Reduction of Permutation Flowshop Problems to Single Machine Problems using Machine Dominance Relations[*]

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

[1] Industrial Management, School of Engineering, University of Seville,
Ave. Descubrimientos s/n, E41092 Seville, Spain, {vfernandezviagas,framinan}@us.es

July 18, 2016

Abstract

The Permutation Flowshop Scheduling Problem with Makespan objective (PFSP-M) is known to be NP-hard for more than two machines, and literally hundreds of works in the last decades have proposed exact and approximate algorithms to solve it. These works –of computational/experimental nature– show that the PFSP-M is also *empirically* hard, in the sense that optimal or quasi-optimal sequences statistically represent a very small fraction of the space of feasible solutions, and that there are big differences among the corresponding makespan values. In the vast majority of these works, it has been assumed that a) processing times are not job- and/or machine-correlated, and b) all machines are initially available. However, some works have found that the problem turns to be almost trivial (i.e. almost every sequence yields an optimal or quasi-optimal solution) if one of these assumptions is dropped. To the best of our knowledge, no theoretical or experimental explanation has been proposed by this rather peculiar fact.

Our hypothesis is that, under certain conditions of machine availability, or correlated processing times, the performance of a given sequence in a flowshop is largely determined by only one stage, thus effectively transforming the flowshop layout into a single machine. Since the single machine scheduling problem with makespan objective is a trivial problem where all feasible sequences are optimal, it would follow that, under these conditions, the equivalent PFSP-M is almost trivial. To address this working hypothesis from a general perspective, we investigate some conditions that allow reducing a permutation flowshop scheduling problem to a single machine scheduling problem, focusing on the two most common objectives in the literature, namely makespan and flowtime. Our work is a combination of theoretical and computational analysis, therefore several properties are derived to prove the conditions for an exact (theoretical) equivalence, together with an extensive computational evaluation to establish an empirical equivalence.

Keywords: Scheduling, Flowshop, processing times, PFSP, makespan, flowtime, Single Machine, dominating machine

[*]Preprint submitted to Computers and Operations Research. http://dx.doi.org/10.1016/j.cor.2016.07.009

[†]Corresponding author. Tel.: +34-954487220.

1 Introduction

The Permutation Flowshop Scheduling Problem (denoted as PFSP) is one of the most studied problems in Operations Research (see e.g. Framinan et al., 2004, Reza Hejazi and Saghafian, 2005, Ruiz and Maroto, 2005, Framinan et al., 2005, Pan and Ruiz, 2013 and Vallada et al., 2008). This decision problem can be defined as follows: n jobs have to be processed on each one of the m machines of the shop where every job follows the same route of machines. The problem consists on determining the processing sequence of the jobs in the shop, assuming that the same sequence is adopted for each machine. Among other assumptions (see e.g. Dudek and Teuton, 1964 for a complete description), machines are always available since time zero as well as sequence-dependent set-up times are insignificant while sequence-independent set-up times are non-anticipatory and therefore, added to the processing times of the jobs.

Most research has focused in the minimisation of makespan and total flowtime (see e.g. the reviews by Framinan et al., 2004, Ruiz and Maroto, 2005 and Pan and Ruiz, 2013), although other objectives have been also considered (see e.g. Leisten and Rajendran, 2014 and Gajpal and Rajendran, 2006 for the homogeneity of the completion times; Fernandez-Viagas and Framinan, 2015b and Vallada and Ruiz, 2010 for total tardiness; M'Hallah, 2014 and Schaller and Valente, 2013 for total tardiness and earliness; or Sun et al., 2011 and Framinan and Leisten, 2006 for several objectives). Following the notation of Graham et al. (1979), the PFSP to minimise makespan and total flowtime are $Fm|prmu|C_{max}$ and $Fm|prmu|\sum C_j$ respectively.

Since $Fm|prmu|C_{max}$ was shown to be NP-hard for $m > 2$, and $Fm|prmu|\sum C_j$ strongly NP-hard for $m \geq 2$ by Rinnooy Kan (1976) and Garey et al. (1976), hundreds of heuristics and metaheuristics have been proposed in the literature to obtain good solutions in reasonable computation times (see e.g. Fernandez-Viagas and Framinan, 2015a, Dong et al., 2013, Rad et al., 2009 and Fernandez-Viagas and Framinan, 2014). The commonly accepted procedure in these works is to prove the effectiveness of the algorithms in statistical terms by solving a collection of published instances or testbeds (see the testbeds by Taillard, 1993, Carlier, 1978, Reeves, 1995, Demirkol et al., 1998, Heller, 1960 and Vallada et al., 2015). In all these testbeds, the processing times of each job on each machine have been generated using a uniform distribution with exactly the same distribution parameters regardless the job or the machine. In other words, in these testbeds there is no job- and/or machine-correlation of the processing times of each instance. When different heuristics and metaheuristics are employed to solve the instances in these testbeds, it turns out that, for most instances, *good sequences* –meaning sequences for which the corresponding objective function value is not far from the optimum– in the solution space statistically represent just a very small fraction of the total, and that there are enormous differences between good and bad sequences.

In a noteworthy contribution, Watson et al. (2002) dropped the lack of correlation assumption for the

$Fm|prmu|C_{max}$ problem, and generated different correlations (i.e. job, machine and mixed correlations) depending if the same parameters of the distribution are used across jobs, across machines or both. Results show that most structured (correlated) $Fm|prmu|C_{max}$ problems tend to be easily solvable in the sense that there are a lot of good sequences in the solution space and therefore, the probability to find a good sequence is very high.

Another common assumption in the testbeds is that the shop floor is empty at the time of the scheduling decision and therefore, each machine is available at time zero. This assumption is questionable for real-life cases, as it could be applied only if it is the first time that jobs are to be scheduled in the shop, or when a long period without processing jobs has occurred. In the work by Perez-Gonzalez and Framinan (2009), the assumption is removed, and restrictions in the initial availability of the machines are considered in a permutation flowshop with makespan objective. The computational results show how the initial availability assumption makes the problem easier (again in the sense of increasing the probability of finding a good sequence), specially if structured processing times are considered.

As a conclusion, the studies by Watson et al. (2002) and Perez-Gonzalez and Framinan (2009) show that, while the $Fm|prmu|C_{max}$ problem is *empirically* hard in the testbeds usually employed in the literature, it turns to be extremely easy under some conditions regarding the processing times and/or machine availability. However, to the best of our knowledge, no paper further explores the ultimate reasons why such assumptions make the instances to be ‘easy’. Our hypothesis is that, under certain conditions, the permutation flowshop and the single machine scheduling problems (denoted as $1||C_{max}$ to minimise makespan and as $1||\sum C_j$ to minimise total flowtime) can be (approximately) equivalent. Note that both structured processing times and the initial machine availability may imbalance the machine workloads as compared to that in the classical benchmarks. This would in turn cause that the performance of a sequence in the whole flowshop is largely determined by its performance in the most loaded machine. If this happens, the almost-trivial behaviour found by Watson et al. (2002) and Perez-Gonzalez and Framinan (2009) for the makespan objective can be explained.

We are not aware of previous literature devoted to analyse the equivalence between both scheduling problems, although some works have been carried out for the PFSP with dominant machines (see e.g. Cepek et al., 2002, Easwaran et al., 2010 and Ho and Gupta, 1995). In this paper we analyse the effect of the most loaded machine in the flowshop in order to determine the conditions that make a PFSP instance to be reduced to a single machine scheduling instance. Several theoretical properties and an extensive computational study is carried out for the two most common objectives in the literature, i.e.: makespan and total flowtime. These properties extend and integrate existing literature on machine dominance for the PFSP, so some previous results are explained as direct consequences of the equivalence between both scheduling problems and therefore, they can be seen as corollaries of the properties and theorems presented in this paper. Note that the seminal paper by Johnson (1954) already establishes conditions for the optimality of the 2- and 3-machine flowshop, and, since our hypothesis would lead to transforming the PFSP with makespan objective into a trivial problem (and thus optimally solvable), our research could be related to Johnson's paper. However, from our perspective, finding the optimal solution for the makespan case is simply a by-product of the reduction of the PFSP to the single machine case and therefore the relation between both papers is limited.

The remainder of the paper is organized as follows: The problems under study are formally stated in Section 2. In Section 3, several dominance rules are established to be able to compare both problems. Section 4 empirically analyses the relations between the problems. Finally, in Section 5, several conclusions are discussed.

2 Problem Statement

The notation of the PFSP can be set as follows: n jobs have to be scheduled in a flowshop with m machines. The processing time of each job j on machine i is defined as p_{ij}. Given a sequence of jobs $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$, $C_{i\pi_j}$ the completion time of job π_j on machine i can be computed according to:

$$C_{i\pi_j} = \max\{C_{i-1,\pi_j}, C_{i\pi_{j-1}}\} + p_{i\pi_j} \quad (1)$$

The makespan or maximum completion time –denoted as C_{max}– is defined as the completion time of the last job of the sequence on the last machine, i.e. $C_{max} = C_{m\pi_n}$. Similarly, $\sum C_j$ is defined as the total flowtime of the sequence: $\sum C_j = \sum_{j=1}^{n} C_{mj}$. Finally, let $IT_{i\pi_k}$ be the idle time immediately before job π_k on machine i. Clearly,

$$IT_{i\pi_k} = \begin{cases} C_{i\pi_k} - C_{i\pi_{k-1}} - p_{i\pi_k}, & k \in 2 \ldots n \\ C_{i\pi_k} - p_{i\pi_k}, & k = 1 \end{cases} \quad (2)$$

or analogously,

$$IT_{i\pi_k} = \max\{0, C_{i-1,\pi_k} - C_{i\pi_{k-1}}\}, \forall k, C_{i0} = C_{0j} = 0 \quad (3)$$

Regarding the single machine scheduling problem, denoted as SMSP, n jobs have to be scheduled in a shop with a unique machine. The processing times and the completion times of job j in that machine are denoted by p_j and C_j respectively.

