundant if the assignment variable is zero. Notice that precedence relationships are considered by accounting for the completion of all the predecessors of a given job. Note also that both types of sequence-dependent setup times (anticipatory and non-anticipatory) are also taken into account. Constraint set (9) gives the completion time on subsequent stages. Here the completion time of the same job in the previous stage along with the lag time is considered. Constraint set (10) defines the maximum completion time. Finally, (11) and (12) define just the decision variables.

6. Heuristic methods

The proposed MIP model can only be tested in moderately sized instances. In order to analyze the effect of the different characteristics of the HFFL problem in larger instances, we also employ some simple heuristics. It is important to remark that there are many methods proposed in the literature and that our intention is to study the effect of the different characteristics on the performance of regular and widely available heuristics. We adapt some well-known dispatching rules and the efficient NEH algorithm of Nawaz et al. [36] to the HFFL. For the dispatching rules, jobs are scheduled in a stage-by-stage basis rather than scheduling each job in all stages. The rules are applied at each moment to the jobs that can be scheduled according to the precedence relationships (i.e., those jobs that are "eligible").

The eligible jobs set is denoted by R. The details of the heuristics are the following:

- Shortest processing time (SPT): For each stage, and among eligible jobs that are processed in that stage, the job with the smallest average processing time in the stage is scheduled. The average processing time (*APT*) for a given job j in a stage $i \in F_j$ is calculated as follows:

$$APT_{ij} = \frac{\sum_{l\in E_{ij}} p_{ilj}}{|E_{ij}|}.$$

- Longest processing time (LPT): Same as SPT but the job scheduled is the one with the largest *APT*.
- Least work remaining (LWR): For each stage, and among eligible jobs to be processed in that stage, the job with the smallest sum of average processing times in the remaining stages (including the present stage) is scheduled. Note that we only consider the remaining stages in which the job is processed.
- Most work remaining (MWR): Same as LWR but the job scheduled is the one with the largest sum of average processing times in the remaining stages.
- Most work remaining with average setup times (MWR-AST): It is a refinement of MWR in which we also consider an average of the pending setup times. This average setup is calculated for the present job and all others in R on the remaining stages and eligible machines, resulting in scheduling the job with the highest *APT*/*AST* calculation as follows:

$$APT/AST_{ij} = \sum_{k=i,k\in F_j}^{m} \left(APT_{kj} + \frac{\sum_{h\in R,h\notin P_j,h\neq jk\in F_h} \sum_{l\in(E_{kj}\cap E_{kh})} S_{kljh}}{|H_{kj}|} \right),$$

where H_{kj} is the set containing all possible setups, i.e.,

$$H_{kj} = \{(h, l \in R \times E_{kj}) | k \in F_h, h \notin P_j, h \neq jl \in E_{kh}\}.$$

- NEH: The NEH algorithm proposed by Nawaz et al. [36] initially for the regular flowshop problem is profusely used in the scheduling literature. Contrary to the previous dispatching rules, the NEH works with a permutation of jobs that are scheduled one by one in all stages, so the schedule is obtained on a job-by-job basis. In the first step of the NEH, jobs are sorted in decreasing total average processing time, $TAPT_j = \sum_{i\in F_j} APT_{ij}$. NEH algorithm starts by taking the first two jobs with higher $TAPT_j$ and the schedules associated with the two possible sequences are calculated. The best sequence from the two is used as a basis for inserting the job with the third highest $TAPT_j$ value. This third job is inserted in the three possible positions of the sequence containing the first two jobs and the best sequence among the three is kept for inserting the fourth job. The process continues until all jobs have been considered. We have modified this insertion step of the NEH method in order to take into account the precedence constraints. When a job is to be scheduled, we look for the earliest and latest possible insertion position in the incumbent sequence, i.e., the job cannot be placed before any of its predecessors and no later than any of its successors.

Both the dispatching rules and the modified NEH algorithm provide the next job to be processed in the current stage or in all stages in the case of the modified NEH. In addition to that, we need a rule for assigning that specific job to one of its eligible machines at a given stage. Following the work of Ruiz and Maroto [26] we assign the job to the eligible machine that can finish it at the earliest possible time. This rule takes into account the possible lag, setup time (anticipatory or not) as well as the different processing speeds and release dates of the machines.

7. Computational evaluation

We define two complete sets of instances to test the MIP model and the heuristics and to investigate the effect of realistic considerations on problem difficulty. Due to the complexity of the problem and the number of different characteristics considered, a total of 10 factors are combined at the levels given in Table 5. The heuristics are tested additionally on a set of larger-sized instances which differ in the factors listed in Table 6.

The differences between the small and large instances are in the n, m, m_i and NP_j factors. The largest instances have 100 jobs, 8 stages and 4 machines per stage (32 machines in total). The distribution of the processing times is

Table 5
Factors considered in the design of the test bed. Small instances

Factor	Symbol	No. of levels	Values
Number of jobs	n	6	5, 7, 9, 11, 13, 15
Number of stages	m	2	2, 3
Number of unrelated parallel machines per stage	m_i	2	1, 3
Distribution of the release dates for the machines	rm_{il}	2	0, $U[1, 200]$
Probability for a job to skip a stage	PF_j	2	0%, 50%
Probability for a machine to be eligible	PE_{ij}	2	50%, 100%
Distribution of the setup times as a percentage of the processing times	DS_{iljk}	2	$U[25, 74]$, $U[75, 125]$
Probability for the setup time to be anticipatory	PA_{iljk}	2	$U[0, 50]$%, $U[50, 100]$%
Distribution of the lag times	$Dlag_{ilj}$	2	$U[1, 99]$, $U[-99, 99]$
Number of preceding jobs	NP_j	2	0, $U[1, 3]$

Table 6
Modified factors for the design of the large instances

Factor	Symbol	No. of levels	Values
Number of jobs	n	2	50, 100
Number of stages	m	2	4, 8
Number of unrelated parallel machines per stage	m_i	2	2, 4
Number of preceding jobs	NP_j	2	0, $U[1, 5]$

fixed to $U[1, 99]$. The total number of combinations is $6 \cdot 2^9 = 3072$ and $2^{10} = 1024$ for the small and large instances, respectively. There are three replicates per combination, so in total there are 9216 small and 3072 large instances. It is important to remark that when generating the instances all restrictions affecting the data (see Section 3 for details) were considered. For example, every job must visit at least one stage and at least one machine on every visited stage must be eligible (and thus the factors PF_j and PE_{ij} must be controlled). Additionally, special care must be given to the generation of the precedences among jobs. We will use the set of small instances to test the MIP model, and both sets for testing the heuristic methods.

7.1. MIP model evaluation

An LP model is constructed for each small problem instance and then solved with CPLEX 9.1 on a Pentium IV 3.2 GHz computer with 1 Gbyte of RAM memory. It could be argued that an ad hoc branch-and-bound algorithm would perform better than the latest and best regarded commercial solver available. However, we refrained from developing such a method mainly due to the fact that obtaining a tight lower bound for the HFFL problem considered is a very difficult task. As a matter of fact, taking into account only the sequence-dependent setup times already defeats most possible lower bounds since the amount of setups depends on the sequence. Using commercial solvers for flowshop problems with setups has been pursued in the literature. For example, Stafford and Tseng [37] solved instances of up to nine jobs and nine machines for a $F/S_{ijk}/C_{\max}$ problem with LINDO commercial solver. The authors needed about 6622 and 300 s CPU time in a Pentium III 800 MHz computer for each one of the two models they proposed, respectively. According to the recent review and evaluation of heuristics for the same problem in Ruiz et al. [38], most exact methods proposed for the $F/S_{ijk}/C_{\max}$ problem are very limited and the bounds proposed not tight. For all the above reasons, it seems plausible that an efficient solver using linear relaxations of variables as bounds would perform reasonably well.

Due to the large number of instances, we impose a time limit for every model of 300 s. For each model, we record a categorical variable called "type of outcome" with three possible values 0, 1 and 2. Outcome 0 means that an optimum solution was found, in which case we record the time needed and the optimum $C_{\max}$ value. Outcome 1 means that the 300 s time limit was reached and a feasible integer solution was found. In this case we record the solution found and the gap between this solution and the best MIP bound. Lastly, the outcome value 2 indicates that no feasible integer solution could be found within the time limit.

