Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective*

Victor Fernandez-Viagas[1]†, Paz Perez-Gonzalez[1], Jose M. Framinan[1]

[1] Industrial Management, School of Engineering, University of Seville,

Camino de los Descubrimientos s/n, 41092 Seville, Spain, {vfernandezviagas,pazperez,framinan}@us.es

January 13, 2020

Abstract

In this paper we address the classical hybrid flow shop scheduling problem with makespan objective. As this problem is known to be NP-hard and a very common layout in real-life manufacturing scenarios, many studies have been proposed in the literature to solve it. These contributions use different solution representations of the feasible schedules, each one with its own advantages and disadvantages. Some of them do not guarantee that all feasible semiactive schedules are represented in the space of solutions –thus limiting in principle their effectiveness– but, on the other hand, these simpler solution representations possess clear advantages in terms of having consistent neighbourhoods with well-defined neighbourhood moves. Therefore, there is a trade-off between the solution space reduction and the ability to conduct an efficient search in this reduced solution space. This trade-off is determined by two aspects, i.e. the extent of the solution space reduction, and the quality of the schedules left aside by this solution space reduction. In this paper, we analyse the efficiency of the different solution representations employed in the literature for the problem. More specifically, we first establish the size of the space of semiactive schedules achieved by the different solution representations and, secondly, we address the issue of the quality of the schedules that can be achieved by these representations using the optimal solutions given by several MILP models and complete enumeration. The results obtained may contribute to design more efficient algorithms for the hybrid flow shop scheduling problem.

Keywords: Scheduling, Hybrid Flow shop, MILP, complete enumeration, Makespan, solution representation, review, flowshop

*Preprint submitted to Computers & Operational Research. DOI:10.1016/j.cor.2019.05.002

†Corresponding author. Email: vfernandezviagas@us.es

1 Introduction

Flow shop scheduling is one of the most studied problems in Operations Research (see e.g. the reviews by Framinan et al., 2004; Ruiz and Maroto, 2005; Fernandez-Viagas et al., 2017). In this setting, n jobs are processed across m single-machine stages, where each job follows the same route of stages. However, the flow shop layout may result in overloading some stages (Fernandez-Viagas and Framinan, 2017) so, in many real-life scenarios, parallel machines –usually assumed to be identical– are placed in these stages to increase the throughput and to balance the workload (Naderi et al., 2010). Scheduling in such flow shop layout with parallel machines in each stage is traditionally labelled as the Hybrid Flow shop Scheduling (HFS) problem. Given its practical interest, it is not surprising that the HFS problem has been widely studied by researchers (see e.g. the reviews by Ruiz and Vázquez-Rodríguez, 2010 or Ribas et al., 2010).

As the aforementioned reviews can attest, the most employed objective is the minimisation of the maximum completion time of the jobs (makespan). Since the HFS problem with makespan objective is known to be NP-hard for two stages and at least one stage with two machines (Gupta, 1988), or for a single-stage with three or more machines (Rinnooy Kan, 1976), many approximate algorithms have been proposed to address it (see e.g. Fernandez-Viagas et al., 2018a; Dios et al., 2018; Chung et al., 2017; Zhong and Shi, 2018 and Ying and Lin, 2018 for the most recent contributions). These algorithms usually represent a semiactive feasible schedule in the HFS problem in the form of one or several sequences of jobs across machines, in some cases with a policy to assign the jobs to the machines in the stages.

While it is clear that every feasible semiactive schedule in the HFS problem can be represented unequivocally e.g. by giving the sequence in which each job is processed on each machine for each stage, many contributions in the HFS problem literature employ solution representations that do not guarantee that all feasible semiactive schedules of the HFS problem can be represented. For instance, some works restrict the space of solutions in the HFS problem to semiactive schedules that can be represented by a sequence of jobs per stage plus a given policy to assign jobs to the machines in each stage (e.g. to assign the jobs to the first available machine in the stage). Clearly, this type of solution representation limits the search of the best schedule to a subset of schedules

within the HFS problem solution space, which, in principle, would reduce the effectiveness of this representation to obtain high-quality solutions. On the other hand, using a simpler solution representation has obvious advantages in terms of having consistent neighbourhoods with well-defined neighbourhood moves, so there is a trade-off between the solution space reduction and the ability to conduct an efficient search in this reduced solution space. This trade-off is determined by two aspects, i.e. the extent of the solution space reduction, and the quality of the schedules left aside by this solution space reduction. Clearly, if a solution representation achieves a large reduction of the solution space without leaving outside most of the high-quality schedules, then this solution representation can be considered as efficient. Conversely, solution representations that do not substantially reduce the solution space of the HFS problem *or* leave outside high-quality schedules, cannot be regarded as efficient. Despite the importance of the representation of the solutions in the performance of both approximate and exact approaches, to the best of our knowledge, no analysis has been performed so far in the literature and the following important research questions are still open:

- Which representation of the solutions, job sequencing rules and machine assignments have been addressed in the literature?
- How far from the optimum we are if we use simpler representation of the solutions?
- How much does the different job sequencing rules and machine assignments influence in the quality of solutions?
- Is it possible to improve the results by combining different solution representations?
- How are the solutions of each representation distributed (i.e. number/percentage of solutions within a given distance of the best/worst solution of the problem)?

To tackle these challenges, in this paper, we analyse the efficiency of the solution representations for the HFS problem with makespan objective. More specifically, we first establish the size of the space of semiactive schedules achieved by the different solution representations employed in the literature, together with another possible representation that, to the best of our knowledge, it has not been employed so far. Secondly, we address the issue of the quality of the schedules

that can be achieved by the different solution representations. This is done via computational evaluation of the best results obtained by the different solution representations employed in the literature. The results obtained may contribute to design more efficient algorithms for the HFS problem.

The remainder of the paper is organised as follows: In Section 2 we give the formal description of the problem, while in Section 3 we classify the different solution representations employed in the literature, together with their corresponding assignment and sequencing rules. We then define four subproblems representing the solution spaces of the most widely-employed representations. The size of the solution spaces of these subproblems is addressed in Section 4, while in Section 5 we propose four mixed integer linear programming (MILP) models to solve them. In Section 6 the quality of the solution of these subproblems and related solution representations is evaluated. Finally, the conclusions are discussed in Section 7.

2 Problem description and notation

In a HFS problem, there is a set $\mathcal{N}$ with n jobs and a set $\mathcal{M}$ of m stages, where each stage i is composed of m_i identical parallel machines, $\forall i \in \mathcal{M}$. Let p_{ij} be the processing time of job j in stage i. Due to technological constraints, the processing of each job across the stages cannot be altered, so all jobs follow the same order across stages. The HFS problem then consists in finding a schedule (i.e. a set of starting times for each job on each machine) that minimises the makespan or maximum completion time among the jobs, i.e. C_{max}. This problem is usually denoted as $FHm, ((PM^k)_{k=1}^m)||C_{max}$ by Ruiz and Vázquez-Rodríguez (2010), and by $HFm||C_{max}$ or $FFm||C_{max}$ by Graham et al. (1979).

Among the feasible schedules that could be obtained for the problem, we will focus on semi-active schedules (see e.g. Pinedo, 1995 for a formal definition), as it is clear that right-shifting some jobs in the Gantt chart would not improve the makespan. Given a (semiactive) schedule, let us denote by C_{ij} the completion time of job j in stage i, by C_i the maximum completion time in stage i and C_j the completion time of job j in the last stage. Consequently, the makespan can be defined as follows: $C_{max} = \max_i C_i = \max_j C_j = \max_{ij} C_{ij}$.

3 Background: Solution Representation

Different solution representations for the HFS problem have been used in the literature. To identify and classify them, we have reviewed the main contributions on the topic. More specifically, we started from the reviews by Ruiz and Vázquez-Rodríguez (2010); Ribas et al. (2010), and conducted an exhaustive review on the Scopus database, not only for the problem under consideration, but also for more constrained HFS problems when the representation of the solution is not influenced by the constraints as e.g. pre-emption. From this review, the following solution representations have been identified:

- $\mathcal{R}_1$: Here we group solution representations that contain all feasible semiactive schedules of the problem. These are:
  - $\mathcal{R}_1^S$, where a solution is represented by $\sum_{i=1}^{m} m_i$ sequences of jobs, each sequence giving the order in which the jobs are processed on each machine in the shop. $\mathcal{R}_1^S$ is used by e.g. Nowicki and Smutnicki (1998); Belkadi et al. (2006). It is clear that one solution in $\mathcal{R}_1^S$ unequivocally defines a feasible semiactive schedule in a HFS problem, and that $\mathcal{R}_1^S$ contains all feasible semiactive schedules.
  - $\mathcal{R}_1^R$, where a solution is represented by a decimal number for each job in each stage. The integer part of this decimal number indicates the machine where the job is assigned, while the sequence of the jobs in each stage is obtained by ordering the fractional parts of the decimal numbers of that stage. $\mathcal{R}_1^R$ is used e.g. by Buddala and Mahapatra (2018); Niu et al. (2009); Naderi et al. (2009). It is also clear that one solution in $\mathcal{R}_1^R$ unequivocally defines a feasible semiactive schedule, and that $\mathcal{R}_1^R$ contains all feasible semiactive schedules.

- $\mathcal{R}_2$: Here we classify solution representations that do not unequivocally define a feasible semiactive schedule in the HFS problem, but to do so, some assignment rule of jobs to machines (denoted as $\mathcal{D}^{\mathcal{A}}$ in the following) has to be specified. In $\mathcal{R}_2$, a solution is typically represented by a set of m sequences, each one composed of n jobs. Each sequence represents the order of the jobs on each stage $i \in \{1, m\}$, i.e. a job j must start its processing in

stage i not later than job k if job j precedes job k in the sequence of stage i. This solution representation is used e.g. by Fernandez-Viagas et al. (2018a); Kouvelis and Vairaktarakis (1998); Koulamas and Kyparisis (2000); Vairaktarakis and Elhafsi (2000); Lee and Vairaktarakis (1994).