Once the PFSP and the SMSP have been formulated, let us introduce some useful definitions. For a given instance of the PFSP, the machine s with the highest sum of processing times is denoted as *saturated* *machine*. More specifically:

$$s = \arg\max_i \sum_{j=1}^{n} p_{ij}$$

The remaining machines $i \neq s$ are denoted as *non saturated machines*. Note that we intentionally avoid the term *bottleneck* for this machine, since this concept is usually related to the long-term busy time of a machine, whereas in our case the saturated machine is related to a specific set of jobs to be scheduled in this shop floor. However, when applied to a short-term focus, both concepts are related.

The above definition allows us to define two types of dominance between machines.

- Dominance type I, (see e.g. Cepek et al., 2002): a machine a dominates (type I) a machine b if $p_{aj} \geq p_{bj'}$, $\forall j \neq j^{'}$, where the machine b is consequently denoted as type-I-dominated machine.
- Dominance type II, (see e.g. Holm, 1979, Cheng et al., 2007 and Wang et al., 2006): a machine a dominates (type II) a machine b (denoted as type-II-dominated) if $\min_{\forall j} p_{aj} \geq \max_{\forall j} p_{bj}$. Note that the only difference between both types is that the processing time of each job on machine a must be higher than on machine b for dominance type II. In fact, dominance type II implies dominance type I, while the opposite is not true in general (Cepek et al., 2002).

Additionally, let us define the following dominance cases for a flowshop of more than two machines:

- Case *ddm*: Each machine i dominates (type I) machine $i+1$, $\forall i \in [1, m-1]$.
- Case *idm*: Each machine i dominates (type I) machine $i-1$, $\forall i \in [2, m]$.
- Case *idm-ddm*: In this case, each machine i_1 ($\forall i_1 \in [2, s]$) dominates (type I) machine $i_1 - 1$ and each machine i_2 ($\forall i_2 \in [s, m-1]$) dominates (type I) machine $i_2 + 1$. Obviously, machine s is the saturated machine.

Finally, let us define the equivalence between the PFSP and the SMSP as follows: Given an instance $\mathcal{I}$ of a PFSP with processing times p_{ij}, and $\hat{\mathcal{I}}$ an artificial instance of a SMSP with $\hat{p}_j = p_{sj}$, we say that, for instance $\mathcal{I}$, both problems are equivalent regarding objective F if, for any feasible sequence Π, $F_{\mathcal{I}}(\Pi) = F_{\hat{\mathcal{I}}}(\Pi) + constant$. In other words, the PFSP and the SMSP are equivalent for an instance if the objective function values of all feasible sequences applied to both problems differ only with respect to a constant. Obviously, for an instance where both problems are equivalent, the optimal sequences are the same.

The notation introduced in this section is summarised in Table 1.

Table 1: Summary of indexes, parameters and variables

Indexes:
i: Index of machines.
j: Index of jobs.
s: Index of the saturated machine.
Parameters:
n, m: Number of jobs and machines respectively.
p_{ij}: Processing time of job j on machine i (permutation flowshop layout).
p_j: Processing time of job j (single machine layout).
$\hat{p}_j$: Processing time of job j on the saturated machine (i.e. on machine s), $\hat{p}_j = p_{sj}$.
$\mathcal{I}$: Instance of the PFSP.
$\hat{\mathcal{I}}$: Artificial instance of a SMSP where $p_j = p_{sj}$.
Variables:
IT_{ij}: Idle time of job j on machine i.
C_{ij}: Completion time of job j on machine i (permutation flowshop layout).
C_j: Completion time of job j on a single machine (single machine layout).

3 Theoretical analysis

Equipped with the above definitions, several properties can be derived to state a some PFSP instance is equivalent to SMSP for makespan and/or total flowtime minimisation under some (rather restrictive) assumptions. Since the PFSP with 2 machines has been widely analysed in the literature as an important particular case of the general m-machines cases, the properties presented in this paper also adopt this division, and are formalised in two separate sections.

3.1 PFSP with 2 machines

Let first assume that machine s is the saturated machine in a 2-machine PFSP. In order to be able to show that one instance of the PFSP with two machines is equivalent to the SMSP, we need to state several properties and corollaries as well as define a condition to be satisfied for the saturated machine (the hardest condition needed to prove the properties and corollaries is that $p_{sj} \geq p_{ij'}, \forall j \neq j', i \neq s$).

3.1.1 First machine is saturated, $s = 1$

First, we study the case where the first machine is saturated, i.e. $s = 1$. Let us define the following property in order to provide further insight in the understanding of this case:

Property 3.1. *Let $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ be a sequence of jobs with $p_{1\pi_k} \geq p_{2\pi_{k-1}}, \forall k \geq 2$. Then, the completion time of job π_k on the second machine equals its completion time on the first machine plus its processing time in the second, i.e. $C_{2\pi_k} = C_{1\pi_k} + p_{2\pi_k}$, $\forall k$.*

Proof. The property can be recursively proved in view of the definition of the completion time of job π_k

on the first machine:

$$C_{1\pi_k} = \begin{cases} C_{1\pi_{k-1}} + p_{1\pi_k}, & \forall k \geq 2 \\ p_{1\pi_k}, & k = 1 \end{cases}$$

and in the second one:

$$C_{2\pi_k} = \begin{cases} \max\{C_{1\pi_k}, C_{2\pi_{k-1}}\} + p_{2\pi_k}, & \forall k \geq 2 \\ p_{1\pi_k} + p_{2\pi_k}, & \forall k = 1 \end{cases}$$

Beginning with the second job of the sequence, π_2: on the one hand, taken into account $p_{1\pi_2} \geq p_{2\pi_1}$, the completion time on the first machine is $C_{1\pi_2} = C_{1\pi_1} + p_{1\pi_2} = p_{1\pi_1} + p_{1\pi_2} \geq p_{1\pi_1} + p_{2\pi_1} = C_{2\pi_1} \longrightarrow C_{1\pi_2} \geq C_{2\pi_1}$; on the other hand, the completion time on the second machine is $C_{2\pi_2} = \max\{C_{1\pi_2}, C_{2\pi_1}\} + p_{2\pi_2} = C_{1\pi_2} + p_{2\pi_2} \longrightarrow C_{2\pi_2} = C_{1\pi_2} + p_{2\pi_2}$ using the expression on the first machine.

Following with the third job of the sequence, π_3: the completion time on the first machine is $C_{1\pi_3} = C_{1\pi_2} + p_{1\pi_3} \geq C_{1\pi_2} + p_{2\pi_2} = C_{2\pi_2} \longrightarrow C_{1\pi_3} \geq C_{2\pi_2}$; on the second machine, the completion time is $C_{2\pi_3} = \max\{C_{1,\pi_3}, C_{2\pi_2}\} + p_{2\pi_3} = C_{1\pi_3} + p_{2\pi_3} \longrightarrow C_{2\pi_3} = C_{1\pi_3} + p_{2\pi_3}$.

Analogously, in a recursive manner, for job in position k, π_k: on the first machine, the completion time is $C_{1\pi_k} = C_{1\pi_{k-1}} + p_{1\pi_k} \geq C_{1\pi_{k-1}} + p_{2\pi_{k-1}} = C_{2\pi_{k-1}} \longrightarrow C_{1\pi_k} \geq C_{2\pi_{k-1}}$; then the completion time on the second machine is $C_{2\pi_k} = \max\{C_{1\pi_k}, C_{2\pi_{k-1}}\} + p_{2\pi_k} = C_{1\pi_k} + p_{2\pi_k} \longrightarrow C_{2\pi_k} = C_{1\pi_k} + p_{2\pi_k}$.

□

This property extends the following result found by Monma and Rinnooy Kan (1983), which can be seen now as a corollary of the above property:

Corollary 3.1. *(Monma and Rinnooy Kan, 1983: First part of Theorem 3 for $m = 2$). Let $\mathcal{I}$ be an instance of the PFSP where machine 2 is type-II-dominated by machine 1. Then, the completion time of a job on the second machine is equal to the completion time of a job on the first machine plus its processing time on the second machine.*

Proof. The proof of the corollary is obvious in view of Property 3.1. □

Corollary 3.2. *Let $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ be a sequence of jobs with $p_{1\pi_k} > p_{2\pi_{k-1}}$, $\forall k \geq 2$. Then, the idle time $IT_{2\pi_k}$ is always greater than 0, i.e. $IT_{2\pi_k} > 0$, $\forall k$.*

Proof. The proof of the corollary is obvious in view of Property 3.1 and taking into account the definition of idle time given in Equation (3). □

The above property and corollaries establish that the completion time of each job on the second machine depends only on its completion time on the first machine and that there are always idle time on the second machine, respectively. This occurs if the processing time of each job on the first machine is higher than its previous job on the second machine. Extending this condition to the processing time of each other job on the second machine, the equivalence between $F2|prmu|C_{max}$ and $1||C_{max}$ of the first machine is theoretically established in Theorem 3.1 with the exception of the last job of the sequence.