Table 7 shows the results for all the controlled factors in the case of $n = 7$, $m = 3$ and $m_i = 3$. Each cell gives the average of the three replicates. In the table, the percentage of instances for which an optimum solution can be found within the time limit (%Opt) and the average time needed to reach this optimum solution (Av time) are displayed. The percentage of instances for which an integer feasible solution is found within the time limit (%Limit) is also displayed in the table.

From Table 7, it follows for the combination $n = 7$, $m = 3$ and $m_i = 3$ that when all stages are visited ($PF_j = 0\%$) the models are more difficult to solve. The same applies to the case when all machines inside a stage are eligible ($PE_{ij} = 100\%$). The combination $PF_j = 0\%$ and $PE_{ij} = 100\%$, i.e., every stage is visited and every machine is eligible, is especially difficult: In no instance an optimal solution could be found within the time limit, regardless of other parameter values. These results confirm what is expected, with more stages and more eligible machines, more feasible solutions and, therefore, more time is needed for obtaining the optimum solution. As regards the MIP model, the factors that affect the distribution of the data in the instance (rm_{il}, DS_{iljk}, PA_{iljk} and $Dlag_{ilj}$) do not seem to have a clear significant effect on the difficulty. The aggregated results for all the values of n, m, m_i and averaged over the other parameters are shown in Table 8. As it has been pointed out, the total number of variables and constraints depends on many factors and ultimately, on all the data in a given instance. We show the average number of variables and constraints for the MIP models in Table 8 as well.

It can be observed in Table 8 that the previous findings are confirmed: increasing n, m and m_i results in harder problems. However, there is an interesting result. Increasing the number of unrelated parallel machines m_i for the larger values of n (13 and 15) seems to have a positive impact, although small, on the percentage of instances with integer optimum solutions. For example, for $n = 15$, $m = 2$ and $m_i = 1$ we find that only 0.26% of the instances end up with optimum solutions but for $m_i = 3$ this percentage increases up to 8.85%. Initially this result might seem counter-intuitive since with more unrelated parallel machines per stage more variables are needed in the model. The explanation to this behavior comes from the fact that n is, by far, the most influential factor. With $n = 15$ the number of variables is very large. Having more unrelated parallel machines at each stage means that the assignment of jobs to machines at each stage becomes more important. With only one machine per stage there is no assignment and solutions are solely influenced by the permutation of the jobs. In other words, more unrelated parallel machines per stage helps lessening the sheer effect of the number of jobs on the difficulty of the instances.

We can say that the overall performance of the proposed MIP model, given its complexity and number of variables, is rather good. In Table 8, we have that the most complex case is given by $n = 15$, $m = 3$ and $m_i = 3$, and only 3.12% of the problems could be solved to optimality and in another 70.31% of the cases a feasible integer solution was obtained before the time limit was reached. Therefore in 26.57% of the problems no solution could be found. As shown, in this case the average number of variables and constraints is 841 and 3386, respectively. The average gap between the feasible integer solutions and the best bounds found by CPLEX is 71.48% which is deemed as large. However, we are only allowing for a total of 300 s of CPU time per instance, which, given the complexity of the problem to be solved, is quite short.

7.2. MIP model statistical analysis

Most valuable statistical tools suited for analyzing the effect of the 10 considered factors on the performance of the MIP model are nullified by the fact that the response variable considered (type of outcome) is categorical. Under this circumstance, ANOVA technique, for example, cannot be applied. Non-parametric statistical tests like the well-known Kuskal-Wallis or Wilcoxon signed-rank tests that can take categorical response variables are also not suitable, since with these tools, the choices are limited to mostly paired tests and with 10 factors and all the possible interactions not much information can be obtained. Therefore, we propose the application of an advanced technique called AID.

AID recursively bisects experimental data according to one factor into mutually exclusive and exhaustive sets that describe the response variable in the best possible and statistically significant way. AID works on an interval scaled or purely categorical response variable and maximizes the sum of squares between groups by means of a given statistic. The original AID technique was proposed by Morgan and Sonquist [39]. Kass [40] developed an improved version called chi-squared automatic interaction detection (CHAID) by including statistical significance testing in the partition process and by allowing multi-way splits of the data. Later, Biggs et al. [41] further improved CHAID method and created what is known as exhaustive CHAID algorithm that does a more thorough job when examining all possible partitions for each factor. These techniques are of common use in the fields of education, population studies, market

Table 7
MIP model results for $n = 7$, $m = 3$ and $m_i = 3$ with a CPU time limit of 300 s

rm_{il}	PF_j	PE_{ij}	DS_{iljk}	PA_{iljk}	$U[0, 50]\%$				$U[50, 100]\%$			
				$Dlag_{ilj}$	$U[1, 99]$		$U[-99, 99]$		$U[1, 99]$		$U[-99, 99]$	
				NP_j	0	$U[1, 3]$	0	$U[1, 3]$	0	$U[1, 3]$	0	$U[1, 3]$
0	0%	50%	$U[25, 74]$	%Opt	66.67	33.33	33.33	100	33.33	33.33	100	100
				Av time	61.89	0.34	30.83	50.03	72.86	1.84	101.68	100.27
				% Limit	33.33	66.67	66.67	0	66.67	66.67	0	0
			$U[75, 125]$	%Opt	100	100	33.33	66.67	0	33.33	66.67	100
				Av time	94.6	5.83	280.97	16.05	0	19.42	169.93	37.66
				% Limit	0	0	66.67	33.33	100	66.67	33.33	0
		100%	$U[25, 74]$	%Opt	0	0	0	0	0	0	0	0
				Av time	0	0	0	0	0	0	0	0
				% Limit	100	100	100	100	100	100	100	100
			$U[75, 125]$	%Opt	0	0	0	0	0	0	0	0
				Av time	0	0	0	0	0	0	0	0
				% Limit	100	100	100	100	100	100	100	100
	50%	50%	$U[25, 74]$	%Opt	100	100	100	100	100	100	100	100
				Av time	0.17	0.04	0.31	0.15	0.15	0.05	0.07	0.07
				% Limit	0	0	0	0	0	0	0	0
			$U[75, 125]$	%Opt	100	100	100	100	100	100	100	100
				Av time	0.14	0.03	0.24	0.01	0.16	0.09	0.13	1.51
				% Limit	0	0	0	0	0	0	0	0
		100%	$U[25, 74]$	%Opt	100	100	100	66.67	100	66.67	100	100
				Av time	14.96	77.52	16.85	127.33	2.17	174.21	7.29	11.82
				% Limit	0	0	0	33.33	0	33.33	0	0
			$U[75, 125]$	%Opt	100	66.67	100	100	100	66.67	66.67	100
				Av time	1.46	62.05	8.99	45.64	55.17	11.87	0.67	52.67
				% Limit	0	33.33	0	0	0	33.33	33.33	0
$U[1200]$	0%	50%	$U[25, 74]$	%Opt	66.67	66.67	66.67	100	0	66.67	33.33	100
				Av time	48.64	33.56	169.77	71.79	0	117.49	27.47	83.72
				% Limit	33.33	33.33	33.33	0	100	33.33	66.67	0
			$U[75, 125]$	%Opt	66.67	100	33.33	100	66.67	100	33.33	100
				Av time	69.32	28.01	41.36	5.06	80.17	45.37	57.58	60.7
				% Limit	33.33	0	66.67	0	33.33	0	66.67	0
		100%	$U[25, 74]$	%Opt	0	0	0	0	0	0	0	0
				Av time	0	0	0	0	0	0	0	0
				% Limit	100	100	100	100	100	100	100	100
			$U[75, 125]$	%Opt	0	0	0	0	0	0	0	0
				Av time	0	0	0	0	0	0	0	0
				% Limit	100	100	100	100	100	100	100	100
	50%	50%	$U[25, 74]$	%Opt	100	100	100	100	100	100	100	100
				Av time	2.03	0.06	9	0.2	0.04	0.04	0.74	0.31
				% Limit	0	0	0	0	0	0	0	0
			$U[75, 125]$	%Opt	100	100	100	100	100	100	100	100
				Av time	0.07	0.04	0.21	0.5	0.23	0.08	0.13	0.16
				% Limit	0	0	0	0	0	0	0	0
		100%	$U[25, 74]$	%Opt	100	33.33	100	100	100	66.67	66.67	100
				Av time	4.27	100.83	0.78	74.54	67.9	8	0.31	4.86
				% Limit	0	66.67	0	0	0	33.33	33.33	0
			$U[75, 125]$	%Opt	100	100	66.67	66.67	100	100	100	100
				Av time	14.81	5.87	103.05	2.22	80.78	87.48	9.43	21.7
				% Limit	0	0	33.33	33.33	0	0	0	0