Clearly, a semiactive feasible schedule is defined by the combination of one solution in $\mathcal{R}_2$ plus a given assignment rule $\mathcal{D}^{\mathcal{A}}$, so we denote as $\mathcal{R}_2(\mathcal{D}^A)$ the so-obtained set of feasible schedules. However, not every feasible semiactive schedule in the HFS problem may be represented in this way.

- $\mathcal{R}_3$: Here we classify solution representations that require a specific rule to sequence the jobs between stages (denoted in the following as $\mathcal{D}^{\mathcal{S}}$), in order to unequivocally define a feasible schedule in the HFS problem. A solution is typically represented by a unique sequence of jobs and the machine assignment of each job in each stage. A semiactive feasible schedule is therefore defined by the combination of $\mathcal{R}_3$ plus a given schedule rule $\mathcal{D}^{\mathcal{S}}$, so we denote as $\mathcal{R}_3(\mathcal{D}^S)$ the so-obtained set of feasible schedules. Note that, as with $\mathcal{R}_2$, not every feasible semiactive schedule in the HFS problem may be represented in this way. Although we are not aware that this representation has been used to solve the problem under consideration, we include it and test its efficiency in Section 6.

- $\mathcal{R}_4$: Here we classify solution representations that, in order to unequivocally define a feasible schedule in the HFS problem, require a given assignment rule of jobs to machines ($\mathcal{D}^{\mathcal{A}}$) *and* a rule to sequence the jobs between stages ($\mathcal{D}^{\mathcal{S}}$). These solution representations are:

  - $\mathcal{R}_4^F$, where a solution is represented by a single sequence of n jobs that denote the order of the jobs in the first stage (forward approach, see Soewandi and Elmaghraby, 2001; Pan et al., 2014; Cui and Gu, 2014 for some examples).

  - $\mathcal{R}_4^B$, where a sequence is given to represent the inverse sequence of jobs (beginning with the last stage and finishing with the first one) (backward approach, see e.g. Wang et al., 2013; Pan et al., 2014; Xu et al., 2013).

Clearly, a semiactive feasible schedule is defined by one solution in $\mathcal{R}_4$ plus a machine assignment rule $\mathcal{D}^{\mathcal{A}}$ and a job sequencing rule $\mathcal{D}^{\mathcal{S}}$, so we denote as $\mathcal{R}_4(\mathcal{D}^A, \mathcal{D}^{\mathcal{S}})$ the so-obtained set of feasible schedules. As in $\mathcal{R}_2$ and $\mathcal{R}_3$, not every feasible schedule can be defined by a combination of a solution in $\mathcal{R}_4$ plus some given machine assignment and job sequencing rules.

Regarding the job sequencing rules ($\mathcal{D}^S$) found in the literature, we note the following:

- FIFO (see e.g. Brah and Loo, 1999; Liao et al., 2012; Pan et al., 2014; Xu et al., 2013; Lahimer et al., 2013; Oguz et al., 2004). This rule sorts the jobs in stage i (with $i > 1$) in non-decreasing order of their completion times in stage $i-1$. Since ties may occur (i.e. two jobs have the same completion time on the previous stage) and no tie-breaking mechanism is described in these references, we assume that ties are broken taken the first tie found.

- FIFO(iLS) (Wang et al., 2013). Similarly to the FIFO rule, FIFO(iLS) sorts jobs in stage i according to non-decreasing completion times in stage $i-1$. However, in case of ties, the jobs with higher remaining processing times are chosen first (i.e. sum of the processing times $\forall i^{'} > i$).

- Q(SPT) (Allahverdi and Al-Anzi, 2006; Barman, 1997; Brah and Wheeler, 1998; Jayamohan and Rajendran, 2000; Han et al., 2018; Fernandez-Viagas et al., 2018a; Hunsucker and Shah, 1992). Jobs are first ordered in each stage according to non-decreasing order of their completion times in the previous stage. After that, when a job should be placed in any machine of a stage, the job with shortest processing time (SPT) among the jobs in the queue (in case that there are more jobs which could be placed in that time) is chosen. Conversely, let Q(LPT) denote when the job with the largest processing time (LPT) is chosen (Brah and Wheeler, 1998; Fernandez-Viagas et al., 2018a; Hunsucker and Shah, 1992).

- Q(PT+WINQ+AT) (Holthaus and Rajendran, 1997; Jayamohan and Rajendran, 2000). Analogously to the Q(SPT) rule, this rule chooses the job in the queue of a stage according to the PT+WINQ+AT rule (for the problem under consideration, this rule sorts the jobs

according to non-decreasing sum of the processing time in that stage and the completion time in the previous stage).

- Q(rand) (Soewandi and Elmaghraby, 2001; Brah and Wheeler, 1998; Fernandez-Viagas et al., 2018a). In this case, a random job in the queue of a stage is chosen when a job should be placed in any machine of that stage.

- Q(MWRF) and Q(LWRF) (Brah and Wheeler, 1998; Hunsucker and Shah, 1992). Similarly to the previous rule, in this case the job in the queue with the most and least processing times remaining is selected for Q(MWRF) and Q(LWRF), respectively.

- Q(MTWF) and Q(LTWF) (Brah and Wheeler, 1998). In this case, the job with the most and least total processing times in the shop is selected for Q(MTWF) and Q(LTWF), respectively.

- Q(LIFO) (Brah and Wheeler, 1998; Hunsucker and Shah, 1992). In this rule, the job to be inserted in a machine is selected from the queue according to the LIFO rule.

- PERM (see e.g. Wang et al., 2013; Xu et al., 2013) This rule uses the same sequence of jobs in each stage, which is typically applied for the flow shop layout (i.e. with $m_i = 1\ \forall i$).

Regarding the assignment of jobs to machines ($\mathcal{D}^A$):

- FAM: Jobs are assigned to the machines following the First Available Machine (FAM rule). It is the most common assignment used in the literature (see e.g. Acero-Dominguez and Paternina-Arboleda, 2004; Brah, 1996; Brah and Loo, 1999; Soewandi and Elmaghraby, 2001; Paternina-Arboleda et al., 2008; Pan et al., 2014; Oguz et al., 2004)[1].

- iPS (Jin et al., 2006; Wang et al., 2013; Xu et al., 2013): Specifically designed for the $\mathcal{R}_4^F$ representation using the same sequence in each machines (i.e. using PERM rule), jobs are assigned to the first machine that becomes available. In case of ties, the rule chooses the machine with the lowest idle time after the new inserted job (if there is still some tie, the machine with the lowest idle time before the new inserted job is chosen).

[1]Note that a rule that assigns a job to the machine which would finish that job first is tantamount to FAM, as identical parallel machines are considered.

- LBM: This rule is typically used to assign jobs to the second stage in 2-stage hybrid flow shops. Jobs are assigned to the stages following the Last Busy Machine rule (LBM) (see e.g. Kouvelis and Vairaktarakis, 1998; Koulamas and Kyparisis, 2000; Vairaktarakis and Elhafsi, 2000; Lee and Vairaktarakis, 1994). Since this rule can only be used for a 2-stage HFS problem, it would be not considered further.

Table 1 summarises the review on solution representations. As already discussed, the optimal solution for the HFS problem can be only guaranteed by using the $\mathcal{R}_1$ type of solution representation. However, the solution space in this representation is much larger than that of $\mathcal{R}_4$ (see Section 4), being this fact one of the reasons for which $\mathcal{R}_4$ has been widely used. Since most references employ $\mathcal{R}_4$ combined with some variant of FIFO for job sequencing and FAM for machine assignment, it would be of interest to analyse other spaces of solutions that lie between $\mathcal{R}_1$ (where no assignment and no sequencing rules are used) and using $\mathcal{R}_4$ with FIFO and FAM. More specifically, we define the following subproblems:

- P1 denotes the problem of obtaining the solution with the lowest makespan within the space of solutions defined by $\mathcal{R}_1$. Clearly, solving P1 is tantamount to obtain the best feasible semiactive schedule for the HFS problem.

- P2 denotes the problem of obtaining the solution with the lowest makespan within the space of solutions defined by $\mathcal{R}_2(FAM)$. In P2, no job sequencing rule is given and, clearly, the optimal solution of P2 is a (possibly non optimal) solution of P1. This subproblem contains all solutions in P1 where jobs are assigned using the most-used machine assignment rule (FAM).

- P3 denotes the problem of obtaining the solution with the lowest makespan within the space of solutions defined by $\mathcal{R}_3$ and using FIFO as job sequencing rule. So this subproblem contains all solutions in P1 where the jobs are sequenced according to the most-employed job sequencing rule (FIFO). Furthermore, it has been mentioned that several tie-breaking rules and variants of FIFO have been used in the literature. In order to explore the efficiency of this approach regardless the tie-breaking rule employed, we consider all the

Reference	$\mathcal{R}$	$\mathcal{D}^S$	$\mathcal{D}^A$
Niu et al. (2009) Buddala and Mahapatra (2018) Naderi et al. (2009) Su et al. (2014)	$\mathcal{R}_1^R$	—	—
Nowicki and Smutnicki (1998) Belkadi et al. (2006)	$\mathcal{R}_1^S$	—	—
Fernandez-Viagas et al. (2018a) Acero-Dominguez and Paternina-Arboleda (2004) Paternina-Arboleda et al. (2008)	$\mathcal{R}_2$	—	FAM
Kouvelis and Vairaktarakis (1998) Koulamas and Kyparisis (2000) Vairaktarakis and Elhafsi (2000) Lee and Vairaktarakis (1994)	$\mathcal{R}_2$	—	FAM and LBM
Hunsucker and Shah (1992) Brah and Loo (1999) Cui and Gu (2014) Dios et al. (2018) Fernandez-Viagas et al. (2018a) Lahimer et al. (2013) Oguz et al. (2004) Oguz and Ercan (2005) Jouglet et al. (2009) Liao et al. (2012) Xu et al. (2013) Pan et al. (2017) Santos et al. (1996) Serifoglu and Ulusoy (2004) Brah and Wheeler (1998) Pan et al. (2014)	$\mathcal{R}_4^F$	FIFO	FAM
Brah and Wheeler (1998) Fernandez-Viagas et al. (2018a) Soewandi and Elmaghraby (2001)	$\mathcal{R}_4^F$	Q(rand)	FAM
Hunsucker and Shah (1992) Jayamohan and Rajendran (2000) Han et al. (2018) Brah and Wheeler (1998) Allahverdi and Al-Anzi (2006) Fernandez-Viagas et al. (2018a) Barman (1997)	$\mathcal{R}_4^F$	Q(SPT)	FAM
Hunsucker and Shah (1992) Brah and Wheeler (1998) Fernandez-Viagas et al. (2018a)	$\mathcal{R}_4^F$	Q(LPT)	FAM
Holthaus and Rajendran (1997) Jayamohan and Rajendran (2000) Fernandez-Viagas et al. (2018a)	$\mathcal{R}_4^F$	Q(PT+WINQ+AT)	FAM
Wang et al. (2013)	$\mathcal{R}_4^F$	FIFO(iLS)	FAM
Hunsucker and Shah (1992) Brah and Wheeler (1998)	$\mathcal{R}_4^F$	Q(MWRF)	FAM
Hunsucker and Shah (1992) Brah and Wheeler (1998)	$\mathcal{R}_4^F$	Q(LWRF)	FAM
Brah and Wheeler (1998)	$\mathcal{R}_4^F$	Q(MTWF)	FAM
Brah and Wheeler (1998)	$\mathcal{R}_4^F$	Q(LTWF)	FAM
Wang et al. (2013) Xu et al. (2013)	$\mathcal{R}_4^F$	PERM	iPS
Pan et al. (2014) Xu et al. (2013) Wang et al. (2013)	$\mathcal{R}_4^B$	FIFO	FAM