Theorem 3.1. *Let $\mathcal{I}$ be an instance of the $F2|prmu|C_{max}$ problem where $p_{1j} \geq p_{2j'}$, $\forall j \neq j'$ (i.e. machine 2 is dominated type I), and $\hat{\mathcal{I}}$ be an instance of the $1||C_{max}$ problem where $\hat{p}_j = p_{1j}$. Let Π be a sequence of the form $\Pi := \{\sigma, g\} := (\sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-1}, g)$ where g is the last job of the sequence and σ is an unknown sequence of $n-1$ jobs. Let C_{max} be the makespan obtained by Π on instance $\mathcal{I}$ and $\hat{C}_{max}$ its makespan on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $C_{max} = \hat{C}_{max} + p_{2g}$.*

Proof. Let us consider the PFSP with two machines to minimise makespan. In view of Property 3.1, $C_{max} = C_{2\pi_n} = C_{1\pi_n} + p_{2\pi_n}$. Then, minimising C_{max} in $F2|prmu|C_{max}$ is equivalent to minimise $C_{1\pi_n} + p_{2\pi_n}$. Considering that π_n is job g, $C_{max} = C_{1\pi_n} + p_{2\pi_n} = C_{1\pi_n} + p_{2,g} = \hat{C}_{max} + p_{2g}$. □

Corollary 3.3. *Under the conditions of Theorem 3.1, the optimal solutions for $\mathcal{I}$ and $\hat{\mathcal{I}}$ are the same. Additionally, any sequence of the form $\Pi := \{\sigma, e\}$ is an optimal sequence for both instances where e is the job with the least processing time on the second machine, i.e. $p_{2e} = \min_{\forall j} p_{2j}$.*

Proof. The proof of the theorem is obvious in view of Theorem 3.1 and taking into account that each feasible solution is an optimal sequence for the $1||C_{max}$ problem. □

Note that the result of this theorem is given in Ho and Gupta (1995) for $m = 2$ under more restrictive conditions, i.e. $\min_{\forall j} p_{1j} \geq \max_{\forall j'} p_{2j'}$, which can be seen now as a special case of the above result:

Corollary 3.4. *(Ho and Gupta, 1995: Theorem 2 for $m = 2$). Let $\mathcal{I}$ be an instance of $F2|prmu|C_{max}$ where the machine 2 is type-II-dominated by the machine 1. Then, any sequence of the form $\Pi := \{\sigma, e\}$ is optimal where σ is any sequence of $n-1$ jobs and e satisfies $p_{2e} = \min_{\forall j} p_{2j}$.*

Proof. The proof of the theorem is obvious in view of Corollary 3.3. □

On the other hand, the equivalence between $F2|prmu|\sum C_j$ and $1||\sum C_j$ is theoretically proved by Theorem 3.2 and Corollary 3.5.

Theorem 3.2. *Let $\mathcal{I}$ be an instance of $F2|prmu|\sum C_j$ where $p_{1j} \geq p_{2j'}$, $\forall j \neq j'$ (i.e. machine 2 is dominated type I), and $\hat{\mathcal{I}}$ be an instance of the $1||\sum C_j$ problem where $\hat{p}_j = p_{1j}$. Let Π be a sequence of the form $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$. Let $S(\Pi)$ be the total flowtime obtained by Π on instance $\mathcal{I}$ and $\hat{S}(\Pi)$ its total flowtime on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $S(\Pi) = \hat{S}(\Pi) + \sum_{j=1}^{n} p_{2j}$.*

Proof. Let us consider the PFSP with two machines to minimise the total flowtime, i.e. $\sum_{j=1}^{n} C_{2j}$. In view of Property 3.1, $\sum_{j=1}^{n} C_{2j} = \sum_{j=1}^{n}(C_{1j} + p_{2j}) = \sum_{j=1}^{n} C_{1j} + \sum_{j=1}^{n} p_{2j} = \sum_{j=1}^{n} C_{1j} + C$ where C is a constant. $\square$

Then, the minimisation of total flowtime on the second machine ($\sum_{j=1}^{n} C_{2j}$), goal of the $F2|prmu|\sum C_j$ problem, is equivalent to the minimisation of total flowtime on the first machine ($\sum_{j=1}^{n} C_{1j}$) which is the goal of the $1||\sum C_j$ problem of the first machine.

Corollary 3.5. *Under the conditions of Theorem 3.2, the optimal solutions for $\mathcal{I}$ and $\hat{\mathcal{I}}$ are the same where an optimal solution is obtained by sorting the jobs according to the non-decreasing processing times on the first machine.*

Proof. The proof of the theorem is obvious in view of Theorem 3.2 and taking into account that the non-decreasing sum of the processing times is an optimal sequence for the $1||\sum C_j$ problem. $\square$

For $m = 2$ and a more restrictive condition of processing times, this result is found by Ho and Gupta (1995):

Corollary 3.6. *(Ho and Gupta, 1995: Theorem 4 for $m = 2$). Let $\mathcal{I}$ be an instance of the $F2|prmu|\sum C_j$ problem where the machine 2 is type-II-dominated by the machine 1. Then, an optimal solution can be obtained by sorting the jobs in ascending order of their processing times on the first machine.*

Proof. The proof of the theorem is obvious in view of Corollary 3.5. $\square$

3.1.2 Second machine is saturated, $s = 2$

For the case where the second machine is saturated, the following property is needed to prove the equivalence between the problems:

Property 3.2. *Let $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ be a sequence of jobs with $p_{2\pi_{k-1}} \geq p_{1\pi_k}, \forall k \geq 2$. Then, the completion time of job π_k on the second machine is equal to its processing time plus the completion time of the previous job π_{k-1} on the second machine with the exception of the first job, i.e. $C_{2\pi_k} = C_{2\pi_{k-1}} + p_{2\pi_k}$, $\forall k \geq 2$, and $C_{2\pi_1} = p_{1\pi_1} + p_{2\pi_1}$.*

Proof. The proof of the property is obvious using the same reasoning as in Property 3.1. $\square$

This property extends the results by Ho and Gupta (1995) and Monma and Rinnooy Kan (1983), but the opposite cannot be asserted.

Corollary 3.7. *(Monma and Rinnooy Kan, 1983, second part of Theorem 3 for $m = 2$; and Ho and Gupta, 1995, Lemma 1 for $m = 2$). Let $\mathcal{I}$ be an instance of the PFSP where the machine 1 is type-II-dominated by the machine 2. Then, the completion time of each job on the second machine is equal to its processing time plus the completion time of the previous job (on the second machine), with the exception of the first job in the sequence which is equal to the sum of the processing times of this job on both machines.*

Proof. The proof of the corollary is obvious in view of Property 3.2. □

Corollary 3.8. *Let $\Pi := (\pi_1, \dots, \pi_k, \dots, \pi_n)$ be a sequence of jobs with $p_{1\pi_k} \leq p_{2\pi_{k-1}}, \forall k \geq 2$. Then, the idle time $IT_{2\pi_k}$ is equal to 0, i.e. $IT_{2\pi_k} = 0$, $\forall k \in 2 \dots n$.*

Proof. The proof of the corollary is obvious in view of Property 3.2 and taking into account the definition of idle time given in Equation (3). □

Then, when the first job of the sequence is fixed, the equivalence between $F2|prmu|C_{max}$ and $1||C_{max}$ is established in Theorem 3.3.

Theorem 3.3. *Let $\mathcal{I}$ be an instance of the $F2|prmu|C_{max}$ problem where machine 1 is dominated type I, and $\hat{\mathcal{I}}$ be an instance of the $1||C_{max}$ problem where $\hat{p}_j = p_{2j}$. Let Π be a sequence of the form $\Pi := \{f, \sigma\} := (f, \sigma_1, \dots, \sigma_k, \dots, \sigma_{n-1})$ where f is the first job of the sequence and σ is an unknown sequence of $n-1$ jobs. Let C_{max} be the makespan obtained by Π on instance $\mathcal{I}$ and $\hat{C}_{max}$ its makespan on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $C_{max} = \hat{C}_{max} + p_{1f}$.*

Proof. The proof of the theorem is obvious in view of Property 3.2 and Corollary 3.8 . Note that this theorem can also be proved using Theorem 3.1 together with the reversibility property of the $Fm|prmu|C_{max}$ problem (reverse problem, see Pinedo, 2012). □

Note that a consequence of this theorem is that the optimal solutions of both problems are the same as stated in the following corollary.

Corollary 3.9. *Under the conditions of Theorem 3.3, the optimal solutions for $\mathcal{I}$ and $\hat{\mathcal{I}}$ are the same. Additionally, any sequence of the form $\Pi := \{f, \sigma\}$ is an optimal sequence for both instances where f is the job with the least processing time on the second machine, i.e. $p_{1f} = \min_{\forall j} p_{1j}$.*

Proof. The proof of the theorem is obvious in view of Theorem 3.3 and taking into account that each feasible solution is an optimal sequence for the $1||C_{max}$ problem. □

A similar result is found by Ho and Gupta (1995) for both $m = 2$ and a more restrictive condition, but the opposite cannot be asserted.

Corollary 3.10. *(Ho and Gupta, 1995: Theorem 1 for $m = 2$). Let $\mathcal{I}$ be an instance of the $F2|prmu|C_{max}$ problem where the machine 1 is type-II-dominated by the machine 2. Then, any sequence of the form $\Pi := \{f, \sigma\}$ is optimal where σ is any sequence of $n - 1$ jobs and f satisfies $p_{1f} = \min_{\forall j} p_{1j}$.*

Proof. The proof of the theorem is obvious in view of Corollary 3.9. □

Additionally, the equivalence between $F2|prmu|\sum C_j$ and $1||\sum C_j$ is established in Theorem 3.4 for the case of a fixed first job in the sequence.

Theorem 3.4. *Let $\mathcal{I}$ be an instance of the $F2|prmu|\sum C_j$ problem where machine 1 is dominated type I, and $\hat{\mathcal{I}}$ be an instance of the $1||\sum C_j$ problem where $\hat{p}_j = p_{2j}$. Let Π be a sequence of the form $\Pi := \{f, \sigma\} := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-1})$ where f is the first job of the sequence and σ is an unknown sequence of $n - 1$ jobs. Let $S(\Pi)$ be the total flowtime obtained by Π on instance $\mathcal{I}$ and $\hat{S}(\Pi)$ its total flowtime on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $S(\Pi) = \hat{S}(\Pi) + n \cdot p_{1f}$.*

Proof. The proof of the theorem is obvious in view of Property 3.2 and Corollary 3.8. □

Corollary 3.11. *Under the conditions of Theorem 3.4 and considering f as a fixed job on the first position of the sequence, an optimal schedule is obtained by sorting the remaining jobs (sequence σ) according to the non-decreasing processing times on the second machine.*

Proof. The proof of the theorem is obvious in view of Theorem 3.4 and taking into account that each feasible solution is an optimal sequence for the $1||\sum C_j$ problem. □

For a more restrictive condition, the same result is found by Ho and Gupta (1995).