Table 8
Aggregated MIP model results for a CPU time limit of 300 s

n	m	2		3	
	m_i	1	3	1	3
5	%Opt	100.00	100.00	100.00	83.07
	Av Time	0.30	1.47	11.03	20.63
	%Limit	0.00	0.00	0.00	16.93
	Variables	34.22	65.49	48.29	93.21
	Constraints	121.68	225.27	172.15	322.00
7	%Opt	79.17	77.60	74.48	63.54
	Av Time	11.78	17.78	8.93	34.67
	%Limit	20.83	22.40	25.52	36.46
	Variables	61.21	126.46	83.79	177.46
	Constraints	227.64	473.41	313.89	660.57
9	%Opt	53.39	59.38	46.35	39.06
	Av Time	30.42	42.73	29.31	33.35
	%Limit	46.61	40.62	37.24	58.85
	Variables	109.70	213.54	153.79	294.41
	Constraints	430.13	832.98	598.00	1136.39
11	%Opt	32.29	29.95	22.92	24.22
	Av Time	56.30	31.26	49.36	53.68
	%Limit	50.00	69.01	51.04	62.24
	Variables	168.55	319.74	236.46	444.31
	Constraints	682.27	1274.33	946.11	1755.09
13	%Opt	8.07	17.45	5.73	12.76
	Av Time	63.94	48.88	113.05	53.07
	%Limit	67.45	71.87	65.63	65.10
	Variables	236.02	445.58	330.86	444.31
	Constraints	975.39	1802.58	1352.49	1755.09
15	%Opt	0.26	8.85	0.78	3.12
	Av Time	81.00	83.13	43.59	92.85
	%Limit	72.40	71.61	63.80	70.31
	Variables	315.40	598.59	441.51	840.52
	Constraints	1319.54	2426.40	1828.30	3386.19

research as well as many others. We use exhaustive CHAID for analyzing our experimental data. The method starts with all data classified into a first (root) node. Then, all factors are considered for splitting the node and the best multi-way split according to the levels of each factor is calculated. To this end, a statistical significance test is carried out so to rank the factors on how well they split the node. A chi-squared (χ^2) test is used for categorical factors. After the node has been split, the same procedure is applied to all sub-nodes until no more significant partitions can be found or until a given stopping criterion is met. Usually, a classification or decision tree is obtained as a result of the application of the method. The resulting tree enables a careful study of the effect of the different factors, and what is more important, the interactions between them.

We use SPSS DecisionTree 3.0 software which implements exhaustive CHAID algorithms. All 10 factors as well as the response variable are deemed as categorical (nominal). We choose a minimum number of cases (data) for each node before splitting of 192. Nodes with fewer cases are not split. Furthermore, if splitting a parent node results in a child node with less than 96 cases, the node will not be split. These values are chosen after a close examination of initial test trees and to avoid splits in the trees on the basis of small data samples. We set a confidence level for splitting of 99.9% and a Bonferroni adjustment for multi-way splits that compensates the statistical bias in multi-way paired tests. The first three levels of the resulting tree are shown in Fig. 2.

In Fig. 2, the root node contains all data of the experiment and at that level, the most significant factor is the number of jobs or n. Therefore, the next level is composed of one node for every possible value of n. Moreover, this split is done

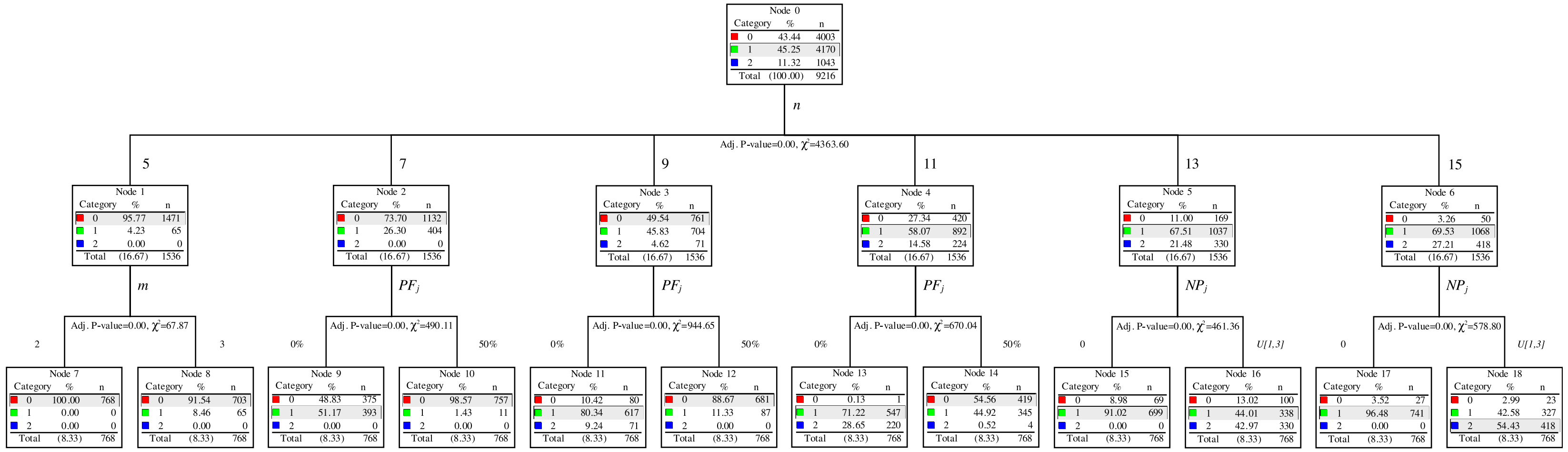

Node 0 Category % n; 0 43.44 4003; 1 45.25 4170; 2 11.32 1043; Total (100.00) 9216
n
Adj. P-value=0.00, χ²=4363.60
5
7
9
11
13
15
Node 1 Category % n; 0 95.77 1471; 1 4.23 65; 2 0.00 0; Total (16.67) 1536
Node 2 Category % n; 0 73.70 1132; 1 26.30 404; 2 0.00 0; Total (16.67) 1536
Node 3 Category % n; 0 49.54 761; 1 45.83 704; 2 4.62 71; Total (16.67) 1536
Node 4 Category % n; 0 27.34 420; 1 58.07 892; 2 14.58 224; Total (16.67) 1536
Node 5 Category % n; 0 11.00 169; 1 67.51 1037; 2 21.48 330; Total (16.67) 1536
Node 6 Category % n; 0 3.26 50; 1 69.53 1068; 2 27.21 418; Total (16.67) 1536
m
PF_j
PF_j
PF_j
NP_j
NP_j
Adj. P-value=0.00, χ²=67.87
Adj. P-value=0.00, χ²=490.11
Adj. P-value=0.00, χ²=944.65
Adj. P-value=0.00, χ²=670.04
Adj. P-value=0.00, χ²=461.36
Adj. P-value=0.00, χ²=578.80
2
3
0%
50%
0%
50%
0%
50%
0
U[1,3]
0
U[1,3]
Node 7 Category % n; 0 100.00 768; 1 0.00 0; 2 0.00 0; Total (8.33) 768
Node 8 Category % n; 0 91.54 703; 1 8.46 65; 2 0.00 0; Total (8.33) 768
Node 9 Category % n; 0 48.83 375; 1 51.17 393; 2 0.00 0; Total (8.33) 768
Node 10 Category % n; 0 98.57 757; 1 1.43 11; 2 0.00 0; Total (8.33) 768
Node 11 Category % n; 0 10.42 80; 1 80.34 617; 2 9.24 71; Total (8.33) 768
Node 12 Category % n; 0 88.67 681; 1 11.33 87; 2 0.00 0; Total (8.33) 768
Node 13 Category % n; 0 0.13 1; 1 71.22 547; 2 28.65 220; Total (8.33) 768
Node 14 Category % n; 0 54.56 419; 1 44.92 345; 2 0.52 4; Total (8.33) 768
Node 15 Category % n; 0 8.98 69; 1 91.02 699; 2 0.00 0; Total (8.33) 768
Node 16 Category % n; 0 13.02 100; 1 44.01 338; 2 42.97 330; Total (8.33) 768
Node 17 Category % n; 0 3.52 27; 1 96.48 741; 2 0.00 0; Total (8.33) 768
Node 18 Category % n; 0 2.99 23; 1 42.58 327; 2 54.43 418; Total (8.33) 768

Fig. 2. Decision tree with the first three levels shown in detail, time limit = 300 s.

with a very high level of confidence since the p-value is very close to 0 and the result of the χ^2 statistic is very high, i.e., n is the most influential factor on the response variable with a statistically significant effect. Within the resulting six nodes, as the value of n increases the number of cases for which no solution is found increases. In node 6, where $n = 15$, few instances were solved to optimality.