Table 1: Summary of solution representations and assignment/sequencing rules of Section 3

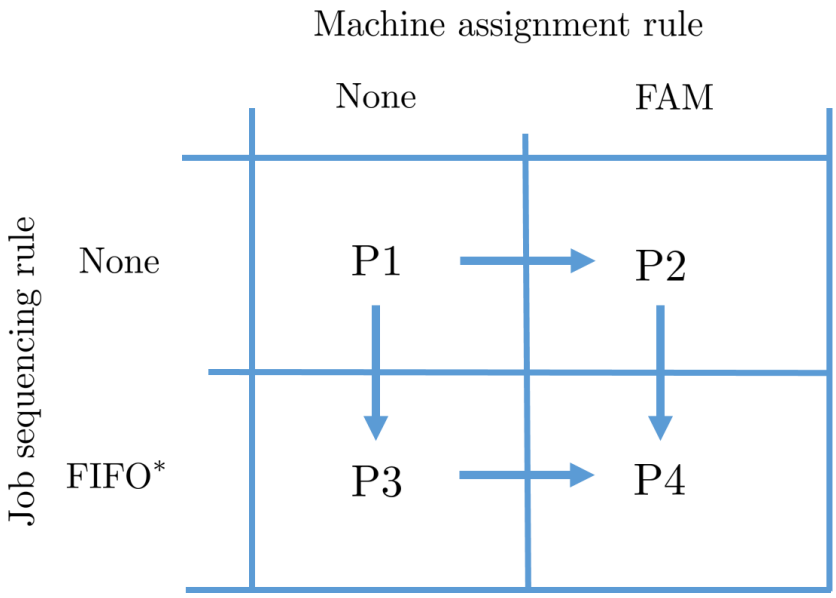

Figure 1: Different subproblems identified.

solutions that can be obtained in case of ties. We denote this job sequencing rule as FIFO* and then, the space of solutions is denoted as $\mathcal{R}_3(FIFO^*)$.

- P4 denotes the problem of obtaining the solution with the lowest makespan within the space of solutions defined by $\mathcal{R}_4^F(FAM, FIFO^*)$, which are the most widely employed machine assignment and job sequencing rules.

The relationship between the subproblems is illustrated in Figure 1. For these subproblems, we:

1. Determine the size of their space of solutions, which is addressed in Section 4. By doing so, we formally state the size of the solution space for each subproblem.

2. Determine the structure of solutions by using the complete enumeration of all solutions. With this analysis, we obtain the distribution of solutions of each problem and assess how easy or difficult is to find good solutions there.

3. Determine the quality of the optimal solution for each of these subproblems with respect to the optimal solution of the HFS problem. To do so, we develop a MILP model for each one of these problems, and use them to solve small and medium instances. The MILP models

are described in Section 5. Furthermore, we explore additional solution spaces that can be constructed using other sequencing and assignment rules to fully assess the efficiency of the different solution representations using complete enumeration. This experimentation is carried out in Section 6.

4 Structure of the solutions: Size of the solution space and complete enumeration

In this section we state the size of the solution space of the four subproblems identified and explain the methodology used to completely define the structure of the subproblems. Regarding the former, the number of total solutions for each subproblem can be defined as follows:

- Subproblem P1. The size of the solution space of P1 is given by the following expression (see Urlings et al., 2010):

$$|P1| = (n!)^m \prod_{i=1}^{m} \binom{n + m_i - 1}{m_i - 1} \tag{1}$$

- Subproblem P2. The size of the solution space of P2 is given by

$$|P2| = (n!)^m \tag{2}$$

  since a sequence of n jobs has to be specified for each stage i.

- Subproblem P3. The size of the solution space of P3 is given by

$$|P3| = (n!) \prod_{i=1}^{m} m_i^n \tag{3}$$

  since each job of the sequence can be assigned to any of the m_i machines of stage i (all possible machine assignment).

- Subproblem P4. The size of the solution space of P4 is given by

$$|P4| = n! \tag{4}$$

since only a sequence of jobs is considered.

Regarding the methodology to evaluate the quality of the solutions of these subproblems, we use complete enumeration of the solutions. This methodology has been commonly used in the literature to analyse the structure of the solutions and/or compare different subproblems (see e.g. Perez-Gonzalez and Framinan, 2009; Fernandez-Viagas and Framinan, 2017; Dios et al., 2018). Basically, the methodology generates all solutions of the problem evaluating their objective functions. In a first iteration, all solutions are run searching the best and worst solution. Once these solutions are identified, specific indicators of the structure of the solutions can be obtained in a second iteration: frequency curve, ARPD of all solutions, etc (see Section 6.2).

5 MILP models

In this section, the subproblems defined in Section 3 are modelled as Mixed Integer Linear Programming (MILP) models. More specifically, the following four MILPs model are stated and described in this section to find the optimal solutions of each one of the previously defined subproblems:

1. Model 1: MILP model which obtains the optimal solution of P1, i.e. in the space of solutions of $\mathcal{R}_1$.

2. Model 2: MILP model which obtains the optimal solution of P2, i.e. in the space of solutions of $\mathcal{R}_2(FAM)$. With this model, we enforce that the FAM rule is applied to assign jobs to machines.

3. Model 3: MILP model which obtains the optimal solution of P3, i.e. in the space of solutions of $\mathcal{R}_3(FIFO^*)$. In this case, the model sequences the jobs between stages using the $FIFO^*$ rule and analyse every possible assignment of jobs to machines.

4. Model 4: MILP model which obtains the optimal solution of P4, i.e. in the space of solutions of $\mathcal{R}_4(FAM, FIFO^*)$. The model then incorporates both FAM and $FIFO^*$ rules.

The four models use the following common variables:

- C_{ij} continuous variable which indicates the completion time of job j at stage i.
- X_{ijk} binary variable equals 1 if job k is processed before job j at stage i, and 0 otherwise.
- Y_{ilj} binary variable equals 1 if job j is processed at stage i on machine l, and 0 otherwise.
- C_{max} continuous variable which indicates the makespan.

5.1 Model 1 to solve P1 (Naderi et al., 2014)

$$\text{minimize } C_{max}$$

$$\text{subject to } \sum_{l \in \mathcal{M}_i} Y_{ilj} = 1 \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M1.1)$$

$$C_{ij} \geq C_{i-1,j} + p_{ij} \qquad i \in \mathcal{M} - \{1\}, j \in \mathcal{N} \qquad (M1.2)$$

$$C_{ij} \geq C_{ik} + p_{ij} - M \cdot (3 - X_{ijk} - Y_{ilj} - Y_{ilk}) \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} | k > j \qquad (M1.3)$$

$$C_{ik} \geq C_{ij} + p_{ik} - M \cdot X_{ijk} - M \cdot (2 - Y_{ilj} - Y_{ilk}) \quad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} | k > j \qquad (M1.4)$$

$$C_{max} \geq C_{mj} \qquad j \in \mathcal{N} \qquad (M1.5)$$

$$C_{ij} \geq 0 \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M1.6)$$

$$X_{ijk} \in \{0, 1\} \qquad i \in \mathcal{M}, j \in \mathcal{N}, k \in \mathcal{N} | k > j \qquad (M1.7)$$

$$Y_{ilj} \in \{0, 1\} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N} \qquad (M1.8)$$

Equations (M1.1) enforce that each job is processed only in one machine on each stage. The completion times of each job are defined in constraints (M1.2), (M1.3), and (M1.4): On the one hand, set of constraints (M1.2) ensures that the ith operation of a job starts to be processed after its previous operation; on the other hand, constraints (M1.3) and (M1.4) assure that it is

not possible to process two jobs in the same machine at the same time, and that a job cannot be processed before the completion time of each previous job at the same machine. M is a large number. Constraints set (M1.5) defines the makespan. Finally, sets of constraints (M1.6), (M1.7), and (M1.8) define the variables of the model.