Corollary 3.12. *(Ho and Gupta, 1995: Theorem 3 for $m = 2$). Let $\mathcal{I}$ be an instance of the $F2|prmu|C_{max}$ problem where the machine 1 is type-II-dominated by the machine 2. Then, an optimal schedule $\Pi := \{f, \sigma\}$, where f is a fixed job on the first position of the sequence, is obtained by sorting the remaining jobs (sequence σ) according to the non-decreasing processing times on the second machine.*

Proof. The proof of the theorem is obvious in view of Corollary 3.11. □

Note that the conditions to reach the equivalence between the PFSP and the SMSP to minimise makespan and total flowtime can be reduced when the initial availabilities of machines are considered, see Theorem 3.5. In this case, both problems are equivalent regardless the sequence of jobs when the conditions are fulfilled.

Theorem 3.5. *Let $\mathcal{I}$ be an instance of the PFSP where machine 1 is dominated type I, and $\hat{\mathcal{I}}$ be an instance of the SMSP where $\hat{p}_j = p_{2j}$. Let a_2 be the initial availability of the second machine on both* *instances. Let δ_j be the difference between the processing time of job j on the first and second machines i.e. $\delta_j = p_{1j} - p_{2j}$. If*

$$a_2 \geq \sum_{\forall \delta_j > 0} \delta_j + \max_j \{p_{1j}\} \tag{4}$$

Then, the completion time of job π_k $\forall k > 1$, on the second machine is equal to its processing time plus the completion time of the previous job (i.e. $C_{2\pi_k} = C_{2\pi_{k-1}} + p_{2\pi_k} = a_2 + \sum_{j=1}^{k} p_{2\pi_j}$ $\forall k$, where $C_{2\pi_0} = a_2$) or, analogously, the idle time before job π_k is always equals to 0.

Proof. According to the definition of idle time, Expression (3), an idle time equals to 0 implies that $C_{2\pi_{k-1}} \geq C_{1\pi_k}$, $\forall k$.

For $k = 1$ (the first job in the sequence), the expression is $C_{2\pi_0} \geq C_{1\pi_1} \leftrightarrow a_2 \geq p_{1\pi_1}$ which is satisfied attending to Expression (4).

For $k = 2$, $C_{1\pi_2} = C_{1\pi_1} + p_{1\pi_2} = p_{1\pi_1} + p_{1\pi_2}$ and $C_{2\pi_1} = a_2 + p_{2\pi_1}$ if $IT_{i\pi_1} = 0$. Then, $IT_{i\pi_2} = 0 \leftrightarrow C_{2\pi_1} \geq C_{1\pi_2} \leftrightarrow a_2 + p_{2\pi_1} \geq p_{1\pi_1} + p_{1\pi_2} \leftrightarrow a_2 \geq p_{1\pi_1} + p_{1\pi_2} - p_{2\pi_1} \leftrightarrow a_2 \geq p_{1\pi_2} + \delta_{\pi_1}$. This condition is fulfilled according to Expression (4).

Analogously, for a generic $k = l$, the condition to reach an idle time equals to zero before π_l is $a_2 \geq p_{1\pi_l} + \sum_{j=1}^{l} \delta_{\pi_j}$. Since $\max_{\forall j}\{p_{1j}\} + \sum_{\forall \delta_j > 0} \delta_j \geq p_{1\pi_l} + \sum_{j=1}^{l} \delta_{\pi_j}$ and according to Expression (4), the previous condition is always satisfied. □

3.2 PFSP with m machines

Similar results of the equivalence between the PFSP and the SMSP can be found for a flowshop with more than two machines. In this paper, we detect four possible requirements to be fulfilled by an instance in order to achieve the equivalence between both problems.

3.2.1 Case *ddm*

Let us define some properties and corollaries before analysing the equivalence between the problems for the *ddm* dominance case.

Property 3.3. *Let $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ be a sequence of jobs with $p_{i\pi_k} \geq p_{i+1,\pi_{k-1}}, \forall k \geq 2$ and $\forall i > 1$. Then, the completion time of job π_k on the last machine equals its completion time on the first machine plus the sum of the processing times on the rest of machines, i.e.:*

$$C_{m\pi_k} = C_{1\pi_k} + \sum_{i=2}^{m} p_{i\pi_k}, \forall k$$

or equivalently:

$$C_{m\pi_k} = \sum_{j=1}^{k-1} p_{1\pi_j} + \sum_{i=1}^{m} p_{i\pi_k}, \forall k$$

Proof. The proof of the property is obvious applying recursively Property 3.1. □

The same result is also found by e.g. Cepek et al. (2002) and Wang et al. (2006) for a more restrictive condition of dominance:

Corollary 3.13. *(Cepek et al., 2002, Corollary 3.3; and Wang et al., 2006, Observation 2). Let $\mathcal{I}$ be an instance of the PFSP where each machine $i+1$ is type-II-dominated by machine i, $\forall i$. Then, the completion time of a job on the last machine can be defined as:*

$$C_{m\pi_k} = \sum_{j=1}^{k-1} p_{1\pi_j} + \sum_{i=1}^{m} p_{i\pi_k}, \forall k$$

Proof. The proof of the corollary is obvious in view of Property 3.3. □

Then, for this case of dominance, the equivalence between $Fm|prmu|\sum C_j$ and $1||\sum C_j$ is defined in Theorem 3.7. The equivalence between $Fm|prmu|C_{max}$ and $1||C_{max}$, when the last job of the sequence is fixed, is established in Theorem 3.6.

Theorem 3.6. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|C_{max}$ problem where the machines are dominated according to ddm, and $\hat{\mathcal{I}}$ be an instance of the $1||C_{max}$ problem where $\hat{p}_j = p_{1j}$. Let Π be a sequence of the form $\Pi := \{\sigma, g\} := (\sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-1}, g)$ where g is the last job of the sequence and σ is an unknown sequence of $n-1$ jobs. Let C_{max} be the makespan obtained by Π on instance $\mathcal{I}$ and $\hat{C}_{max}$ be its makespan on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $C_{max} = \hat{C}_{max} + \sum_{i=2}^{m} p_{ig}$.*

Proof. The proof of the theorem is obvious in view of Property 3.3 and Theorem 3.1. □

Theorem 3.7. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|\sum C_j$ problem where the machines are dominated according to ddm, and $\hat{\mathcal{I}}$ be an instance of the $1||\sum C_j$ problem where $\hat{p}_j = p_{1j}$. Let Π be a sequence of the form $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$. Let $S(\Pi)$ be the total flowtime obtained by Π on instance $\mathcal{I}$ and $\hat{S}(\Pi)$ be its total flowtime on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $S(\Pi) = \hat{S}(\Pi) + \sum_{j=1}^{n}\sum_{i=2}^{m} p_{ij}$.*

Proof. The proof of the theorem is obvious in view of Property 3.3 and Theorem 3.2. □

3.2.2 Case *idm*

For the *idm* case, the following property establishes the value of the completion time of each job on the last machine:

Property 3.4. *Let* $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ *be a sequence of jobs with* $p_{i\pi_{k-1}} \geq p_{i-1,\pi_k}, \forall k \geq 2$ *and* $\forall i > 1$. *Then, the completion time of job* π_k *on the last machine is equal to its processing time plus the completion time of the previous job* π_{k-1} *on the last machine, with the exception of the first job, i.e.* $C_{m\pi_k} = C_{m\pi_{k-1}} + p_{m\pi_k}$, $\forall k \geq 2$, *and* $C_{m\pi_1} = \sum_{i=1}^{m} p_{i\pi_1}$. *Equivalently,*

$$C_{m\pi_k} = \sum_{i=1}^{m-1} p_{i\pi_1} + \sum_{j=1}^{k} p_{m\pi_j}, \forall k$$

Proof. The proof of the property is obvious applying recursively Property 3.2. □

For more restrictive conditions, the same result is found by e.g. Cepek et al. (2002), Ho and Gupta (1995) and Wang et al. (2006).

Corollary 3.14. *(Ho and Gupta, 1995, Lemma 1; Cepek et al., 2002, Corollary 3.1; and Wang et al., 2006, Observation 1). Let* $\mathcal{I}$ *be an instance of the PFSP where each machine* i *is type-II-dominated by machine* $i + 1$. *Then, the completion time of a job on the last machine can be defined as:*

$$C_{m\pi_k} = \sum_{i=1}^{m-1} p_{i\pi_1} + \sum_{j=1}^{k} p_{m\pi_j}, \forall k$$

Proof. The proof of the corollary is obvious in view of Property 3.4. □

Additionally, Property 3.4 implies that there is no idle time on the last machine after the first job of the sequence:

Corollary 3.15. *Let* $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ *be a sequence of jobs and* $\mathcal{I}$ *be an instance of the PFSP where the machines are dominated according to idm. Then, the idle time* $IT_{m\pi_k}$ *is equal to 0,* $\forall k > 1$.

Proof. The proof of the corollary is obvious in view of Property 3.4 and taking into account the definition of idle time given in Equation (3). □

The equivalence between $Fm|prmu|C_{max}(\sum C_j)$ and $1||C_{max}(\sum C_j)$, when the first job of the sequence is fixed, is established in Theorem 3.8 (3.9).

Theorem 3.8. *Let* $\mathcal{I}$ *be an instance of the* $Fm|prmu|C_{max}$ *problem where the machines are dominated according to idm, and* $\hat{\mathcal{I}}$ *be an instance of the* $1||C_{max}$ *problem where* $\hat{p}_j = p_{mj}$. *Let* Π *be a sequence of the form* $\Pi := \{f, \sigma\} := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-1})$ *where* f *is the first job of the sequence and* σ *is an unknown sequence of* $n - 1$ *jobs. Let* C_{max} *be the makespan obtained by* Π *on instance* $\mathcal{I}$ *and* $\hat{C}_{max}$ *its makespan on instance* $\hat{\mathcal{I}}$. *Then, for each feasible sequence,* $C_{max} = \hat{C}_{max} + \sum_{i=1}^{m-1} p_{if}$.