After this first multi-way split, each node is split into two according to different factors. For $n = 5$, m is the most influential factor whereas for $5 < n \leqslant 11$ the factor PF_j (probability for a job j to skip a stage) is more important. As it has been mentioned, a 0% probability for a stage to be skipped results in more difficult instances for all the values of n. Surprisingly, for $n = 13$ and 15 the factor NP_j (number of preceding jobs for job j) is the most discriminating. In the child nodes of nodes 5 and 6, there is an interesting observation. For the instances, where there are no precedence relations (nodes 15 and 17 with $NP_j = 0$) either an optimal solution or an integer feasible solution is found within the time limit of 300 s. In nodes 16 and 18 with $NP_j = U[1, 3]$, about half of the instances remain unsolved. This outcome is again counter-intuitive and a careful analysis is needed. While adding predecessors results in fewer variables since a job cannot be scheduled before one of its predecessors, it greatly complicates some of the constraints of the model, more precisely, constraint set (8). This affects the branch-and-bound algorithm used by CPLEX and results in instances being more difficult to solve. For reasons of space, the full tree cannot be shown in detail. Instead we have constructed a simplified tree shown in Fig. 3.

In this tree, we omit the values of the three types of outcome from the first three levels, since they can be seen in Fig. 2. From the fourth level until the last significant level we show at the edges the factors according to which parent node is split into child nodes. At a given node we show the absolute values of the three types of outcome. Also shown is the factor that results in further child node division or "–" if no further statistically significant divisions are found or if the stopping criterion for branching is met.

Apart from the already mentioned factors n, m and PF_j, there are other factors which determine differences on the three levels of the response variable. These are m_i, NP_j and PE_{ij} (probability for a machine in stage i to be eligible for job j). It is interesting that all other factors which affect mainly the distributions of setup times, anticipatory setups, lags and release dates for machines do not appear to be significant. Although not shown here, extending the previous tree by allowing parent and children nodes to have any number of data results in very little variations. Therefore, the proposed MIP model does not seem to be affected by the factors rm_{il}, DS_{iljk}, PA_{iljk} or $Dlag_{ilj}$.

As it has been mentioned before, the average gap obtained for the type of outcome 1 is more than 70% which makes us think that allowing for more time would not change the results significantly. In order to test this hypothesis, we ran all the experiments once more with the only difference that the allowed CPU time was increased from 300 to 900 s. The aggregated results for all the values of n, m and m_i are shown in Table 9.

It can be observed that in all situations the percentage of instances with optimum solutions (%Opt) increases. However, this increase is rather small. For $n = 15$, $m = 3$ and $m_i = 3$ we see that the percentage of optimum solutions has increased from 3.12 to 5.21 and the average time from 92.85 to 261.60 s. The total CPU time necessary for solving all instances with 900 s stopping time has been 1294 h (almost 54 days). Allowing for more CPU time seems to have a small effect on the number of optimum solutions obtained. Carrying out the exhaustive CHAID analysis yields the tree depicted in Fig. 4 (only the first three levels shown).

The two trees of Figs. 2 and 4 have little differences as regards the most influential factors. As expected, the number of optimum solutions and integer solutions (types of outcome 1 and 2) increase in general in Fig. 4, while the number of cases for which no solutions are found decreases. At the root node 46.9% of instances are solved to optimality (from the original 43.44% obtained with 300 s maximum CPU time), and the percentage of unsolved instances has decreased from 11.32 to 10.03. One interesting test that we can carry out is to see if this observed 3.46% of additional instances that are solved to optimality when 900 s of CPU time are allowed is statistically significant. Since the two sets of results represent dependent samples, we have carried out a McNemar test on paired proportions (see [42]). The results of this test are sound. There is a statistically significant difference between the percentages of instances solved to optimality when increasing the CPU time with a χ^2 value of 313.08 and a p-value close to 0. The 95% confidence interval of the percentage increase on solutions solved to optimality is [3.35, 3.50]%. Although there is a statistically significant difference, this is really small. Obtaining a maximum of 3.50% additional instances solved to optimality does not compensate the tripled CPU time. The full simplified tree is given in Fig. 5.

As can be seen in Fig. 5, there are fewer levels in the full simplified tree. This is also an expected outcome since the factors that had a weak effect in the case with 300 s maximum CPU time are nullified when more CPU time is

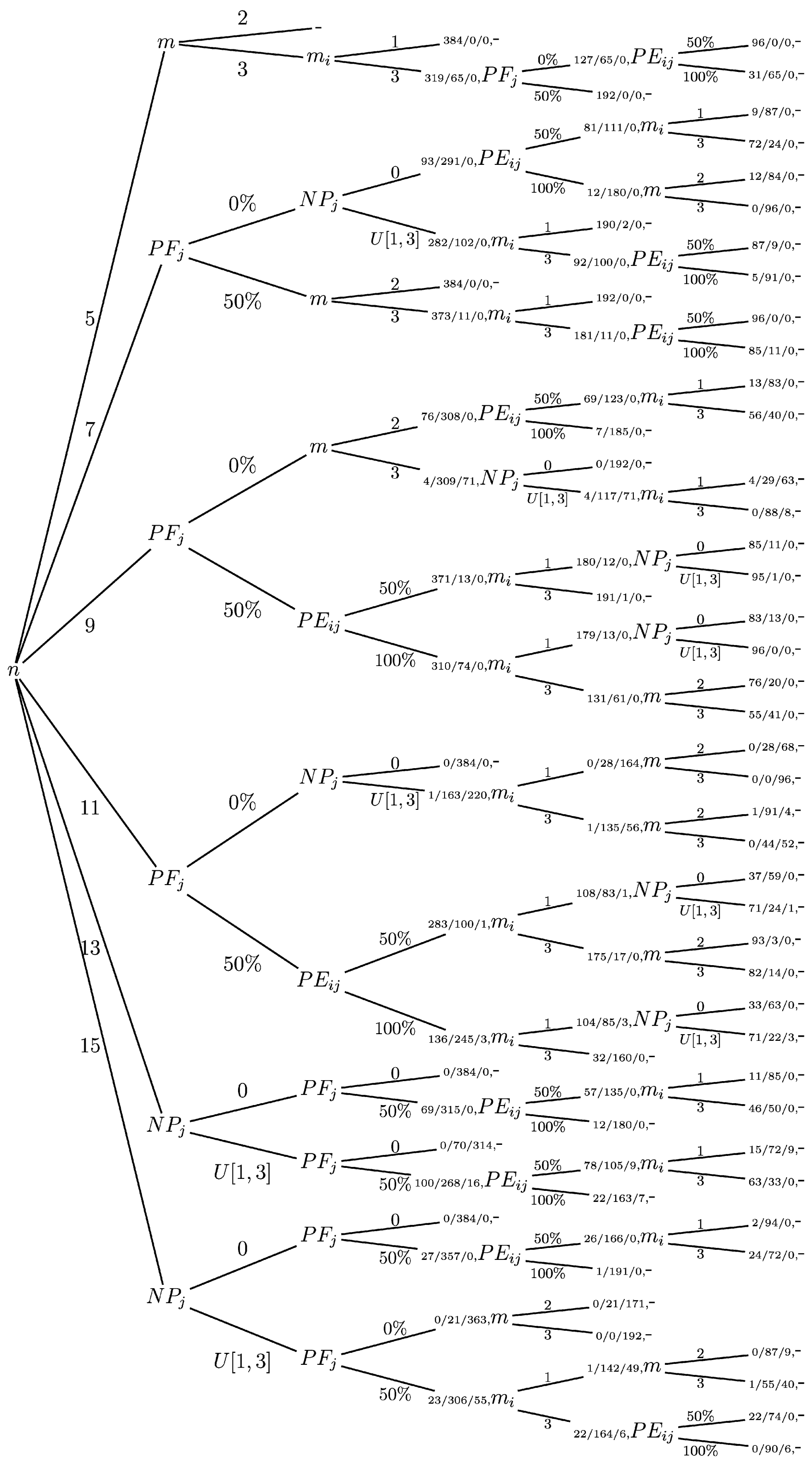


Fig. 3. Full simplified decision tree, time limit = 300 s.