5.2 Model 2 to solve P2

$$\text{minimize } C_{max}$$

$$\text{subject to } \sum_{l \in \mathcal{M}_i} Y_{ilj} = 1 \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M2.1)$$

$$C_{ij} = C_{i-1,j} + p_{ij} + h_{ij} \qquad i \in \mathcal{M} - \{1\}, j \in \mathcal{N} \qquad (M2.2)$$

$$C_{ij} = C_{ik} + p_{ij} - M \cdot (3 - X_{ijk} - Y_{ilj} - Y_{ilk}) + h'_{iljk} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M2.3)$$

$$X_{ijk} + X_{ikj} = 1 \qquad i \in \mathcal{M}, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M2.4)$$

$$h_{ij} \leq M \cdot V_{ij} \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M2.5)$$

$$h'_{iljk} + M \cdot (3 - X_{ijk} - Y_{ilj} - Y_{ilk}) \leq M' \cdot (1 - V'_{iljk}) \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M2.6)$$

$$\sum_{l \in \mathcal{M}_i} \sum_{k \in \mathcal{N} - \{j\}} V'_{iljk} \geq 1 - M \cdot (1 - V_{ij}) \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M2.7)$$

$$C_{ij} - p_{ij} + M \cdot (1 - X_{ijk}) \geq C_{ik} - p_{ik} \qquad i \in \mathcal{M}, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M2.8)$$

$$C_{max} \geq C_{mj} \qquad j \in \mathcal{N} \qquad (M2.9)$$

$$S_{ilj} + M \cdot (2 - X_{ijk} - Y_{ilk}) = C_{ik} + h''_{iljk} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in (\mathcal{N} \cup dummy) - \{j\} \qquad (M2.10)$$

$$h''_{iljk} \leq M' \cdot V''_{iljk} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in (\mathcal{N} \cup dummy) - \{j\} \qquad (M2.11)$$

$$\sum_{k \in (\mathcal{N} \cup dummy) - \{j\}} V''_{iljk} = 1 \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N} \qquad (M2.12)$$

$$S_{ilj} \leq S_{irj} + M \cdot (1 - Y_{ilj}) \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, r \in \mathcal{M}_i, j \in \mathcal{N} \qquad (M2.13)$$

$$C_{ij}, h_{ij} \geq 0 \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M2.14)$$

$$X_{ijk} \in \{0, 1\} \qquad i \in \mathcal{M}, j \in \mathcal{N}, k \in (\mathcal{N} \cup dummy) - \{j\} \qquad (M2.15)$$

$$Y_{ilj} \in \{0, 1\} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N} \qquad (M2.16)$$

$$h'_{iljk}, h''_{iljk} \geq 0 \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in (\mathcal{N} \cup dummy) - \{j\} \qquad (M2.17)$$

$$V_{ij} \in \{0, 1\} \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M2.18)$$

$$V'_{iljk}, V''_{iljk} \in \{0, 1\} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in (\mathcal{N} \cup dummy) - \{j\} \qquad (M2.19)$$

As compared to Model 1, this model does not use symmetry for variable X_{ijk}. In addition, we enforce the use of semiactive sequences to represent the influence in the results of the different

sequences between stages. Otherwise, a job could be enforced to finish later and therefore, another job would be assigned according to the FAM rule. The following new variables have been added to the model:

- S_{ilj} continuous variable which indicates the completion time of each machine l before inserting job j in stage i
- h_{ij}, h'_{iljk}, h''_{iljk}, slack variables.
- V_{ij}, V'_{iljk}, V''_{iljk} binary variables which bound the previous slack variables.

The set of constraints (M2.1) and (M2.9) are the same as (M1.1) and (M1.5) of Model 1, respectively. Constraints set (M2.2) transforms the set of constraint (M1.2) of Model 1 into an equality constraint by adding a slack variable (denoted as h_{ij}). Constraints (M2.3) (equivalent to constraints (M1-3) and (M1.4) of Model 1) assure that a job cannot be processed before the completion time of each previous job at the same machine. Slack variable h'_{iljk} is introduced to transform these inequality constraints into equality constraints. Constraints set (M2.4) ensures that either job j precedes job k in machine i or job k precedes job j in that machine. Set of constraints (M2.5), (M2.6), and (M2.7) assure the considerations of only semiactive schedules. The purpose of introducing the slack variables h_{ij} and h'_{iljk} is that a job must start either exactly after its previous operation (i.e. contraints M2.2 with $h_{ij} = 0$) or exactly after the previous job in the same machine (i.e. contraints M2.2 with $h'_{iljk} = 0$ for some l and k). More specifically, either $h_{ij} = 0$ (and set M2.2 is $C_{ij} = C_{i-1,j} + p_{ij}$) or both $h'_{iljk} = 0$ and $3 - X_{ijk} - Y_{ilj} - Y_{ilk} = 0$ for some l and k (i.e. set of constraints M2.3 is $C_{ij} = C_{ik} + p_{ij}$ for at least one value of k, $\forall i, j$). By means of the binary variables V_{ij} and V'_{iljk}, the constraints (M2.7) enforce that if $h_{ij} > 0$ then $h'_{iljk} = 0$ for some l and k, and the opposite. M' is a big number bigger than M ($M' >> M$). Constraints (M2.8) assures that the starting time of job k at stage i must be lower than the starting time of job j at that stage if job k precedes job j at that stage. Set of constraints (M2.10), (M2.11), and (M2.12) fully define S_{ilj} variable, i.e. they linearise the expression $S_{ilj} = \max\{C_{ik} : k \text{ verifies } X_{ijk} = 1 \& Y_{ilk} = 1\}$. Note that an artificial job (denoted dummy) is added to be able to model a job without predecessors. Finally, constraints (M2.13)

with previous constraints (M2.8) fully model the FAM rule, they assure S_{ilj} using that a job j is assigned to machine l which firstly become available.

5.3 Model 3 to solve P3

minimize C_{max}

$$\text{subject to} \sum_{l \in \mathcal{M}_i} Y_{ilj} = 1 \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M3.1)$$

$$C_{ij} = C_{i-1,j} + p_{ij} + h_{ij} \qquad i \in \mathcal{M} - \{1\}, j \in \mathcal{N} \qquad (M3.2)$$

$$C_{ij} = C_{ik} + p_{ij} - M \cdot (3 - X_{ijk} - Y_{ilj} - Y_{ilk}) + h'_{iljk} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.3)$$

$$X_{ijk} + X_{ikj} = 1 \qquad i \in \mathcal{M}, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.4)$$

$$h_{ij} \leq M \cdot V_{ij} \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M3.5)$$

$$h'_{iljk} + M \cdot (3 - X_{ijk} - Y_{ilj} - Y_{ilk}) \leq M' \cdot (1 - V'_{iljk}) \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.6)$$

$$\sum_{l=1 \in \mathcal{M}_i} \sum_{k=1 \mathcal{N}} V'_{iljk} \geq 1 - M \cdot (1 - V_{ij}) \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M3.7)$$

$$C_{i-1,j} + M \cdot (1 - X_{ijk}) \geq C_{i-1,k} \qquad i \in \mathcal{M} - \{1\}, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.8)$$

$$C_{max} \geq C_{mj} \qquad j \in \mathcal{N} \qquad (M3.9)$$

$$C_{ij}, h_{ij} \geq 0 \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M3.10)$$

$$X_{ijk} \in \{0, 1\} \qquad i \in \mathcal{M}, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.11)$$

$$Y_{ilj} \in \{0, 1\} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N} \qquad (M3.12)$$

$$V_{ij} \in \{0, 1\} \qquad i \in \mathcal{M}, j \in \mathcal{N} \qquad (M3.13)$$

$$V'_{iljk} \in \{0, 1\} \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.14)$$

$$h'_{iljk} \geq 0 \qquad i \in \mathcal{M}, l \in \mathcal{M}_i, j \in \mathcal{N}, k \in \mathcal{N} - \{j\} \qquad (M3.15)$$

This model is identical to the first nine constraints of Model 2 with the exception that it replaces the set of constraints (M2.8) by (M3.8), to use the FIFO sequencing rule to extend sequences between stages. More specifically, this set ensures that job k is sequenced before job j at stage i if the completion time of j is greater or equal than the completion time of k at stage

$i-1$. Note that as the MILP looks for the optimal solution, $FIFO^*$ rule is guaranteed.

5.4 Model 4 to solve P4

$$\text{minimize } C_{max}$$

$$\text{subject to } \sum_{l\in\mathcal{M}_i} Y_{ilj} = 1 \qquad i\in\mathcal{M}, j\in\mathcal{N} \qquad (M4.1)$$

$$C_{ij} = C_{i-1,j} + p_{ij} + h_{ij} \qquad i\in\mathcal{M}-\{1\}, j\in\mathcal{N} \qquad (M4.2)$$

$$C_{ij} = C_{ik} + p_{ij} - M\cdot(3 - X_{ijk} - Y_{ilj} - Y_{ilk}) + h^{'}_{iljk} \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N}, k\in\mathcal{N}-\{j\} \qquad (M4.3)$$

$$X_{ijk} + X_{ikj} = 1 \qquad i\in\mathcal{M}, j\in\mathcal{N}, k\in\mathcal{N}-\{j\} \qquad (M4.4)$$

$$h_{ij} \le M\cdot V_{ij} \qquad i\in\mathcal{M}, j\in\mathcal{N} \qquad (M4.5)$$

$$h^{'}_{iljk} + M\cdot(3 - X_{ijk} - Y_{ilj} - Y_{ilk}) \le M^{'}\cdot(1 - V^{'}_{iljk}) \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N}, k\in\mathcal{N}-\{j\} \qquad (M4.6)$$

$$\sum_{l\in\mathcal{M}_i}\sum_{k\in\mathcal{N}-\{j\}} V^{'}_{iljk} \ge 1 - M\cdot(1 - V_{ij}) \qquad i\in\mathcal{M}, j\in\mathcal{N} \qquad (M4.7)$$

$$C_{ij} - p_{ij} + M\cdot(1 - X_{ijk}) \ge C_{ik} - p_{ik} \qquad i\in\mathcal{M}, j\in\mathcal{N}, k\in\mathcal{N}-\{j\} \qquad (M4.8)$$

$$C_{i-1,j} + M\cdot(1 - X_{ijk}) \ge C_{i-1,k} \qquad i\in\mathcal{M}-\{1\}, j\in\mathcal{N}, k\in\mathcal{N}-\{j\} \qquad (M4.9)$$

$$C_{max} \ge C_{mj} \qquad j\in\mathcal{N} \qquad (M4.10)$$

$$S_{ilj} + M\cdot(2 - X_{ijk} - Y_{ilk}) = C_{ik} + h^{''}_{iljk} \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N}, k\in(\mathcal{N}\cup dummy)-\{j\} \qquad (M4.11)$$

$$h^{''}_{iljk} \le M^{'}\cdot V^{''}_{iljk} \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N}, k\in(\mathcal{N}\cup dummy)-\{j\} \qquad (M4.12)$$

$$\sum_{k\in(\mathcal{N}\cup dummy)-\{j\}} V^{''}_{iljk} = 1 \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N} \qquad (M4.13)$$

$$S_{ilj} \le S_{irj} + M\cdot(1 - Y_{ilj}) \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, r\in\mathcal{M}_i, j\in\mathcal{N} \qquad (M4.14)$$

$$C_{ij}, h_{ij} \ge 0 \qquad i\in\mathcal{M}, j\in\mathcal{N} \qquad (M4.15)$$

$$X_{ijk}\in\{0,1\} \qquad i\in\mathcal{M}, j\in\mathcal{N}, k\in(\mathcal{N}\cup dummy)-\{j\} \qquad (M4.16)$$

$$Y_{ilj}\in\{0,1\} \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N} \qquad (M4.17)$$

$$h^{'}_{iljk}, h^{''}_{iljk} \ge 0 \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N}, k\in(\mathcal{N}\cup dummy)-\{j\} \qquad (M4.18)$$

$$V_{ij}\in\{0,1\} \qquad i\in\mathcal{M}, j\in\mathcal{N} \qquad (M4.19)$$

$$V^{'}_{iljk}, V^{''}_{iljk}\in\{0,1\} \qquad i\in\mathcal{M}, l\in\mathcal{M}_i, j\in\mathcal{N}, k\in(\mathcal{N}\cup dummy)-\{j\} \qquad (M4.20)$$

This model is identical to Model 2 with the exception that it adds the inequalities as (M3.8) of Model 3, now denoted (M4.9).