Proof. The proof of the theorem is obvious in view of Property 3.4 and Corollary 3.15. □

Theorem 3.9. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|\sum C_j$ problem where the machines are dominated according to idm, and $\hat{\mathcal{I}}$ be an instance of the $1||\sum C_j$ problem where $\hat{p}_j = p_{mj}$. Let Π be a sequence of the form $\Pi := \{f, \sigma\} := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-1})$ where f is the first job of the sequence and σ is an unknown sequence of $n-1$ jobs. Let $S(\Pi)$ be the total flowtime obtained by Π on instance $\mathcal{I}$ and $\hat{S}(\Pi)$ its total flowtime on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence, $S(\Pi) = \hat{S}(\Pi) + n \cdot \sum_{i=1}^{m-1} p_{if}$.*

Proof. The proof of the theorem is obvious in view of Property 3.4 and Corollary 3.15. □

3.2.3 Case *idm-ddm*

Similar to the previous case, the completion time of each job on the last machine is defined by the following property for the *idm-ddm* case:

Property 3.5. *Let $\Pi := (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$ be a sequence of jobs with $p_{i_1\pi_{k-1}} \geq p_{i_1-1,\pi_k}, \forall k \geq 2, s \geq i_1 > 1$ and $p_{i_2\pi_k} \geq p_{i_2+1,\pi_{k-1}}, \forall k \geq 2, i_2 \geq s$, i.e. following a dominance configuration type idm-ddm where the saturated machine is s. Then, the completion time of job π_k on the last machine is:*

$$C_{m\pi_k} = \sum_{i_1=1}^{s-1} p_{i_1,\pi_1} + \sum_{j=1}^{k} p_{s\pi_j} + \sum_{i_2=s+1}^{m} p_{i_2\pi_k}, \forall k$$

Proof. The proof of the property is obvious in view of Properties 3.3 and 3.4. □

For a more restrictive condition, the same result is found by Wang et al. (2006).

Corollary 3.16. *(Wang et al., 2006, Observation 4). Let $\mathcal{I}$ be an instance of the PFSP where each machine $i_1 < s$ is type-II-dominated by machine $i_1 + 1$ as well as each machine $s < i_2 \leq m$ is type-II-dominated by machine $i_2 - 1$. Then, the completion time of a job on the last machine can be defined as:*

$$C_{m\pi_k} = \sum_{i_1=1}^{s-1} p_{i_1\pi_1} + \sum_{j=1}^{k} p_{s\pi_j} + \sum_{i_2=s+1}^{m} p_{i_2\pi_k}, \forall k$$

Proof. The proof of the corollary is obvious in view of Property 3.5. □

Then, for this case of dominance, the equivalence between $Fm|prmu|\sum C_j$ and $1||\sum C_j$ is defined in Theorem 3.11 by fixing the first job of the sequence as well as the equivalence between $Fm|prmu|C_{max}$ and $1||C_{max}$, when the first and the last job of the sequence are fixed, is established in Theorem 3.10.

Theorem 3.10. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|C_{max}$ problem where the machines are dominated according to idm-ddm, and $\hat{\mathcal{I}}$ be an instance of the $1||C_{max}$ problem where $\hat{p}_j = p_{sj}$.Let Π be a sequence of the form $\Pi := \{f, \sigma, g\} := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-2}, g)$ where f and g are respectively the first and last* *job of the sequence and σ is an unknown sequence of $n-2$ jobs. Let $S(\Pi)$ be the total flowtime obtained by Π on instance $\mathcal{I}$ and $\hat{S}(\Pi)$ its total flowtime on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence,*

$$C_{max} = \hat{C}_{max} + \sum_{i_1=1}^{s-1} p_{i_1 f} + \sum_{i_2=s+1}^{m} p_{i_2 g}$$

Proof. The proof of the theorem is obvious in view of Property 3.5. □

Theorem 3.11. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|\sum C_j$ problem where the machines are dominated according to idm-ddm, and $\hat{\mathcal{I}}$ be an instance of the $1||\sum C_j$ problem where $\hat{p}_j = p_{sj}$.Let Π be a sequence of the form $\Pi := \{f, \sigma\} := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-1})$ where f is respectively the first job of the sequence and σ is an unknown sequence of $n-1$ jobs. Let C_{max} be the makespan obtained by Π on instance $\mathcal{I}$ and $\hat{C}_{max}$ its makespan on instance $\hat{\mathcal{I}}$. Then, for each feasible sequence,*

$$S(\Pi) = \hat{S}(\Pi) + \sum_{j=1}^{n} \sum_{i_2=s+1}^{m} p_{i_2 j} + n \cdot \sum_{i_1=1}^{s-1} p_{i_1 f}$$

Proof. The proof of the theorem is obvious in view of Property 3.5. □

3.2.4 Generic Case

The assumptions to achieve this equivalence are much harder in the generic case of a flowshop with $m > 2$ machines. In fact, it is necessary that the processing time of each job j on the saturated machine s is higher than both the sum of the processing times on the machines before s of each job $j^{'} \neq j$, and the sum of the processing times on the machines after s of each job $j^{'} \neq j$. Obviously, this behaviour is hardly found in real-life environments. It thus represents only a sufficient but not necessary condition to state the equivalence.

Theorem 3.12. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|C_{max}$ problem with $p_{sj} \geq \sum_{i=1}^{s-1} p_{ij'}$ and $p_{sj} \geq \sum_{i=s+1}^{m} p_{ij'}$, $\forall j \neq j^{'}$, and $\hat{\mathcal{I}}$ be an instance of the $1||C_{max}$ problem where $\hat{p}_j = p_{sj}$. Let f and g be the fixed first and last job of a sequence of jobs $\Pi := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-2}, g)$ where σ is an unknown sequence of $n-2$ jobs. Then, $Fm|prmu|C_{max}$ is equivalent to $1||C_{max}$ of machine s.*

Proof. The proof is obvious using the same reasoning as Properties 3.1 and 3.2. □

Theorem 3.13. *Let $\mathcal{I}$ be an instance of the $Fm|prmu|\sum C_j$ problem with $p_{sj} \geq \sum_{i=1}^{s-1} p_{ij'}$ and $p_{sj} \geq \sum_{i=s+1}^{m} p_{ij'}$, $\forall j \neq j^{'}$, and $\hat{\mathcal{I}}$ be an instance of the $1||\sum C_j$ problem where $\hat{p}_j = p_{sj}$. Let f and g be the fixed first and last job of a sequence of jobs $\Pi := (f, \sigma_1, \ldots, \sigma_k, \ldots, \sigma_{n-2}, g)$ where σ is an unknown sequence of $n-2$ jobs. Then, $Fm|prmu|\sum C_j$ is equivalent to $1||\sum C_j$ of machine s.*

Proof. The proof is obvious using the same reasoning as Properties 3.1 and 3.2. □

All the properties presented in Sections 3.1 and in this section analyse the assumptions required to theoretically prove the equivalence between the PFSP and the SMSP of the saturated machine for the minimisation of makespan and total flowtime. The equivalence between both problems is theoretically proved in Theorems 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13 for different conditions. Given an instance, the fulfillment of these conditions then indicates that solving the equivalent SMSP is analogous to solving the original PFSP. A summary of the conditions is shown in Table 2. However, this equivalence could be approximately satisfied under milder conditions. In the next section, we empirically analyse them by means of extensive computational experiments.

Table 2: Summary of conditions to reduce $Fm|prmu|C_{max}(\sum C_j)$ to $1||C_{max}(\sum C_j)$

#Machines	Subcase	Makespan	Total Flowtime
2	s=1	Machine 2 is dominated type I Fixed last job	Machine 2 is dominated type I
	s=2	Machine 1 is dominated type I Fixed first job	Machine 1 is dominated type I Fixed first job
	s=1 or 2	$a_2 \geq \sum_{\forall \delta_j > 0} \delta_j + \max_j\{p_{1j}\}$	$a_2 \geq \sum_{\forall \delta_j > 0} \delta_j + \max_j\{p_{1j}\}$
m	Generic	$p_{sj} \geq \sum_{i=1}^{s-1} p_{ij'}$ and $p_{sj} \geq \sum_{i=s+1}^{m} p_{ij'}, \forall j \neq j'$ Fixed first and last job	$p_{sj} \geq \sum_{i=1}^{s-1} p_{ij'}$ and $p_{sj} \geq \sum_{i=s+1}^{m} p_{ij'}, \forall j \neq j'$ Fixed first and last job
	Subcase 1	Case DDM Fixed last job	Case DDM
	Subcase 2	Case IDM Fixed first job	Case IDM Fixed first job
	Subcase 3	Case IDM-DDM Fixed first and last job	Case IDM-DDM Fixed first job

4 Empirical Analysis

This section is organised as follows: in Section 4.1, the procedure to generate the instances is described. The implemented heuristics are briefly explained in Section 4.2. In Section 4.3, the indicators to measure both the quality of the solutions and the computational effort are presented. Results for several values of the parameters of the testbed are shown in Section 4.4. The boundary lines between both problems are further analysed in Section 4.5. Finally, both problems are also compared under the set of instances by Watson et al. (2002) in Section 4.6.

4.1 Testbed generation

Using the above formulations and definitions, existing testbeds for the PFSP can be analysed. As mentioned in Section 1, most algorithms for the PFSP have been tested on benchmarks where the processing times follow a uniform distribution. However, in the experiments by Watson et al. (2002), a structured benchmark with job-correlated, machine-correlated and mixed-correlated processing times is employed. Regarding machine-correlation, processing times are generated using a uniform distribution considering the following two aspects:

- For each machine, the mean of the processing times is generated from 1 to 100 depending on a parameter.
- For each machine, the width of the uniform distribution is uniformly generated from 2 to 10.

The goal of this paper is to show that the PFSP with a saturated machine is equivalent to a SMSP. The study of this equivalence must obviously be done without the influence of additional effects and therefore, a specific benchmark is generated to test the experiments performed in Section 4, where different levels for the saturated machine are considered. Firstly, the same distribution is considered for each non-saturated machine since it is the worst case. Additionally, the consideration of different distributions in the non-saturated machines would strongly hinder the understanding on the cause of the equivalence between the problems. Secondly, the same width of the uniform distribution (or, equivalently, its variance) is considered for all machines (including the most saturated one) both for clarity and to reduce the parameters of the proposed benchmark.