Table 9
Aggregated MIP model results for a CPU time limit of 900 s

n	m	2		3	
	m_i	1	3	1	3
5	%Opt	100.00	100.00	100.00	90.36
	Av Time	0.32	2.06	10.47	73.14
	%Limit	0.00	0.00	0.00	9.64
7	%Opt	83.85	85.16	75.26	69.27
	Av Time	60.58	99.33	18.31	75.81
	%Limit	16.15	14.84	24.74	30.73
9	%Opt	60.16	65.36	48.44	41.41
	Av Time	124.30	89.95	51.38	65.79
	%Limit	39.84	34.64	38.54	58.33
11	%Opt	35.68	34.11	28.91	26.56
	Av Time	106.81	125.49	140.87	124.99
	%Limit	51.56	65.89	45.31	61.98
13	%Opt	14.06	20.31	8.85	16.93
	Av Time	254.17	146.95	230.03	209.46
	%Limit	61.98	73.44	63.54	61.46
15	%Opt	1.82	12.24	1.56	5.21
	Av Time	492.76	176.77	246.60	261.60
	%Limit	71.61	72.40	67.45	69.79

allowed, i.e., only the main factors n, m, m_i, PF_j, NP_j and PE_{ij} are statistically affecting the difficulty of the MIP model instances.

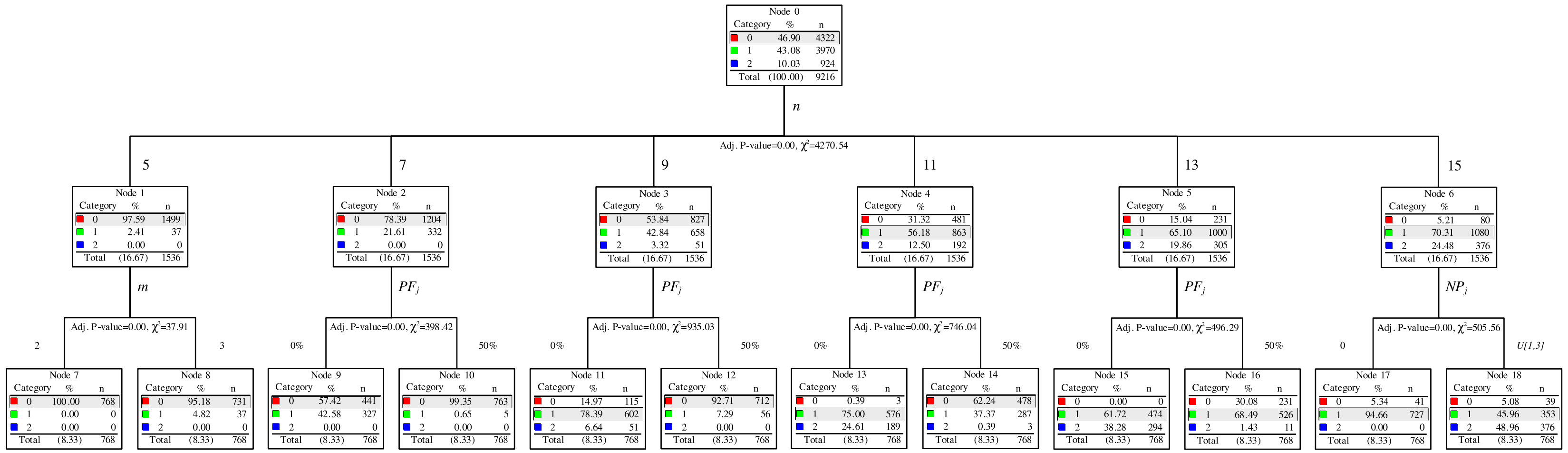
7.3. Evaluation of heuristics

We test the five proposed dispatching rules and the modified NEH algorithm both on small as well as on large instances. All heuristics have been coded in Delphi 2005 and run on the same computer used for testing the MIP model. In this case, for the small instances we measure the average percentage increase over the optimum solution, if known, or over the best solution obtained by the heuristics. For the large instances we measure the average percentage increase over the best solution obtained by the heuristics. The results are shown in Table 10.

As expected, the larger the value of n, the greater the deviations are, especially for the dispatching rules. Increasing the number of stages (m) also results in larger deviations. The effect of the number of machines per stage (m_i) is less clear for the smallest instances. However, as it was the case for the MIP model, more machines per stage seem to yield lower deviations for all methods. All the heuristics yield lower deviations in the larger instances when going from $m_i = 2$–4. As we can see, on average, SPT and LPT are the poorest performers, reaching almost a 300% deviation in the larger instances of $n = 100$, $m = 8$ and $m_i = 2$. Clearly, the modified NEH algorithm is vastly superior to the dispatching rules. For the small instances and lower values of n (where most optimum solutions are known) we see that the deviation of the modified NEH from these optimums is much lower than those of the dispatching rules and sometimes below 2%, like for example in $n = 5$, $m = 3$ and $m_i = 1$. For the largest instances, the modified NEH is the best performer. Although the small percentage deviations just indicate that the modified NEH is obtaining the best solutions, as there is no indication of how far these solutions might be from real optima.

The CPU times needed for all the heuristics are very small. As a matter of fact, the CPU times for the five dispatching rules are below what can be reliably measured. For the modified NEH algorithm, the average is less than 0.5 s for the largest instances of $n = 100$, $m = 8$ and $m_i = 4$ with precedence constraints and about 1.5 s for the same size and without precedence constraints.

In order to study the effect of the different characteristics of the HFFL problem on the application of heuristic methods, we use statistical experimental design on the continuous variables given by average percentage deviations.


n
Adj. P-value=0.00, χ²=4270.54
5
7
9
11
13
15
m
Adj. P-value=0.00, χ²=37.91
2
3
PF_j
Adj. P-value=0.00, χ²=398.42
0%
50%
PF_j
Adj. P-value=0.00, χ²=935.03
0%
50%
PF_j
Adj. P-value=0.00, χ²=746.04
0%
50%
PF_j
Adj. P-value=0.00, χ²=496.29
0%
50%
NP_j
Adj. P-value=0.00, χ²=505.56
0
U[1,3]