6 Evaluation of the quality of the solution representations

In this section, we show the computational results of our experimentation. All methods have been coded in C# under Visual Studio in an Intel Core i7-3770 with 3.4 GHz, 16 GB RAM, and with Microsoft Windows 8.1 64 bit. The instances of the problems used are detailed in Subsection 6.1. In Subsection 6.2, we analyse the structure of solutions of the problem under consideration, P1, and of the three P2, P3, and P4 reduced problems. Finally, in Subsection 6.3, computational results in small-medium set of instances are presented to compare the presented MILP models and different representations of the solutions and rules.

6.1 Sets of instances

In this section, we generate two different sets of instances, denoted as β_1, and β_2. β_1 is a set of small instances used to compare the different complete enumeration of the problem, depending of the representation of the solutions used. β_2 benchmark is a set of small-medium instances to compare the MILP models and different combinations of assignment and sequencing rules for $\mathcal{R}_4$. The composition of both benchmarks is detailed as follows:

- β_1: Regarding the number of jobs and stages, the following combinations are considered ($n \times m \in \{3 \times 2, 3 \times 3, 3 \times 4, 4 \times 2, 4 \times 3, 4 \times 4, 5 \times 2, 5 \times 3, 6 \times 2\}$). Note that a higher number of jobs and/or stages cannot be considered due to the huge number of solutions in the P1 problem (see Section 4 for more details). Regarding the number of machines per stage, three different approaches are used ($\tau \in \{0, 1, 2\}$) following a similar procedure as Fernandez-Viagas et al. (2018a).
  - $\tau = 0$: Instances with 3 machines in each stage with the exception of a unique stage with 2 machines (see e.g. Carlier and Néron, 2000; Kouvelis and Vairaktarakis, 1998 for similar approaches);
  - $\tau = 1$: Instances with 3 machines in each stage (see e.g. Naderi et al., 2009);

– $\tau = 2$: Instances with random number of machines in each stage between 1 and 3 (see e.g. Liao et al., 2012; Pan et al., 2014; Dios et al., 2018 which also uses uniform distribution to generate the number of machines).

10 instances are generated for each combination of these parameters with processing times uniformly distributed between 1 and 99. By doing so, a total of 270 instances are generated.

- β_2: This set of 360 small-medium instances is generated by combining all the values of $n \in \{6, 7, 8, 9, 10, 11\}$ and $m \in \{2, 3, 4, 5\}$. Regarding the number of machines per stage, the same procedures as in β_1 is applied ($\tau \in \{0, 1, 2\}$). For each combination of these parameters, 5 instances are generated and the processing times are generated using a uniform distribution $[1, 99]$.

6.2 Structure of the solution space

In this section, we analyse the quality of all solutions for the P1, P2, P3, and P4 problems by a complete enumeration of all solutions (see e.g. Perez-Gonzalez and Framinan, 2009; Fernandez-Viagas and Framinan, 2017; Dios et al., 2018). Regarding the size of the solution space of each subproblem, the expressions have been explained in Section 4. An example of this size of the solution space of each subproblem is performed next for the set β_1 of small instances. Table 2 shows such number of solutions. We can observe, in the example, the huge differences between the different approaches, as e.g. the number of solution of P1 and P4 in instance $n = 5$, $m = 3$, and $\tau = 1$ are 16,003,008,000 and 120, respectively. Thereby, once the difference between the size of the different subproblems have been established, we analyse the quality or structure of all these solutions for each subproblem.

n	m	τ	P1	P2	P3	P4
3	2	0	1440	36	1296	6
3	2	1	3600	36	4374	6
3	2	2	489.6	36	367.8	6
3	3	0	86400	216	34992	6
3	3	1	216000	216	118098	6
3	3	2	46872	216	20601	6
3	4	0	5184000	1296	944784	6
3	4	1	12960000	1296	3188646	6
3	4	2	1138924.8	1296	194404.2	6
4	2	0	43200	576	31104	24
4	2	1	129600	576	157464	24
4	2	2	13363.2	576	7764	24
4	3	0	15552000	13824	2519424	24
4	3	1	46656000	13824	12754584	24
4	3	2	2654208	13824	322332	24
4	4	0	5598720000	331776	204073344	24
4	4	1	16796160000	331776	1033121304	24
4	4	2	811855872	331776	24769656	24
5	2	0	1814400	14400	933120	120
5	2	1	6350400	14400	7085880	120
5	2	2	432000	14400	189696	120
5	3	0	4572288000	1728000	226748160	120
5	3	1	16003008000	1728000	1721868840	120
5	3	2	1124928000	1728000	46621008	120
6	2	0	101606400	518400	33592320	720
6	2	1	406425600	518400	382637520	720
6	2	2	173612160	518400	156423672	720

Table 2: Example of the number of solutions using β_1

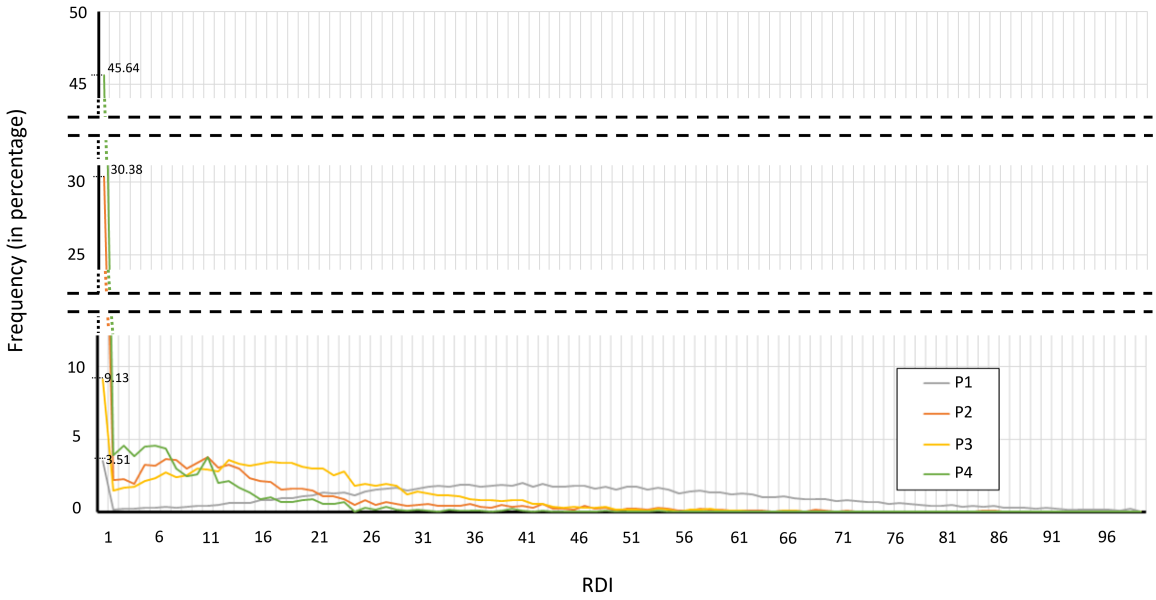

Figure 2: Structure of the solutions in average for all instances of β_1. x-axis indicates the RDI value, while y-axis indicates the number of solutions (in percentage) for this value.

The structure of solutions as an average for all instances of benchmark β_1 is shown in Figure 2. In this figure, the number of solutions is shown (in percentage) against the value of RDI indicator ($RDI = \frac{OF-Min}{Max-Min} \cdot 100$, see e.g. Fernandez-Viagas et al., 2018b; Yu et al., 2018, where Max and Min are the worst and best value found in the P1 problem, original HFS problem), i.e. we measure the distance between the best solution (value 0) and the worst solution (value 100) of P1. An empirical cumulative distribution function of these values is shown in Figure 3. From these curves, we observe that is more difficult to find a good solution in P1, then in P3, P2, and finally P4. Although, there are few solutions to explore in P4 as compared to P1, the quality of these solutions is significantly better and we find many solutions close to the optimal solution of the HFS problem. Thereby, for instance, 79.33% of the solutions are lower than 10 (i.e. an objective function, OF, which satisfies $\frac{OF-Min}{Max-Min} \cdot 100 \leq 10$) in the P4 problem, while this percentage decreases to only 5.96% and 27.62% for P1 and P3. Furthermore, more than 95% and only 13.27% of the solutions are found in 20 in P4, and P1, respectively.