Taking into account the previous discussion, the processing times for our computational experiments are then generated according to Expression (5) for the non-saturated machines and Expression (6) for the saturated machine s:

$$p_{ij} \rightarrow U[\epsilon \cdot (1-\beta), \epsilon \cdot (1+\beta)], \forall i \neq s, j \in 1, \ldots, n \tag{5}$$

$$p_{sj} \rightarrow U[\epsilon \cdot (1+\gamma-\beta), \epsilon \cdot (1+\gamma+\beta)], j \in 1, \ldots, n \tag{6}$$

where the following parameters must be defined:

- ϵ: Mean of the processing times on non-saturated machines.
- β: Half length of the interval of the uniform distribution of each machine with respect to the mean processing time ϵ, i.e. $\epsilon\beta$ yields the half width of the interval, and $2 \cdot \epsilon \cdot \beta$ is the full length of the interval of the uniform distribution of each machine (including the saturated machine).
- γ: Increase of the mean of the processing times on the saturated machine s relative to ϵ. In this way, $(1+\gamma)\epsilon$ represents the expected processing times on machine s whereas the expected value of the processing time for the rest of the machines is ϵ.

Regarding the initial availability of the machines, the following parameter must be considered in the benchmark:

- δ: Number of jobs being processed in the shop floor to create a fictitious initial unavailability. More specifically, we generate δ jobs, which are sequenced according to certain heuristic (in this paper, we use the NEH of Nawaz et al., 1983 and Framinan et al., 2002 for makespan and flowtime minimisation respectively). The processing of these jobs according to such sequence creates a_i, the initial unavailability on each machine i, which can be computed as follows: $a_i = C_{i,\pi_n} - C_{1,\pi_n}$.

4.2 Implemented heuristics

The following simple 11 algorithms are implemented to analyse the relationship between the SMSP and the PFSP. More specifically, we design the following procedures:

- PF_B(i) ($i \in [MK, FT]$) is designed to provide good –hopefully the best– solutions for the PFSP with makespan and flowtime objective, respectively. In this paper, we use the NEH heuristic of Nawaz et al. (1983) for makespan and total flowtime minimisation as PF_B(MK) and PF_B(FT), respectively. Note that the NEH heuristic initially sorts the jobs according to non-increasing sum of the processing times. Then, one by one, each job of the initial order is tested in each position of an initially empty partial sequence. The partial sequence that minimises the makespan, for PF_B(MK), and the total flowtime, for PF_B(FT), is chosen in each iteration.
- PF_W(i) ($i \in [MK, FT]$) is designed to obtain bad –hopefully the worst– solutions for the PFSP with makespan and flowtime objective, i.e. we seek for makespan and flowtime *maximization*. Analogously, we use the NEH heuristic for makespan and total flowtime *maximisation*, i.e. we use the decreasing sum of the processing times as initial order and keep in each iteration the partial sequence that maximises the makespan and the total flowtime respectively.
- SM_R(MK) and SM_B(FT) are designed to provide the best solutions for the equivalent SMSP if only the saturated machine in the PFSP is considered. The idea is to compare the sequence obtained by heuristics designed for the PFSP with heuristics designed for the SMSP on the saturated machine. On the one hand, in SM_R(MK), jobs are simply sorted according to sequence $(1, \ldots, n)$, which is a random solution. Since, for the SMSP with makespan objective, each solution is optimal, this procedure would yield both the best and worst solutions for this problem. On the other hand, in SM_B(FT), jobs are sorted in non-decreasing order of their processing times on machine s, which corresponds to the optimal solutions of the equivalent $1||\sum C_j$ problem.

- SM_W(FT) is designed to provide bad –hopefully the worst– solutions for the equivalent SMSP with total flowtime objective if only the saturated machine in the PFSP is considered (note that the corresponding procedure for makespan would be also SM_R(MK)). To do so, jobs are sorted in non-increasing processing times on machine s, which provides the worst solution for the equivalent $1||\sum C_j$ problem.

- SM_EB(i) ($i \in [MK, FT]$) is designed to provide good –hopefully the best– solutions for the equivalent SMSP considering the saturated machine in the PFSP and the influence of the last and first jobs in this machine. In Section 3, it was shown that, under certain conditions, the PFSP and the SMSP are equivalent when both the first and the last job of the sequences are fixed. Therefore, the idea behind these methods is to solve the equivalent SMSP taking into account the influence of the first and the last job of the sequence on the non-saturated machines. Thereby, SM_EB(MK) reduces the $Fm|prmu|C_{max}$ problem to the $1||C_{max}$ problem where the processing times of the first and the last job are equivalent to their original processing times plus the processing times of the machines before and after the saturated machines, i.e. given a sequence Π of jobs, the processing times are:

$$p'_{\pi_k} = \begin{cases} p_{s\pi_k} + \sum_{i=1}^{s-1} p_{i\pi_k}, & k = 1 \\ p_{s\pi_k}, & \forall k \neq 1, n \\ p_{s\pi_k} + \sum_{i=s}^{m} p_{i\pi_k}, & k = n \end{cases}$$

  To find a good Π^f final sequence, as in SM_R(MK), jobs are first sorted randomly (let us denoted Π^R to this sequence). Then, two simple phases are carried out as follows to find the first and the last job:

  - For $\delta = 0$, the first job of the sequence is the job with minimal sum of processing times before machine s, i.e. π_1^f is the job F satisfying that $\sum_{i=1}^{s-1} p_{iF} \leq \sum_{i=1}^{s-1} p_{ij}$, $\forall j$. For $\delta > 0$, the first job is the same as in SM_R(MK), $\pi_1^f = \pi_1^R$.
  - The last job of the sequence is the job with minimal sum of processing times after machine s, i.e. π_n^f is the job L which satisfies that $\sum_{i=s+1}^{m} p_{iL} \leq \sum_{i=s+1}^{m} p_{ij}$, $\forall j$.

  The pseudo-code of SM_EB(MK) is shown in Figure 1.

```
$\Pi^R := (\pi_1^R, ...\pi_n^R)$ where $\pi_i^R = i$, $\forall i \in [1,n]$;
if $\delta = 0$ then
    Determine job $F$ which satisfies $\sum_{i=1}^{s-1} p_{iF} \leq \sum_{i=1}^{s-1} p_{ij}$, $\forall j$;
else
    $F$:= null;
end
Determine job $L$ which satisfies $\sum_{i=s+1}^{m} p_{iL} \leq \sum_{i=s+1}^{m} p_{ij}$, $\forall j$;
$\Pi^f = (F, \Pi^R - \{F, L\}, L)$;
```

Figure 1: Heuristic SM_EB(MK).

Regarding flowtime, SM_EB(FT) solves the equivalent SMSP as SM_B(FT). However, in contrast to that heuristic, SM_EB(FT) considers the influence of the first job on the machines before the saturated machine, i.e. $i < s$. Thereby, the processing time of the first job is the sum of the processing times on machines $i \leq s$, i.e. given a sequence Π of jobs, their processing times are:

$$p'_{\pi_k} = \begin{cases} p_{s\pi_k} + \sum_{i=1}^{s-1} p_{i\pi_k}, & k = 1 \\ p_{s\pi_k}, & \forall k > 1 \end{cases}$$

The procedure of the heuristic consists of two phases: in the first phase, jobs are sorted according to non-decreasing processing times on machine s; in the second phase, the first job of the sequence is the job with minimal sum of processing times up to machine s, i.e. π_1^f is the job F which satisfies that $\sum_{i=1}^{s} p_{iF} \leq \sum_{i=1}^{s} p_{ij}$, $\forall j$. See pseudo-code in Figure 2.

```
$\Pi^i := (\pi_1^i, ...\pi_n^i)$ Jobs ordered by non-decreasing $p_{sj}$ (Phase I);
Determine job $F$ which satisfies $\sum_{i=1}^{s} p_{iF} \leq \sum_{i=1}^{s} p_{ij}$, $\forall j$ (Phase II);
$\Pi^f = (F, \Pi^i - \{F\})$;
```

Figure 2: Heuristic SM_EB(FT).

- M_B(i) ($i \in [MK, FT]$) is designed to provide good –hopefully the best– solutions for a "reduced" PFSP formed by machines $i' \in \{s, m\}$ for makespan and flowtime objectives, respectively. These heuristics use PF_B(i) ($i \in [MK, FT]$) to solve a instance $\mathcal{I}'$ of the "reduced" PFSP with $m - s$ machines (i.e. $i' \in \{s, m\}$), and processing times defined by $p'_{i'j} = p_{i'j}$, $\forall i' \in \{s, m\}$. The operations of the jobs in the rest of the machines are omitted. Note that the initial availabilities a_i are calculated using the saturated machine, i.e. $a_{i'} = C_{i'\pi_n} - C_{s\pi_n}$, $\forall i' \in \{s, m\}$.

With these procedures we can check the statistical equivalence between the SMSP and the PFSP on a set of instances, since, for the cases where the objective function values found by SM_B(FT), SM_R(MK), SM_EB(i) and M_B(i) are close to those provided by PF_B(i), and those found by SM_W(FT) and PF_W(i) are similar, then both problems (PFSP and SMSP) are (approximately) equivalent.

4.3 Evaluation of the solutions

Traditionally, the related literature employs the Relative Percentage Deviation (RPD) and the CPU time to measure both the quality of the solution and the required computational effort of heuristic r in an instance $\mathcal{I} \in \mathcal{V}$. More specifically, the average RPD ($ARPD$) and the average CPU time ($ACPU$) obtained by heuristic r over a set $\mathcal{V}$ can be defined as follows:

$$ARPD_r = \frac{\sum_{\mathcal{I} \in \mathcal{V}} RPD_r(\mathcal{I})}{|\mathcal{V}|} \tag{7}$$

$$ACPU_r = \frac{\sum_{\mathcal{I} \in \mathcal{V}} T_r(\mathcal{I})}{|\mathcal{V}|} \tag{8}$$

where

$$RPD_r(\mathcal{I}) = \frac{OFV_r(\mathcal{I}) - Best(\mathcal{I})}{Best(\mathcal{I})} \cdot 100 \tag{9}$$

$OFV_r(\mathcal{I})$ is the objective function value (makespan or total flowtime) obtained by heuristic r in instance $\mathcal{I}$. $Best(\mathcal{I})$ is the best solution among the implemented heuristics for that instance, i.e. $Best(\mathcal{I}) := \min_r OFV_r(\mathcal{I})$. Finally, $T_r(\mathcal{I})$ is the CPU time of heuristic r for instance $\mathcal{I}$.