Node 0

Category	%	n
🟥 0	46.90	4322
🟩 1	43.08	3970
🟦 2	10.03	924
Total	(100.00)	9216

Node 1

Category	%	n
🟥 0	97.59	1499
🟩 1	2.41	37
🟦 2	0.00	0
Total	(16.67)	1536

Node 2

Category	%	n
🟥 0	78.39	1204
🟩 1	21.61	332
🟦 2	0.00	0
Total	(16.67)	1536

Node 3

Category	%	n
🟥 0	53.84	827
🟩 1	42.84	658
🟦 2	3.32	51
Total	(16.67)	1536

Node 4

Category	%	n
🟥 0	31.32	481
🟩 1	56.18	863
🟦 2	12.50	192
Total	(16.67)	1536

Node 5

Category	%	n
🟥 0	15.04	231
🟩 1	65.10	1000
🟦 2	19.86	305
Total	(16.67)	1536

Node 6

Category	%	n
🟥 0	5.21	80
🟩 1	70.31	1080
🟦 2	24.48	376
Total	(16.67)	1536

Node 7

Category	%	n
🟥 0	100.00	768
🟩 1	0.00	0
🟦 2	0.00	0
Total	(8.33)	768

Node 8

Category	%	n
🟥 0	95.18	731
🟩 1	4.82	37
🟦 2	0.00	0
Total	(8.33)	768

Node 9

Category	%	n
🟥 0	57.42	441
🟩 1	42.58	327
🟦 2	0.00	0
Total	(8.33)	768

Node 10

Category	%	n
🟥 0	99.35	763
🟩 1	0.65	5
🟦 2	0.00	0
Total	(8.33)	768

Node 11

Category	%	n
🟥 0	14.97	115
🟩 1	78.39	602
🟦 2	6.64	51
Total	(8.33)	768

Node 12

Category	%	n
🟥 0	92.71	712
🟩 1	7.29	56
🟦 2	0.00	0
Total	(8.33)	768

Node 13

Category	%	n
🟥 0	0.39	3
🟩 1	75.00	576
🟦 2	24.61	189
Total	(8.33)	768

Node 14

Category	%	n
🟥 0	62.24	478
🟩 1	37.37	287
🟦 2	0.39	3
Total	(8.33)	768

Node 15

Category	%	n
🟥 0	0.00	0
🟩 1	61.72	474
🟦 2	38.28	294
Total	(8.33)	768

Node 16

Category	%	n
🟥 0	30.08	231
🟩 1	68.49	526
🟦 2	1.43	11
Total	(8.33)	768

Node 17

Category	%	n
🟥 0	5.34	41
🟩 1	94.66	727
🟦 2	0.00	0
Total	(8.33)	768

Node 18

Category	%	n
🟥 0	5.08	39
🟩 1	45.96	353
🟦 2	48.96	376
Total	(8.33)	768

Fig. 4. Decision tree with the first three levels shown in detail, time limit = 900 s.

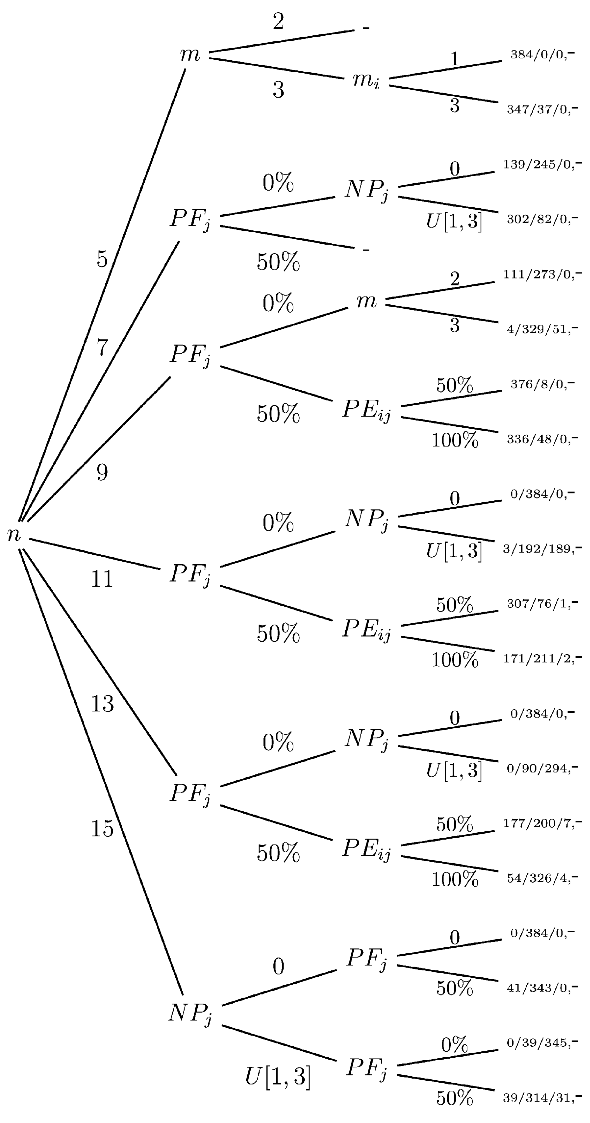


Fig. 5. Full simplified decision tree, time limit = 900 s.

More precisely, we carry out a complete factorial experiment where all characteristics of the problem instances (see Table 5) as well as the algorithms are considered as factors. The experiment is divided between small and large instances and the results analyzed by means of the ANOVA technique. For the small instances experiment we have $9216\times 6=55,296$ data. With such large experiment, all three hypotheses of the parametric ANOVA technique (normality, homoscedasticity and independence of the residuals) are easily satisfied. In the experiment, we only consider the simple factors and two level interactions. The full ANOVA table is omitted for the sake of brevity. However, there are a number of interesting findings. By far, the most influential factor is NP_j, larger even than the type of algorithm itself, which is the second most important factor. As a matter of fact, the interaction between these two factors is the third most influential element in the ANOVA. Fig. 6 shows the means plot of this interaction.

Recall that overlapping least significative difference (LSD) intervals mean that there are no statistically significant differences between the observed means. As we see, adding predecessors results in instances that are easier to solve by the dispatching rules. More importantly, the differences in performance among the heuristics are much more pronounced when there are no precedence constraints. This plot confirms that the modified NEH yields statistically better solutions and an interesting outcome: for the modified NEH, predecessors result in harder instances. The difference is small

Table 10
Aggregated results for the heuristics: average percentage increase over optimum or known best solutions

n	m	2		3	
	m_i	1	3	1	3
5	SPT	22.02	22.87	33.58	25.68
	LPT	22.67	20.09	32.12	23.32
	LWR	19.91	24.27	28.58	26.32
	MWR	14.85	18.07	18.34	18.76
	MWR-AST	12.89	18.30	17.33	18.66
	NEH	1.45	9.72	1.74	11.14
7	SPT	30.54	29.24	46.50	30.59
	LPT	31.25	27.94	46.58	27.80
	LWR	26.98	31.34	36.58	30.24
	MWR	20.98	22.10	25.70	18.94
	MWR-AST	20.48	22.45	25.32	19.11
	NEH	2.84	11.26	3.09	9.23
9	SPT	38.39	34.63	60.38	35.76
	LPT	38.79	31.19	59.21	31.41
	LWR	34.36	34.02	49.09	33.68
	MWR	25.54	24.36	33.54	22.41
	MWR-AST	24.25	26.01	33.49	22.64
	NEH	3.12	10.46	2.60	8.10
11	SPT	42.68	34.26	70.21	38.17
	LPT	44.47	32.26	68.30	37.15
	LWR	38.24	35.38	54.32	35.81
	MWR	28.41	23.79	39.11	24.78
	MWR-AST	27.64	24.57	37.99	24.63
	NEH	2.39	7.64	2.88	6.40
13	SPT	47.98	35.63	76.25	43.00
	LPT	46.71	33.47	75.12	40.22
	LWR	42.01	35.59	57.09	41.42
	MWR	29.67	23.93	39.86	25.91
	MWR-AST	28.87	23.86	40.10	26.73
	NEH	2.30	6.32	2.07	5.42
15	SPT	51.00	36.74	81.44	45.89
	LPT	49.03	33.88	81.45	43.35
	LWR	43.07	35.88	62.09	41.58
	MWR	30.35	23.38	43.26	25.67
	MWR-AST	28.43	23.89	42.62	26.82
	NEH	1.31	5.28	0.95	4.48
n	m	4		8	
	m_i	2	4	2	4
50	SPT	126.39	79.69	209.64	123.36
	LPT	126.03	78.22	210.05	122.11
	LWR	84.88	55.89	110.57	66.06
	MWR	54.19	31.76	67.12	36.67
	MWR-AST	52.49	31.51	62.67	35.42
	NEH	0.25	0.91	0.16	1.08
100	SPT	160.16	113.31	293.74	191.98
	LPT	162.70	111.23	291.86	190.62
	LWR	102.99	73.42	146.55	94.19
	MWR	64.31	41.73	89.15	52.87
	MWR-AST	63.81	42.54	84.01	50.90
	NEH	0.03	0.61	0.06	0.97

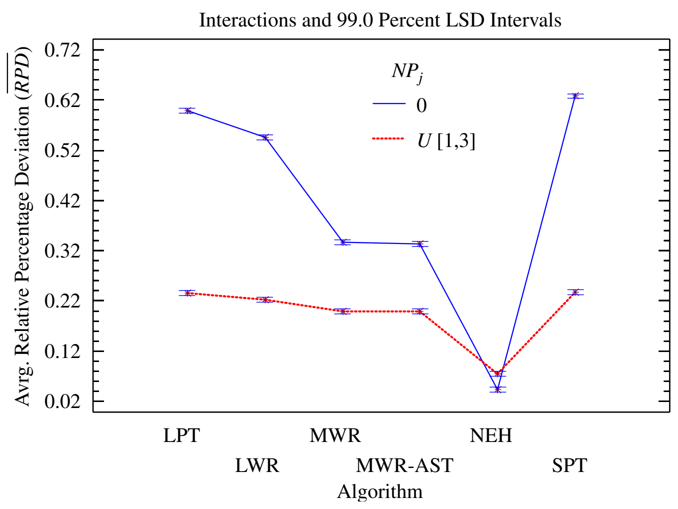


Fig. 6. Means plot and LSD intervals for the interaction between the type of algorithm and NP_j (number of preceding jobs) factors. Small instances.