The Average Relative Percentage Deviation (ARPD) using the best solution, $Best_{Pi}(\mathcal{I})$, in each instance $\mathcal{I} \in \beta_1$ –denoted as $ARPD(best)$, see Eq. (5)– and using the ARPD of all solutions –$ARPD(all)$, see Eq. (6) where OF_{Pi}^{j} refers to the C_{max} of the j-th solution of a total of T

explored solutions– are shown in Table 3, both with respect to the optimal solution provided for the P1 problem. In this table, we also show the average CPU time (in seconds) and the number of solutions T. We can observe the huge different in these values between the P1 and P4 problems (e.g. the CPU times is 0.00 s and 2105.69 s for P4 and P1, respectively). In addition, the best solution provided for P4 has an ARPD only 0.36 higher than in P1. Regarding P2, and P3, both have values of ARPD very similar to P1 analysing much less solutions (especially in the case of P2 with an average CPU time of 0.50 s as compared to 2105.69 s of P1).

$$ARPD_{Pi}(best) = \frac{\sum_{\mathcal{I}\in\beta_1} \frac{Best_{Pi}(\mathcal{I}) - Best_{P1}(\mathcal{I})}{Best_{P1}(\mathcal{I})}}{|\beta_1|} \cdot 100 \tag{5}$$

$$ARPD_{Pi}(all) = \frac{\sum_{\mathcal{I}\in\beta_1} \sum_{j=0}^{T} \frac{OF_{Pi}^{j}(\mathcal{I}) - Best_{P1}(\mathcal{I})}{Best_{P1}(\mathcal{I})}}{|\beta_1| \cdot T} \cdot 100 \tag{6}$$

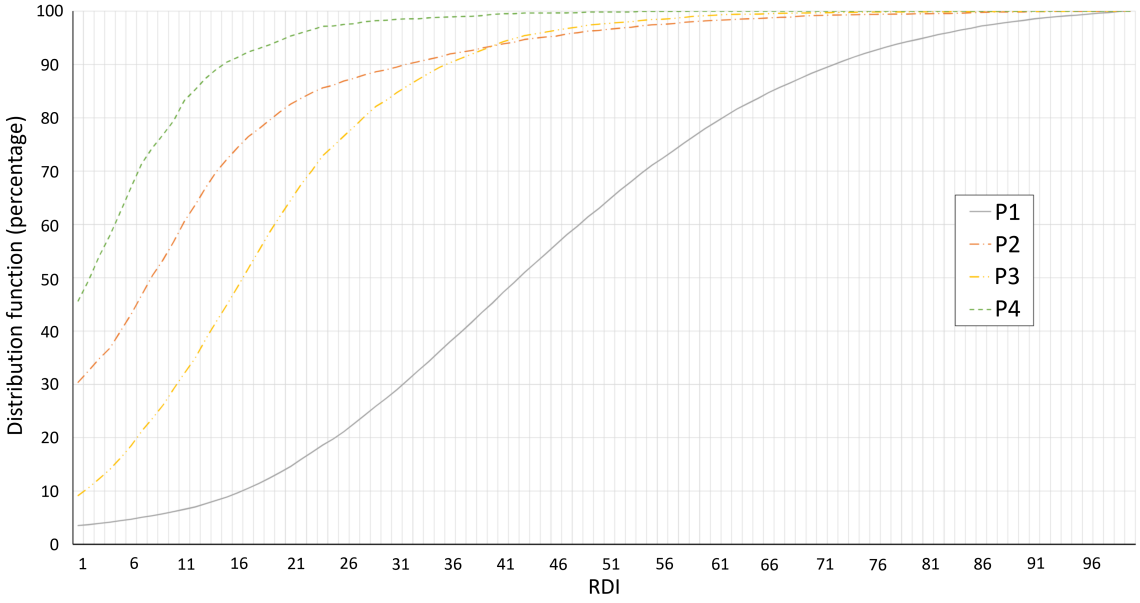

Figure 3: Cumulative distribution function

	P1	P2	P3	P4
$ARPD(best)$	0.00	0.00	0.03	0.36
$ARPD(all)$	74.15	17.71	31.04	8.00
CPU time (s)	2105.69	0.50	227.18	0.00
# Solutions (T)	1674450484.27	289836.00	139556401.64	116.67

Table 3: Structure of the solutions. Average values

6.3 Quality of the solutions

In this subsection, we compare the quality of the optimal solutions for different solution spaces. More specifically, we compare the MILP models in Section 5, and the main combinations of solution representations in $\mathcal{R}_4$ with $\mathcal{D}^{\mathcal{S}}$ and $\mathcal{D}^{\mathcal{A}}$ (see Table 1) by complete enumeration of all the solutions. The MILP models are solved using Gurobi 7.02 solver with 1500 seconds as stopping criterion, while the algorithms for complete enumeration are coded in C# (with Visual Studio 2010).

The following approaches have been tested using the MILP models:

- OS1: Optimal solution of P1 (i.e. $FHm, ((PM^k)_{k=1}^m)||C_{max}$). This solution is obtained by solving Model 1.
- OS2: Optimal solution of P2 by solving Model 2.
- OS3: Optimal solution of P3 by solving Model 3.
- OS4: Optimal solution of P4 by solving Model 4.

In addition, the optima of the solution spaces of some combinations of $\mathcal{R}_4$ –found by performing the complete enumeration of all solutions– with different machine assignment and job sequencing rules are tested:

- OS5: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, FIFO)$. Note that this (as well as the following variants of FIFO) differs from OS4 in that all possibilities for the ties are considered in the latter (i.e. $\mathcal{R}_4^F(FAM, FIFO^*)$).

- OS6: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, FIFO(iLS))$.
- OS7: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(SPT))$.
- OS8: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(LPT))$.
- OS9: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(PT + WINQ + AT))$.
- OS10: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(MWRF))$.
- OS11: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(LWRF))$.
- OS12: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(MTWF))$.
- OS13: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(LTWF))$.
- OS14: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, Q(LIFO))$.
- OS15: Best solution in the solution space defined by $\mathcal{R}_4^F(FAM, PERM)$.
- OS16: Best solution in the solution space defined by $\mathcal{R}_4^F(iPS, PERM)$.
- OS17: Best solution in the solution space defined by $\mathcal{R}_4^B(FAM, FIFO)$.

These combinations encompass all combinations for $\mathcal{D}^A$ and $\mathcal{D}^S$ found in the literature for $\mathcal{R}_4$ (see Table 1), with the exception of using the FIFO* and FIFO(rand) rules for job sequencing, which does not provide a unequivocal schedule. The complete enumeration of different combinations for other solution representations is not feasible even for small problem sizes.

Finally, we include two spaces of solutions that have not been employed in the literature. The idea behind is to assess the potential improvement that can be obtained by solving P1 (i.e. using $\mathcal{R}_1$) with respect to the best solutions that can be obtained with the most common solution representation adopted in the literature, i.e. $\mathcal{R}_4^F(FAM, FIFO)$. These approaches are:

- OS18: Best solution provided by Model 1 by forcing the sequence in the first stage to be OS5 (note that the sequences in the rest of the stages are not fixed).
- OS19: Best solution provided by Model 1 if the sequence in each stage is OS5. More specifically, Model 1 is solved fixing the sequence in all stages as the solution given by OS5

(note that OS5 obtains a unique sequence at the beginning of the shop, and sequences in the other stages are obtained by using the FIFO rule).

The computational results in terms of ARPD are shown in Table 4 (regarding MILP models, only the values when the models find an optimal solution are used in the table). Regarding average CPU times (in seconds), they are shown in Table 6. Finally, the number of optimal, feasible and no solutions found are summarised in Table 5.

Parameter	OS1	OS2	OS3	OS4	OS5	OS6	OS7	OS8	OS9	OS10	OS11	OS12	OS13	OS14	OS15	OS16	OS17	OS18	OS19
$n = 6$	0.00	0.00	0.02	0.41	0.62	0.45	0.77	0.71	0.54	0.42	1.31	0.42	1.30	0.37	1.87	0.85	0.45	0.15	0.26
$n = 7$	0.00	0.00	0.00	0.38	0.35	0.34	0.71	0.84	0.43	0.26	1.33	0.37	0.96	0.27	1.28	0.68	0.40	0.10	0.27
$n = 8$	0.00	0.00	0.00	0.24	0.41	0.40	0.89	0.81	0.44	0.49	1.63	0.47	1.36	0.42	1.61	0.60	0.38	0.18	0.34
$n = 9$	0.00	0.00	0.01	0.07	0.39	0.36	0.59	0.66	0.47	0.30	1.10	0.32	1.09	0.31	1.75	0.68	0.52	0.19	0.27
$n = 10$	0.00	0.00	0.02	0.00	0.15	0.15	0.47	0.40	0.21	0.20	1.10	0.18	0.90	0.15	1.39	0.68	0.21	0.08	0.13
$n = 11$	0.00	0.00	0.01	—	0.32	0.31	0.88	0.70	0.39	0.43	1.22	0.43	1.28	0.19	1.73	0.87	0.33	0.12	0.17
$m = 2$	0.00	0.00	0.00	0.12	0.14	0.09	0.36	0.09	0.22	0.09	0.36	0.10	0.27	0.09	0.76	0.38	0.14	0.04	0.04
$m = 3$	0.00	0.00	0.00	0.20	0.29	0.22	0.48	0.50	0.39	0.27	0.97	0.27	0.84	0.29	1.43	0.61	0.34	0.12	0.17
$m = 4$	0.00	0.00	0.01	0.73	0.52	0.50	1.24	1.20	0.56	0.49	2.18	0.56	1.95	0.35	1.73	0.84	0.56	0.18	0.36
$m = 5$	0.00	0.00	0.03	0.68	0.54	0.53	0.80	0.96	0.48	0.55	1.62	0.53	1.53	0.41	2.50	1.08	0.49	0.22	0.38
$\tau = 0$	0.00	0.00	0.01	0.30	0.37	0.37	0.49	0.63	0.38	0.39	0.84	0.35	0.80	0.27	1.84	0.86	0.51	0.08	0.22
$\tau = 1$	0.00	0.00	0.02	0.29	0.48	0.40	0.65	0.38	0.57	0.34	0.74	0.36	0.70	0.40	2.45	0.93	0.44	0.19	0.28
$\tau = 2$	0.00	0.00	0.00	0.37	0.27	0.23	1.01	1.05	0.29	0.31	2.26	0.39	1.94	0.18	0.52	0.39	0.19	0.14	0.21
Average	0.00	0.00	0.01	0.33	0.37	0.33	0.72	0.69	0.41	0.35	1.28	0.37	1.15	0.29	1.60	0.73	0.38	0.14	0.24