The consideration of initial availability introduces a distortion in the evaluation of the objective function which must be taken into account. This distortion is illustrated with the following example: Let us assume a PFSP with two machines and two jobs. Processing times of the first and second jobs on the machines are $p_{11} = 10$, $p_{21} = 40$, and $p_{12} = 10$, $p_{22} = 50$, respectively. For the two possible sequences i.e. $\pi^1 = (1,2)$ and $\pi^2 = (2,1)$, the total flowtimes are $\sum C_{m,\pi^1_j} = 150$ and $\sum C_{m,\pi^2_j} = 160$. In terms of RPD, $RPD(\sum C_{m,\pi^1_j}) = 0$ and $RPD(\sum C_{m,\pi^2_j}) = 6.67$. Let us now assume that the second machine is not available until time 300. Then, the total flowtime of both sequences change to $\sum C_{m\pi^1_j} = 730$ and $\sum C_{m\pi^2_j} = 740$ respectively, while RPD are $RPD(\sum C_{m\pi^1_j}) = 0$ and $RPD(\sum C_{m\pi^2_j}) = 1.37$. Although in this case the initial availability of the second machine clearly does not influence the hardness of the problem, its influence in the $RPDs$ is very high.

To avoid this issue, we do not consider the time 0 as reference for the completion times. Instead, we consider a reference (denoted as B) based on the first job that is scheduled in the flowshop. Nevertheless, in order not to have a sequence-dependent reference, we consider as the first job to be scheduled an artificial job consisting of sequencing all jobs and then average their completion times, i.e. $B = \frac{\sum_{j=1}^{n} C_{m\pi_1=j}}{n}$. Once B is obtained, the completion time of each job on the last machine is reduced by B time units for each heuristic.

4.4 Computational Results

All algorithms are coded in the same language (C# under Visual Studio 2013) and under an Intel Core i7-3770 with 3.4 GHz and 16 GB RAM. They are tested on an set of instances following the indications of Section 4.1, which includes $n \in \{20, 50, 100, 200\}$, $m \in \{2, 5, 10, 20\}$ and two values of $\delta \in \{0, 100\}$ representing an initially empty and loaded shop respectively. Processing times are generated according to the expression (5) and (6) with the following parameters:

- $\epsilon = 50$.
- $\beta \in \{0.10, 0.20, 0.40, 0.60, 1.00\}$.
- $\gamma \in \{0.00, 0.04, 0.08, \ldots, 2.96, 3.00\}$.

For each combination of parameters (n, m, δ, ϵ, β and γ), 10 instances are generated forming a total of 121,600 instances.

The values of $ARPD$ of the heuristics with initial availabilities ($\delta = 100$) and without initial availabilities ($\delta = 0$) are shown in Table 3 for each value of n, m and β, and for some values of γ. Each row represents the average RPD for the parameter of the first column, e.g. the value 7.84 (of the second row and second column) is the average $RPD_{SM\_R(MK)}(\mathcal{I})$, $\forall \mathcal{I} \mid \gamma = 0.00$. Clearly, the heuristics SM_B(FT), SM_R(MK), SM_EB(i) and M_B(i) are closer to PF_B(i) when β and m decrease, and γ and n increase. Figures 3 and 4 show the $ARPD$ of the heuristics for the complete set of values of γ and δ, as well as the decreasing trend of each curve for makespan and total flowtime minimisation respectively. Several aspects can be highlighted from the results:

Table 3: $ARPD$ of the heuristics

		$ARPD$ of heuristics for makespan minimisation					$ARPD$ of heuristics for total flowtime minimisation					
		SM_R(MK)	SM_EB(MK)	PF_B(MK)	PF_W(MK)	M_B(MK)	SM_R(FT)	SM_EB(FT)	PF_B(FT)	PF_W(FT)	SM_W(FT)	M_B(FT)
$\delta = 0$	$\gamma = 0.00$	7.84	7.32	0.00	28.64	8.35	11.10	10.51	0.04	35.14	20.11	8.26
	$\gamma = 0.20$	5.48	4.41	0.01	20.88	4.21	7.30	6.65	0.06	31.79	19.50	5.04
	$\gamma = 0.40$	3.80	2.69	0.00	13.95	2.54	4.99	4.30	0.11	27.57	18.19	3.74
	$\gamma = 0.60$	3.09	1.81	0.00	9.68	1.89	3.59	2.79	0.13	23.97	16.69	2.81
	$\gamma = 0.80$	2.63	1.27	0.01	6.93	1.53	2.53	1.76	0.16	21.17	15.21	1.97
	$\gamma = 1.00$	2.20	0.97	0.00	5.28	1.09	1.87	1.10	0.18	18.90	13.80	1.57
	$\gamma = 2.00$	1.30	0.32	0.00	2.62	0.61	1.00	0.21	0.22	12.32	9.36	1.00
	$\beta = 0.10$	0.36	0.06	0.00	0.80	0.18	0.30	0.07	0.08	3.16	2.38	0.28
	$\beta = 0.20$	0.81	0.23	0.00	1.93	0.43	0.72	0.27	0.15	6.85	5.08	0.64
	$\beta = 0.40$	1.82	0.79	0.00	4.77	1.04	1.72	0.96	0.23	14.50	10.45	1.41
	$\beta = 0.60$	2.96	1.56	0.00	8.33	1.83	3.03	2.03	0.25	22.64	16.05	2.42
	$\beta = 1.00$	5.49	3.52	0.00	16.97	3.73	6.30	4.99	0.24	39.74	27.19	4.74
	$m = 2$	0.53	0.13	0.00	2.21	0.16	0.41	0.40	0.04	15.29	12.64	0.21
	$m = 5$	1.70	0.67	0.00	5.61	1.12	1.93	1.27	0.30	16.78	11.84	1.55
	$m = 10$	2.84	1.52	0.00	8.03	1.89	3.07	2.04	0.25	18.09	11.98	2.46
	$m = 20$	4.09	2.61	0.00	10.38	2.58	4.25	2.94	0.17	19.36	12.46	3.38
	$n = 20$	4.63	2.29	0.01	10.19	2.56	4.03	2.34	0.32	19.23	14.60	3.34
	$n = 50$	2.37	1.32	0.00	6.61	1.49	2.49	1.76	0.20	17.29	12.25	1.97
	$n = 100$	1.37	0.83	0.00	5.14	1.01	1.80	1.42	0.14	16.66	11.31	1.34
	$n = 200$	0.78	0.49	0.00	4.30	0.71	1.33	1.14	0.10	16.34	10.75	0.95
$\delta = 100$	$\gamma = 0.00$	3.47	3.32	0.02	16.21	3.36	8.34	8.34	0.11	27.57	16.71	2.61
	$\gamma = 0.20$	3.30	2.71	0.04	14.82	0.64	4.09	4.09	0.12	28.09	17.18	0.74
	$\gamma = 0.40$	2.33	1.69	0.03	10.01	0.07	1.69	1.69	0.09	25.38	16.27	0.26
	$\gamma = 0.60$	1.76	1.11	0.01	6.50	0.03	0.75	0.75	0.06	22.10	14.70	0.14
	$\gamma = 0.80$	1.48	0.79	0.00	4.45	0.02	0.40	0.40	0.06	19.30	13.02	0.07
	$\gamma = 1.00$	1.25	0.51	0.01	3.20	0.01	0.25	0.25	0.04	17.19	11.76	0.04
	$\gamma = 2.00$	0.74	0.14	0.00	1.44	0.00	0.05	0.05	0.00	10.72	7.58	0.00
	$\beta = 0.10$	0.20	0.03	0.00	0.47	0.01	0.04	0.04	0.00	2.74	1.92	0.01
	$\beta = 0.20$	0.46	0.11	0.00	1.16	0.03	0.12	0.12	0.01	5.96	4.14	0.03
	$\beta = 0.40$	1.02	0.41	0.00	2.96	0.09	0.47	0.47	0.02	12.75	8.67	0.11
	$\beta = 0.60$	1.64	0.84	0.01	5.25	0.17	0.98	0.98	0.04	19.94	13.39	0.21
	$\beta = 1.00$	3.09	2.00	0.03	11.12	0.48	2.57	2.57	0.11	35.63	23.41	0.54
	$m = 2$	0.49	0.10	0.00	2.02	0.10	0.40	0.40	0.04	15.09	12.60	0.13
	$m = 5$	0.97	0.38	0.00	3.54	0.13	0.67	0.67	0.02	15.32	10.46	0.19
	$m = 10$	1.48	0.78	0.01	4.85	0.17	0.94	0.94	0.03	15.51	9.42	0.18
	$m = 20$	2.20	1.45	0.02	6.35	0.22	1.33	1.33	0.05	15.70	8.76	0.22
	$n = 20$	2.59	1.26	0.02	6.15	0.15	1.20	1.20	0.05	15.28	10.42	0.18
	$n = 50$	1.31	0.71	0.01	4.20	0.14	0.85	0.85	0.04	15.41	10.33	0.17
	$n = 100$	0.78	0.46	0.00	3.43	0.15	0.70	0.70	0.03	15.42	10.26	0.18
	$n = 200$	0.46	0.29	0.00	2.98	0.18	0.60	0.60	0.02	15.50	10.21	0.18

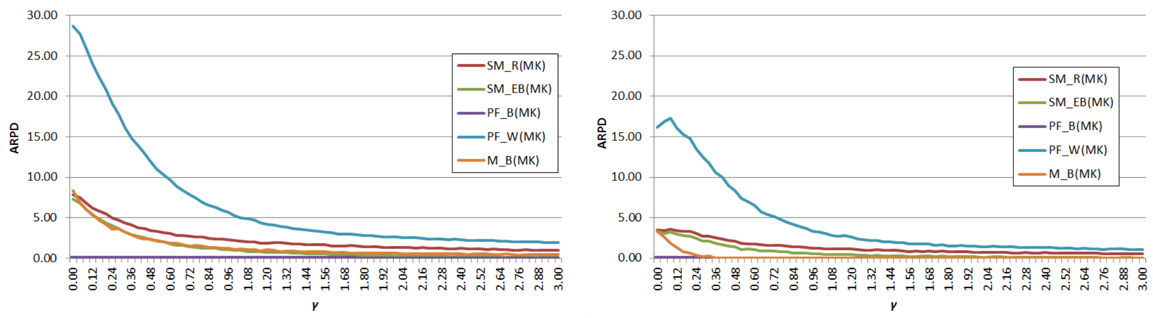

Figure 3: $ARPD$ of the heuristics versus γ for makespan minimisation. On the left, no initial availability is considered and on the right an initial $\delta = 100$ is taken into account.