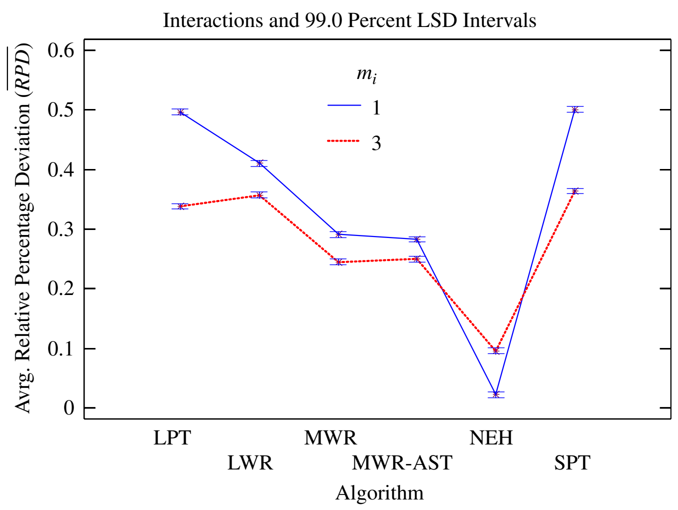


Fig. 7. Means plot and LSD intervals for the interaction between the type of algorithm and m_i (number of unrelated parallel machines per stage) factors. Small instances.

although statistically significant. The explanation to this outcome is clear, with predecessors there are less feasible permutations and therefore, less insertions, which lead to a slight deterioration in solution quality. Single factor plots (not shown) for n and m show what is expected. With more jobs and machines instances are harder to solve. A very interesting result is the behavior of the factor m_i (number of unrelated parallel machines per stage): with more machines per stage, instances are easier to solve. This is also the same case with the MIP, and it is confirmed by the interaction plot between the algorithm and m_i shown in Fig. 7.

With the exception of the modified NEH algorithm, all dispatching rules perform better if more machines per stage are available. As said in the case of the MIP, more machines per stage lessens the effect of poor job permutations as n grows. Although not shown here, the factors affecting the distribution of setup times (DS_{iljk}) and anticipatory setups (PA_{iljk}) are not even significant with a p-value of 0.13 and 0.22, respectively. Lag times distribution ($Dlag_{ilj}$), is marginally significant with the value $U[1, 99]$ being statistically better, although with a negligible difference. The distribution of the release dates for the machines (rm_{il}) is statistically significant but with a small difference in the

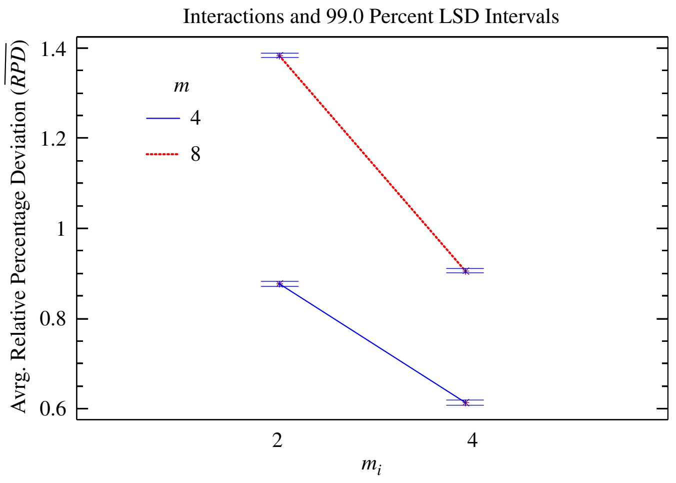


Fig. 8. Means plot and LSD intervals for the interaction between the number of stages m and the number of unrelated parallel machines per stage m_i. Large instances.

means of 0.05% between the two levels, the level $U[1, 200]$ being better. The last two remaining factors, PF_j and PE_{ij} yield expected results, with skipping stages and less eligible machines, instances are easier for most methods.

In the second experiment with the large instances the results are almost identical. NP_j and the algorithms have a very big influence. However, the differences for some factors are more pronounced. Factors DS_{iljk} and PA_{iljk} are not significant. In this case, the factor PE_{ij} is not significant either but the observed differences in $Dlag_{ilj}$ and rm_{il} are again marginally significant. The factors that affect the size of the instance, n and PF_j are also clearly significant with larger instances being noticeably more difficult to solve. Once again, $m_i = 4$ results in much easier instances. Additionally, the effect of more unrelated parallel machines increases with the number of stages as the interaction between these two factors shows in Fig. 8.

As can be seen, increasing the number of stages from 4 to 8 results in worse solutions but increasing the number of unrelated parallel machines per stage yields better solutions, with a more acute effect for eight stages.

This exhaustive analysis allows us to characterize the difficulty of the instances for this complex HFFL problem. A salient conclusion is that the setup to processing time ratio has no effect on the difficulty as regards the MIP model and the heuristics. The same can be said about other factors that initially were regarded as important, like the possibility of setups to be non/anticipatory or to a lesser extent the distribution of the time lags or the release dates for machines.

8. Conclusions and future work

In this paper, we have shown a complete formulation as well as a mixed integer programming mathematical model and some heuristics for a complex and realistic flowshop problem. In this problem several realistic characteristics are jointly considered. These include release dates for machines, existence of unrelated parallel machines at each stage of the flowshop, machine eligibility, possibility for jobs to skip stages, sequence dependent setup times, possibility for setup times to be both anticipatory as well as non-anticipatory, positive and/or negative time lags between operations and generalized precedence relationships between jobs. To the best of our knowledge, such a complex problem has not been considered before.

This modelization allows for the consideration of realistic scheduling environments, like those arising in the production of prepared food, ceramic tiles, wood processing and manufacture of furniture as well as many others.

In order to clearly identify the effect of each considered characteristic on the proposed mathematical model, we have solved a comprehensive benchmark and carried out an extensive statistical analysis by means of decision trees. This tool, to the best of our knowledge, has not been applied to the analysis of MIP model performance before. The analysis has allowed us to identify some interesting and counter-intuitive interactions between the many different characteristics of the realistic problem considered. The results establish a sound basis for further analyses of such a complex problem and for the development of more heuristics and/or metaheuristics, which are needed to solve larger sized problems in tolerable times. Although considering instances of up to 15 jobs is not practically relevant, our aim here is not to solve practically sized problems using MIP models and CPLEX, but to investigate the effect of the realistic characteristics included in the modelization on the problem difficulty. Furthermore, we have also proposed simple dispatching rules and an adaptation of the NEH algorithm that have been tested both on small as well as on large instances of up to 100 jobs. The effect of the realistic constraints has also been investigated with experimental designs. As a result, the following conclusions from the analysis can be drawn:

- Analysis of the MIP model on small instances with 300 s CPU time limit:
  - Preliminary analyses via tables:
    - When all stages are visited and/or when all machines inside a stage are eligible, the problem is especially difficult regardless of other parameter values.
    - Increasing n, m and m_i results in harder problems.
    - For larger values of n, more unrelated parallel machines per stage helps lessening the sheer effect of the number of jobs on the difficulty of the instances and has a small but positive effect on the performance.
  - Analysis with exhaustive CHAID:
    - The most significant factor is the number of jobs or n.
    - Adding predecessors greatly complicates some of the constraints of the model, which in turn results in instances being more difficult to solve for CPLEX.
    - n, m, PF_j (probability for a job to skip a stage), m_i, NP_j (number of predecessors), and PE_{ij} (probability for a machine in stage i to be eligible for job j) are found to have a significant effect on problem difficulty, while distributions of setup times, anticipatory setups, lags and release dates for machines are not significant.