Table 4: Computational results of schemes of solutions: ARPD

Parameter	OS1			OS2			OS3			OS4			OS18			OS19		
	#O	#F	#N	#O	#F	#N	#O	#F	#N	#O	#F	#N	#O	#F	#N	#O	#F	#N
$n = 6$	60	0	0	60	0	0	60	0	0	53	7	0	60	0	0	60	0	0
$n = 7$	60	0	0	43	0	17	60	0	0	45	1	14	60	0	0	60	0	0
$n = 8$	60	0	0	31	0	29	60	0	0	22	1	37	60	0	0	60	0	0
$n = 9$	58	2	0	14	7	39	60	0	0	15	3	42	60	0	0	60	0	0
$n = 10$	51	9	0	7	8	45	57	3	0	4	4	52	59	1	0	60	0	0
$n = 11$	45	15	0	1	8	51	51	9	0	0	4	56	57	3	0	60	0	0
$m = 2$	90	0	0	61	13	16	90	0	0	60	6	24	90	0	0	90	0	0
$m = 3$	83	7	0	45	6	39	88	2	0	35	3	52	90	0	0	90	0	0
$m = 4$	84	6	0	30	2	58	87	3	0	30	2	58	90	0	0	90	0	0
$m = 5$	77	13	0	20	2	68	83	7	0	14	2	74	86	4	0	90	0	0
$\tau = 0$	111	9	0	47	7	66	116	4	0	42	1	77	117	3	0	120	0	0
$\tau = 1$	108	12	0	41	5	74	116	4	0	40	0	80	119	1	0	120	0	0
$\tau = 2$	115	5	0	68	11	41	116	4	0	57	12	51	120	0	0	120	0	0
All	334	26	0	156	23	181	348	12	0	139	13	208	356	4	0	360	0	0

Table 5: Computational results of schemes of solutions using MILP models: optimal solutions (#O), feasible solutions (#F), no solution is found (#N)

Parameter	OS1	OS2	OS3	OS4	OS18	OS19
$n = 6$	0.38	269.54	0.53	59.22	0.28	0.19
$n = 7$	0.99	550.72	1.02	522.65	0.54	0.25
$n = 8$	8.76	833.52	6.65	1059.64	2.53	0.51
$n = 9$	110.17	1230.49	24.71	1238.26	27.13	2.34
$n = 10$	286.66	1452.09	153.22	1418.82	56.45	21.65
$n = 11$	501.73	1500.38	421.54	1495.79	383.71	257.69
$m = 2$	27.20	610.62	19.90	599.23	42.33	38.12
$m = 3$	143.88	987.00	103.20	841.33	50.31	43.30
$m = 4$	176.43	1109.45	108.85	1067.38	75.20	50.19
$m = 5$	258.27	1334.84	173.16	1204.23	145.92	56.81
$\tau = 0$	157.48	1079.94	90.77	998.15	98.59	46.88
$\tau = 1$	208.91	1064.24	105.43	1030.63	84.90	49.23
$\tau = 2$	87.95	887.24	107.63	755.36	51.84	45.21
All	151.45	1010.48	101.28	928.04	78.44	47.11

Table 6: Computational results of schemes of solutions: CPU times (s)

In view of the results, the following comments can be done:

1. OS1 to OS3 are solutions of the same quality, as it can be seen from Table 4. A non-

parametric Mann-Whitney test is conducted between OS1 and OS3 and confirms that the hypotheses that the ARPD of OS1 and OS3 are equal cannot be rejected with $\alpha = 0.05$ (p-value=0.157, comparison performed in the instances where both approaches found their corresponding optimal solution). This means there are not (statistical) differences between exploring the full space of semiactive schedules of the HFS problem and restricting the search to $\mathcal{R}_2(FAM)$ $\mathcal{R}_3(FIFO^*)$. However, the computation times in Tables 5 and 6 show that –at least if the MILP models presented in Section 5 are employed– obtaining OS2 requires much more CPU time than obtaining OS1 and OS3, which require roughly the same computational effort.

2. The quality of solutions of OS1 and OS4 is not the same, but the difference in terms of ARPD is somewhat meagre: 0.32 on average. Although according to a non-parametric Mann-Whitney, both OS1 and OS4 are statistically significant with a p-value of 0.000 (this statistical difference between $\mathcal{R}_1$ and $\mathcal{R}_4$ is also found using OS5 and comparing against OS1), the small difference in the ARPD speaks for the high efficiency of $\mathcal{R}_4$, particularly if the interest lies on obtaining good solutions in short computation times.

3. Although the differences in ARPD for OS4 and OS5 are not statistically significant (p-value equals 0.842 using a Mann-Whitney test) on the overall testbed (the difference in ARPD, when OS4 finds an optimal solution, is 0.074), they can be notable for some parameters (see e.g. for $n = 8$ or $m = 4$ in Table 4). Recall that the difference between both approaches is that OS4 is obtained using $\mathcal{R}_4^F(FAM, FIFO^*)$ considering all ties (FIFO*), while in OS5, the ties are solved at random (i.e. $\mathcal{R}_4^F(FAM, FIFO)$). This gives some hints regarding the possibility of improving the solutions by elaborating smart tie-breaking mechanisms.

4. OS5 and OS17 are solutions of similar quality. This is confirmed by conducting a non-parametric Mann-Whitney test with a confidence level of $\alpha = 0.05$, resulting that the hypothesis cannot be rejected as the p-value equals 0.441. Therefore, we can conclude that $\mathcal{R}_4^F$ and $\mathcal{R}_4^B$ (using a sequence of jobs to represent the first or the last stage) produce similar results if FIFO and FAM (the most common approaches for machine assignment and job sequencing) are used.

5. OS6 to OS14 are not better than OS5. A series of tests are conducted, resulting in that these hypotheses cannot be rejected with a confidence level of $\alpha = 0.05$ (Mann-Whitney). These results imply that, when using $\mathcal{R}_4^F$ and FAM for machine assignment, the variants of the FIFO rule for job sequencing –i.e. FIFO(iLS), Q(SPT), Q(LPT), Q(PT+WINQ+AT), Q(MWRF), Q(LWRF), Q(MTWF), Q(LTWF), and Q(LIFO)– do not improve the original rule.

6. In Table 4 it can be seen that the ARPD of OS5 is 0.37 and the ARPD of OS18 is 0.13. Therefore, the optimal solution of $\mathcal{R}_4^F(FAM, FIFO)$ could be used to reduce the search space in $\mathcal{R}_1$ (by fixing the sequence in the first stage OS5), and this reduced solution space would still contain very high quality solutions (indeed a hypotheses test shows that there are statistical differences (Mann-Whitney test) between OS5 and OS18 with a confidence level of $\alpha = 0.05$, p-value equals 0.000). Fixing the sequences in all stages (i.e. using the sequence in the first stage given by OS5 and the sequences in other stages using the FIFO rule) to restrict the search in $\mathcal{R}_1$ (as it is done to obtain OS19) does not perform bad either, as the ARPD of OS18 is 0.24. In this latter case, however, there are not statistically significant differences between OS5 and OS19 (p-value equals 0.059 using a non-parametric Mann-Whitney test).

7. In view of the poor quality of OS15 (i.e. using the same sequence of jobs in all stages), it can be concluded that employing the same sequence of jobs in every stage (permutation restriction) does not provide good results for the HFS problem. The differences between OS15 and OS5 are statistically significant (p-value equals 0.000 using a non-parametric Mann-Whitney test).

7 Conclusions

In this paper, the efficiency of solution representations for the hybrid flow shop scheduling problem to minimise the makespan has been studied. First, we have reviewed and classified the different solution representations employed in the literature, together with the main job sequencing and machine assignment rules required for some of the representations. In addition, we include a

solution representation that, to the best of out knowledge, has not been employed so far. Then, the spaces of solutions defined by the most common solution representations and rules have been studied in terms of their size and in terms of the quality of the solutions that can be obtained. Regarding the size of the solution space, we explicitly give the size of the solution space of the different representations. Regarding the quality of the solutions that can be obtained, we carry out an exhaustive computational evaluation of the most employed solution representations and combinations of rules. A number of conclusions and future research lines can be obtained from the evaluation, which can be summarised as follows:

- High-quality solutions can be obtained using solution representations ($\mathcal{R}_2$ and $\mathcal{R}_3$) that only explore a small portion of the full space of semiactive schedules (given by representation $\mathcal{R}_1$). On the one hand the use of $\mathcal{R}_2$ and $\mathcal{R}_3$ allows to find extremely high-quality solutions not further than 0.02% from the optimal solution. On the other hand, $\mathcal{R}_4$ (i.e. a sequence of n jobs) greatly reduces the search space while their best solutions are only marginally outperformed by the best solutions in $\mathcal{R}_1$. Consequently, this solution representation seems to be quite apt for approximate algorithms.

- In contrast, $\mathcal{R}_1$ seems to be suitable if optimal solutions are sought. If this is the case, the fact that the optimal solutions in $\mathcal{R}_4$ are very good solutions for $\mathcal{R}_1$ can be used in order to either provide a tight upper bound, or to restrict the search space in $\mathcal{R}_1$ without greatly diminishing the quality of the solutions by fixing the job sequence in the first/all stages.

- Most contributions employing $\mathcal{R}_4$ use some variant of FIFO for job sequencing and FAM for machine assignment. However, our experimentation shows that the results obtained can be improved if the sequence obtained by $\mathcal{R}_4$ is fixed as the sequence in the first stage and either other assignments in machines or sequencing rules in other stages are tested (i.e. OS18). This seems to indicate that there is room for improving the results using $\mathcal{R}_4$ by devising alternative local search on the space of solutions of $\mathcal{R}_1$, $\mathcal{R}_2$, and/or $\mathcal{R}_3$ (in this regard, see e.g. Urlings et al., 2010; Fernandez-Viagas et al., 2018a).

- If $\mathcal{R}_4$ plus FIFO for job sequencing and FAM for machine assignment is used, i.e.