- Analysis of the MIP model on small instances with 900 s CPU time limit shows that tripling the time limit:
  - Yields a statistically significant but small difference of 3.46% increase in the number of instances solved to optimality, and nullifies factors that have a weak effect in the case with 300 s maximum CPU time.
  - Only the main factors n, m, m_i, PF_j, NP_j and PE_{ij} statistically affect the difficulty of the MIP model instances.

- Evaluation of the heuristics:
  - The larger the values of n and m, the greater the deviations are, especially for the dispatching rules. More machines per stage seem to yield lower deviations for all methods.
  - On average, SPT and LPT are the poorest performers whereas the modified NEH algorithm is vastly superior to the dispatching rules.
  - NEH is clearly dominant in comparison to some dispatching rules with effective machine assignments. This is an interesting result. NEH is an important heuristic applicable to complex production environments, not only to regular flowshop problems.

- Statistical analysis of heuristic results indicate:
  - The most influential factor is NP_j, the second being the type of algorithm, and the third one being the interaction between these two factors.
  - Adding predecessors results in instances that are easier to solve by the dispatching rules. The differences in performance among the heuristics are much more pronounced when there are no precedence constraints. For the modified NEH, predecessors result in harder instances.
  - Similar to the case with MIP model, instances with more machines per stage are easier to solve.

Overall, it can be concluded that factors that usually are regarded as important, like the setup to processing time ratio, the possibility of setups to be non/anticipatory or to a lesser extent the distribution of the time lags or the release dates for machines have no effect on the difficulty as regards the MIP model and the heuristics.

Although in this paper a highly realistic and complex hybrid flexible flowshop problem has been considered, there is still a number of assumptions. For example, it is assumed that there are no errors when processing the jobs on machines and no breakdowns nor wearing on the machines. An unlimited buffer capacity between machines is assumed. Other possible situations like recirculation, preemption and others have not been tackled. All data used in the considered

HFFL is deterministic and known in advance. Apart from developing new solution techniques, more work is needed in order to further close the gap between the theory and practice of scheduling. We are currently working on the problem using metaheuristics and on other more practical optimization criteria involving due dates.

Acknowledgments

Rubén Ruiz is partly funded by the Spanish Department of Education and Science under the research project ref. DPI2004-06366-C03-01 financed by FEDER funds. The authors would like to thank the anonymous referees for the insightful comments that have helped improve this work.

References

[1] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1954;1(1): 61–8.
[2] Graves SC. A review of production scheduling. Operations Research 1981;29(4):646–75.
[3] Ledbetter WN, Cox JF. Operations research in production management: an investigation of past and present utilization. Production and Inventory Management 1977;18(3):84–91.
[4] Ford FN, Bradbard DA, Ledbetter WN, Cox JF. Use of operations research in production management. Production and Inventory Management 1987;28(3):59–62.
[5] McKay KN, Safayeni FR, Buzacott JA. Job-shop scheduling theory: what is relevant?. Interfaces 1988;4(18):84–90.
[6] Olhager J, Rapp B. Operations research techniques in manufacturing planning and control systems. International Transactions in Operational Research 1995;2(1):29–43.
[7] Dudek RA, Panwalkar SS, Smith ML. The lessons of flowshop scheduling research. Operations Research 1992;40(1):7–13.
[8] MacCarthy BL, Liu J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. International Journal of Production Research 1993;31(1):59–79.
[9] McKay KN, Pinedo M, Webster S. Practice-focused research issues for scheduling systems. Production and Operations Management 2002;11(2):249–58.
[10] Allahverdi A, Gupta JND, Aldowaisan T. A review of scheduling research involving setup considerations. OMEGA, The International Journal of Management Science 1999;27(2):219–39.
[11] Linn R, Zhang W. Hybrid flow shop scheduling: a survey. Computers & Industrial Engineering 1999;37(1–2):57–61.
[12] Vignier A, Billaut JC, Proust C. Les problèmes d'ordonnancement de type flow-shop hybride: État de l'art. RAIRO Recherche opérationnelle 1999;33(2):117–83 (in French).
[13] Reisman A, Kumar A, Motwani J. Flowshop scheduling/sequencing research: a statistical review of the literature, 1952–1994. IEEE Transactions on Engineering Management 1997;44(3):316–29.
[14] Oduguwa V, Tiwari A, Roy R. Evolutionary computing in manufacturing industry: an overview of recent applications. Applied Soft Computing 2005;5(3):281–99.
[15] Parthasarathy S, Rajendran C. An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs. International Journal of Production Economics 1997;49(3):255–63.
[16] Pearn WL, Chung SH, Chen AY, Yang MH. A case study on the multistage IC final testing scheduling problem with reentry. International Journal of Production Economics 2004;88(3):257–67.
[17] Bertel S, Billaut JC. A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. European Journal of Operational Research 2004;159(3):651–62.
[18] Tanev IT, Uozumi T, Morotome Y. Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach. Applied Soft Computing 2004;5(1):87–100.
[19] Andrés C, Albarracín JM, Tormo G, Vicens E, García-Sabater JP. Group technology in a hybrid flowshop environment: a case study. European Journal of Operational Research 2005;167(1):272–81.
[20] Kochhar S, Morris R, Wong W. The local search approach to flexible flow line scheduling. Engineering Costs and Production Economics 1988;14(1):25–37.
[21] Schutten JMJ. Practical job shop scheduling. Annals of Operations Research 1998;83:161–78.
[22] Botta-Genoulaz V. Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. International Journal of Production Economics 2000;64(1–3):101–11.
[23] Kurz ME, Askin RG. Comparing scheduling rules for flexible flow lines. International Journal of Production Economics 2003;85(3):371–88.
[24] Kurz ME, Askin RG. Scheduling flexible flow lines with sequence-dependent setup times. European Journal of Operational Research 2004;159(1):66–82.
[25] Fondrevelle J, Oulamara A, Portmann M-C. Permutation flowshop scheduling problems with maximal and minimal time lags. Computers & Operations Research 2006;33(6):1540–56.
[26] Ruiz R, Maroto C. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 2006;169(3):781–800.
[27] Gourgand M, Grangeon N, Norre S. Metaheuristics for the deterministic hybrid flow shop problem. In: Proceedings of the international conference on industrial engineering and production management, IEPM'99. Glasgow, FUCAM, INRIA, 1999. p. 136–45.

[28] Gupta JND. Two-stage, hybrid flowshop scheduling problem. Journal of the Operational Research Society 1988;39(4):359–64.
[29] Lee C-Y, Vairaktarakis GL. Minimizing makespan in hybrid flowshops. Operations Research Letters 1994;16(3):149–58.
[30] Pinedo M. Scheduling: theory algorithms and systems. 2nd ed., New Jersey: Prentice-Hall; 2002.
[31] Gupta JND. Flowshop schedules with sequence dependent setup times. Journal of the Operational Research Society of Japan 1986;29(3): 206–19.
[32] Brah SA, Hunsucker JL. Branch and bound algorithm for the flow shop with multiple processors. European Journal of Operational Research 1991;51(1):88–99.
[33] Rajendran C, Chaudhuri D. A multi-stage parallel-processor flowshop problem with minimum flowtime. European Journal of Operational Research 1992;57(1):111–22.
[34] Santos DL, Hunsucker JL, Deal DE. Global lower bounds for flow shops with multiple processors. European Journal of Operational Research 1995;80(1):112–20.
[35] Sawik T. Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. Mathematical and Computer Modelling 2000;31(13):39–52.
[36] Nawaz M, Enscore Jr EE, Ham I. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. OMEGA, The International Journal of Management Science 1983;11(1):91–5.
[37] Stafford Jr EF, Tseng FT. Two models for a family of flowshop sequencing problems. European Journal of Operational Research 2002;142(2): 282–93.
[38] Ruiz R, Maroto C, Alcaraz J. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. European Journal of Operational Research 2005;165(1):34–54.
[39] Morgan JA, Sonquist JN. Problems in the analysis of survey data and a proposal. Journal of the American Statistical Association 1963;58: 415–34.
[40] Kass GV. An exploratory technique for investigating large quantities of categorical data. Applied Statistics 1980;29(2):119–27.
[41] Biggs D, De Ville B, Suen E. A method of choosing multiway partitions for classification and decision trees. Journal of Applied Statistics 1991;18(1):49–62.
[42] McNemar Q. Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 1947;12:153–7.