$\mathcal{R}_4(FAM, FIFO)$, then the quality of solutions is similar if the encoding represents the job sequence in the first stage ($\mathcal{R}_4^F$) or in the last stage ($\mathcal{R}_4^B$).

- Although most of the different variants of job sequencing rules do not seem to statistically improve the results of the original one –at least when FAM is employed for machine assignment–, the manner in which the ties are solved by FIFO may play a role. However, the use of different job sequencing rules than FIFO is very limited in approximate algorithms in the literature. Again, this seems to speak for some possibility of improving the performance of the current algorithms: either by using some smart tie-breaking rule for FIFO or by changing/combining different job sequencing rules along the algorithms (in this regard, see e.g. Wang et al., 2013).

- Although the best solutions are found using $\mathcal{R}_1$, the number of solutions in this space is huge as compared to other representations of the solutions. Future exact/approximate algorithms should include specific properties of the problem to avoid or bound the evaluation of such a high number of solutions.

- The combination of the MILP models developed in this paper together with the different solution representations could result in efficient matheuristics for the problem.

Acknowledgements

The authors are sincerely grateful to the anonymous referees, who provide very valuable comments on the earlier version of the paper. This research has been funded by the Spanish Ministry of Science and Innovation, under the project "PROMISE" with reference DPI2016-80750-P.

References

Acero-Dominguez, M. and Paternina-Arboleda, C. (2004). Scheduling jobs on a k-stage flexible flow shop using a TOC-based (bottleneck) procedure. *2004 IEEE Systems and Information Engineering Design Symposium*, pages 295–298.

Allahverdi, A. and Al-Anzi, F. (2006). Scheduling multi-stage parallel-processor services to minimize average response time. *Journal of the Operational Research Society*, 57(1):101–110.

Barman, S. (1997). Simple priority rule combinations: An approach to improve both flow time and tardiness. *International Journal of Production Research*, 35(10):2857–2870.

Belkadi, K., Gourgand, M., and Benyettou, M. (2006). Parallel genetic algorithms with migration for the hybrid flow shop scheduling problem. *Journal of Applied Mathematics and Decision Sciences*, 2006.

Brah, S. (1996). A comparative analysis of due date based job sequencing rules in a flow shop with multiple processors. *Production Planning and Control*, 7(4):362–373.

Brah, S. and Loo, L. (1999). Heuristics for scheduling in a flow shop with multiple processors. *European Journal of Operational Research*, 113(1):113–122.

Brah, S. and Wheeler, G. (1998). Comparison of scheduling rules in a flow shop with multiple processors: A simulation. *Simulation*, 71(5):302–311.

Buddala, R. and Mahapatra, S. (2018). Improved teaching-learning-based and JAYA optimization algorithms for solving flexible flow shop scheduling problems. *Journal of Industrial Engineering International*, 14(3):555–570.

Carlier, J. and Néron, E. (2000). An exact method for solving the multi-processor flow-shop. *RAIRO - Operations Research*, 34(1):1–25.

Chung, T.-P., Sun, H., and Liao, C.-J. (2017). Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint. *Computers and Industrial Engineering*, 113:859–870.

Cui, Z. and Gu, X. (2014). A discrete group search optimizer for hybrid flowshop scheduling problem with random breakdown. *Mathematical Problems in Engineering*, 2014.

Dios, M., Fernandez-Viagas, V., and Framinan, J. (2018). Efficient heuristics for the hybrid flow shop scheduling problem with missing operations. *Computers and Industrial Engineering*, 115:88–99.

Fernandez-Viagas, V. and Framinan, J. (2017). Reduction of permutation flowshop problems to single machine problems using machine dominance relations. *Computers and Operations Research*, 77:96–110.

Fernandez-Viagas, V., Molina-Pariente, J. M., and Framinan, J. M. (2018a). New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Systems with Applications*, 114:345 – 356.

Fernandez-Viagas, V., Ruiz, R., and Framinan, J. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*, 257(3):707–721.

Fernandez-Viagas, V., Valente, J., and Framinan, J. (2018b). Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. *Expert Systems with Applications*, 94:58–69.

Framinan, J., Gupta, J., and Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255.

Graham, R. L., Lawler, E. L., 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.

Gupta, J. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society*, 39(4):359–3641.

Han, Z., Sun, Y., Ma, X., and Lv, Z. (2018). Hybrid flow shop scheduling with finite buffers. *International Journal of Simulation and Process Modelling*, 13(2):156–166.

Holthaus, O. and Rajendran, C. (1997). Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48(1):87–105.

Hunsucker, J. and Shah, J. (1992). Performance of priority rules in a due date flow shop. *Omega*, 20(1):73 – 89.

Jayamohan, M. and Rajendran, C. (2000). A comparative analysis of two different approaches to scheduling in flexible flow shops. *Production Planning and Control*, 11(6):572–580.

Jin, Z., Yang, Z., and Ito, T. (2006). Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics*, 100(2):322–334.

Jouglet, A., Oguz, C., and Sevaux, M. (2009). Hybrid flow-shop: A memetic algorithm using constraint-based scheduling for efficient search. *Journal of Mathematical Modelling and Algorithms*, 8(3):271–292.

Koulamas, C. and Kyparisis, G. (2000). Asymptotically optimal linear time algorithms for two-stage and three-stage flexible flow shops. *Naval Research Logistics*, 47(3):259–268.

Kouvelis, P. and Vairaktarakis, G. (1998). Flowshops with processing flexibility across production stages. *IIE Transactions (Institute of Industrial Engineers)*, 30(8):735–746.

Lahimer, A., Lopez, P., and Haouari, M. (2013). Improved bounds for hybrid flow shop scheduling with multiprocessor tasks. *Computers and Industrial Engineering*, 66(4):1106–1114.

Lee, C.-Y. and Vairaktarakis, G. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters*, 16(3):149–158.

Liao, C.-J., Tjandradjaja, E., and Chung, T.-P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Applied Soft Computing Journal*, 12(6):1755–1764.

Naderi, B., Gohari, S., and Yazdani, M. (2014). Hybrid flexible flowshop problems: Models and solution methods. *Applied Mathematical Modelling*, 38(24):5767–5780.

Naderi, B., Ruiz, R., and Zandieh, M. (2010). Algorithms for a realistic variant of flowshop scheduling. *Computers and Operations Research*, 37(2):236–246.

Naderi, B., Zandieh, M., Khaleghi Ghoshe Balagh, A., and Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications*, 36(6):9625–9633.

Niu, Q., Zhou, T., and Ma, S. (2009). A quantum-inspired immune algorithm for hybrid flow shop with makespan criterion. *Journal of Universal Computer Science*, 15(4):765–785.

Nowicki, E. and Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, 106(2-3):226–253.

Oguz, C. and Ercan, M. (2005). A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling*, 8(4):323–351.

Oguz, C., Zinder, Y., Do, V. H., Janiak, A., and Lichtenstein, M. (2004). Hybrid flow-shop scheduling problems with multiprocessor task systems. *European Journal of Operational Research*, 152(1):115 – 131.

Pan, Q.-K., Gao, L., Li, X.-Y., and Gao, K.-Z. (2017). Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. *Applied Mathematics and Computation*, 303:89–112.

Pan, Q.-K., Wang, L., Li, J.-Q., and Duan, J.-H. (2014). A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. *Omega (United Kingdom)*, 45:42–56.

Paternina-Arboleda, C., Montoya-Torres, J., Acero-Dominguez, M., and Herrera-Hernandez, M. (2008). Scheduling jobs on a k-stage flexible flow-shop. *Annals of Operations Research*, 164(1):29–40.

Perez-Gonzalez, P. and Framinan, J. (2009). Scheduling permutation flowshops with initial availability constraint: Analysis of solutions and constructive heuristics. *Computers and Operations Research*, 36(10):2866–2876.

Pinedo, M. (1995). *Scheduling: Theory, Algorithms and Systems*. Prentice Hall.

Ribas, I., Leisten, R., and Framinan, J. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers and Operations Research*, 37(8):1439–1454.

Rinnooy Kan, A. H. G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations*. Martinus Nijhoff, The Hague.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494.

Ruiz, R. and Vázquez-Rodríguez, J. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1–18.

Santos, D., Hunsucker, J., and Deal, D. (1996). An evaluation of sequencing heuristics in flow shops with multiple processors. *Computers and Industrial Engineering*, 30(4):681–692.

Serifoglu, F. and Ulusoy, G. (2004). Multiprocessor task scheduling in multistage hybrid flow-shops: A genetic algorithm approach. *Journal of the Operational Research Society*, 55(5):504–512.

Soewandi, H. and Elmaghraby, S. (2001). Sequencing three-stage flexible flowshops with identical machines to minimize makespan. *IIE Transactions (Institute of Industrial Engineers)*, 33(11):985–993.

Su, S., Yu, H., Wu, Z., and Tian, W. (2014). A distributed coevolutionary algorithm for multiobjective hybrid flowshop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 70(1-4):477–494.

Urlings, T., Ruiz, R., and Stützle, T. (2010). Shifting representation search for hybrid flexible flowline problems. *European Journal of Operational Research*, 207(2):1086–1095.

Vairaktarakis, G. and Elhafsi, M. (2000). The use of flowlines to simplify routing complexity in two-stage flowshops. *IIE Transactions (Institute of Industrial Engineers)*, 32(8):687–699.

Wang, S.-Y., Wang, L., Liu, M., and Xu, Y. (2013). An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. *International Journal of Advanced Manufacturing Technology*, 68(9-12):2043–2056.

Xu, Y., Wang, L., Wang, S., and Liu, M. (2013). An effective shuffled frog-leaping algorithm for solving the hybrid flow-shop scheduling problem with identical parallel machines. *Engineering Optimization*, 45(12):1409–1430.

Ying, K.-C. and Lin, S.-W. (2018). Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. *Expert Systems with Applications*, 92:132–141.

Yu, C., Semeraro, Q., and Matta, A. (2018). A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. *Computers and Operations Research*, 100:211–229.

Zhong, W. and Shi, Y. (2018). Two-stage no-wait hybrid flowshop scheduling with inter-stage flexibility. *Journal of Combinatorial Optimization*, 35(1):108–125.