- For high values of γ ($\gamma \gtrsim 1.00$), solving the equivalent SMSP or the original PFSP yields a similar solution (i.e. the $ARPD$ obtained by SM_B(FT), SM_R(MK), SM_EB(i) and SM_B(FT) are very close to the $ARPD$ by PF_B(i)).
- Additionally, for high values of γ ($\gamma \gtrsim 1.00$), the worst solutions found for the $Fm|prmu|\sum C_j$ problem by PF_W(FT) yield similar total flowtime to that of the solutions found for the equivalent $1||\sum C_j$ problem by SM_W(FT).
- The $ARPD$ found by SM_R(MK) for the equivalent $1||C_{max}$ problem is always between the best and worst $ARPD$ found by PF_B(MK) and PF_W(MK). The distance between the three curves sharply decreases as γ increases, which explains the trivial behaviour of the $Fm|prmu|C_{max}$ problem for these cases.
- The initial availability δ has a strong influence on the $ARPD$ as seen in Figures 3 and 4. Thereby, e.g. the $ARPD$ of SM_EB(i) becomes close to that of PF_B(i) from $\gamma \gtrsim 0.30$ regardless the other parameters (m, n or β).

According to the dominance rules in Section 3.1 and 3.2, the number of machines and the bounds of the processing times play an essential role in the comparison between PFSP and SMSP. This influence is also empirically shown in this section. Thereby, Figure 5 shows the evolution of the $ARPD$ with γ for different number of machines. The $ARPD$ curves clearly decrease with the decrease of the number of machines in the shop. Note that, for clarity, only the SM_EB(i) heuristics are represented although the behaviour is similar for the other heuristics. Since the $ARPDs$ for the PF_B(i) heuristics are always approximately zero, values closes to zero for SM_EB(i) indicate the similarities between both the PFSP and the SMSP. Thereby, e.g. it can be seen that the $ARPD$ of SM_EB(MK) for $\delta = 100$ is always less than 1, regardless the value of γ.

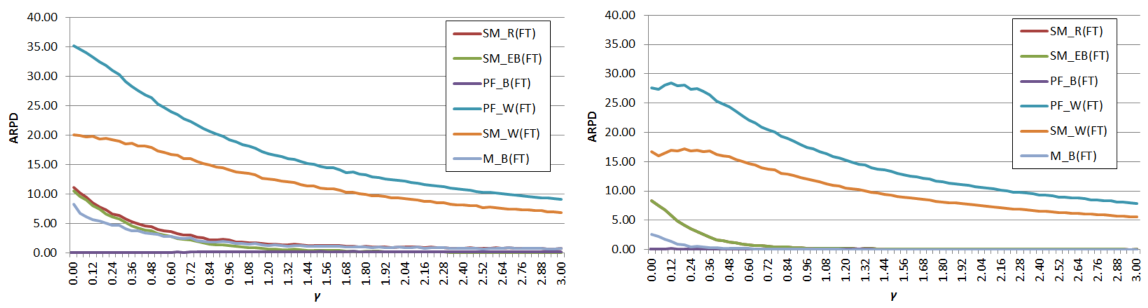

Figure 4: *ARPD* of the heuristics versus γ for total flowtime minimisation. On the left, no initial availability is considered and on the right an initial $\delta = 100$ is taken into account. Note that the curve SM_R(FT) is exactly over the curve SM_EB(FT), since the initial availability of the machines are considered (see definitions of both heuristics).

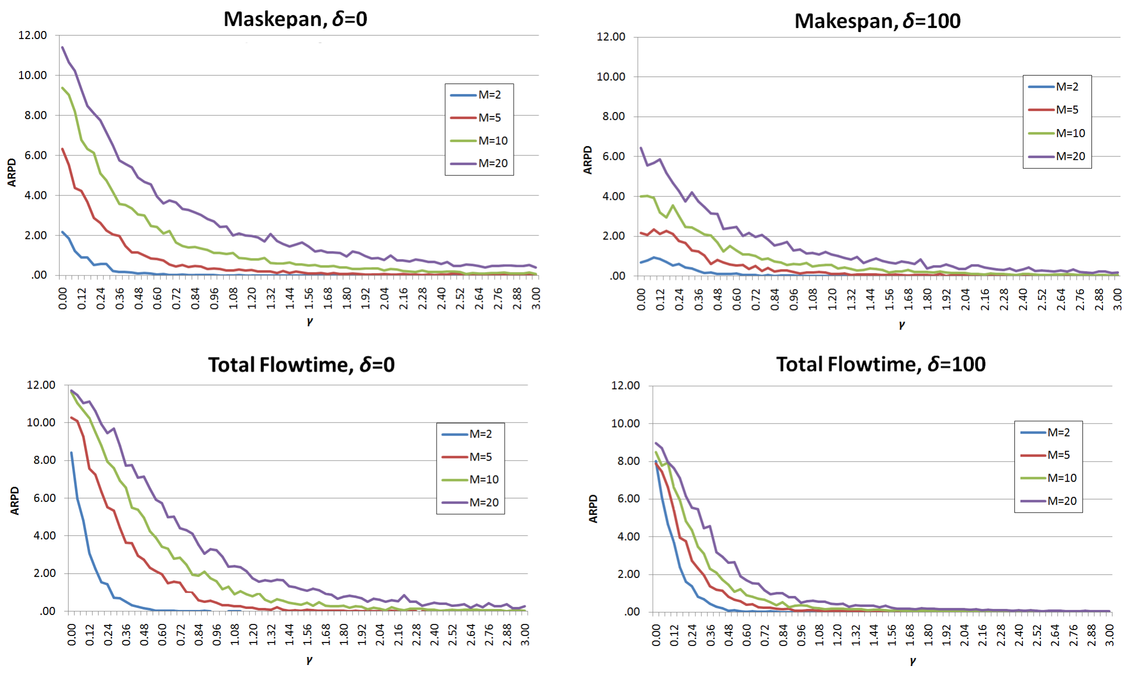

Figure 5: *ARPD* of SM_EB(i) for different values of m

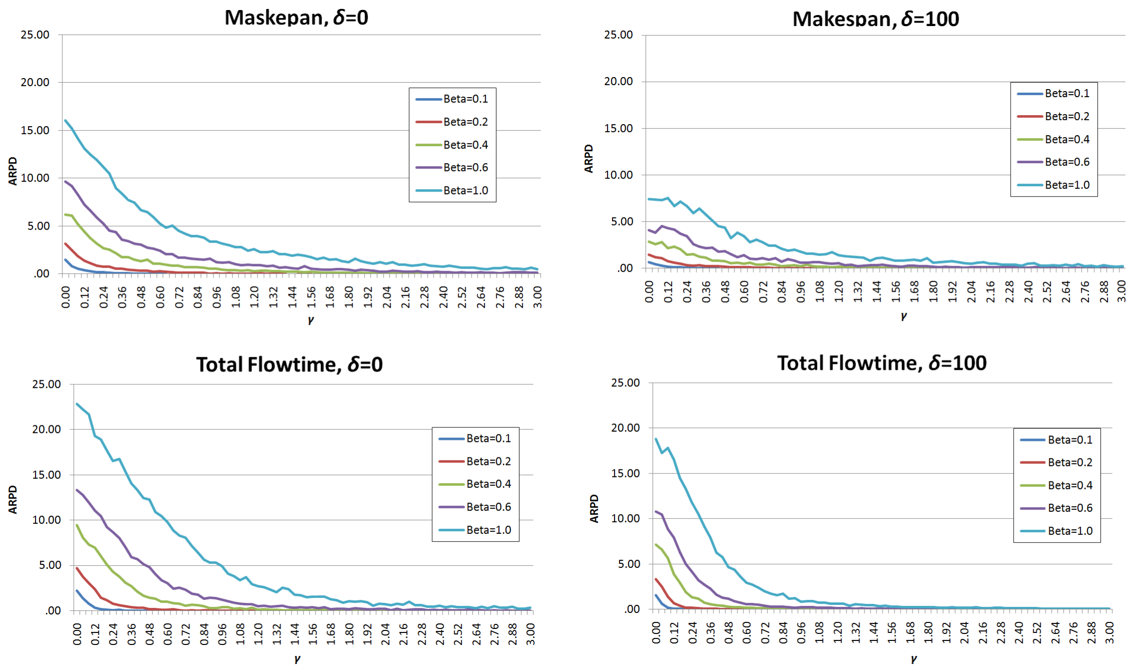

Figure 6: $ARPD$ of SM_EB(i) for different values of $beta$

Regarding β, its influence over the $ARPD$ is shown in Figure 6 for the heuristics SM_EB(i). The curves also present a sharp decrease in $ARPD$ when β decreases. In fact, from $\gamma \gtrsim 0.08$, the $ARPD$ for the curve $\beta = 0.10$ is always less than 1, regardless the objective or the value of δ.

4.5 Boundary Lines between the PFSP and the SMSP

In previous sections, we have proved the relationship between both scheduling problems and shown that the $ARPDs$ of several heuristics (designed for the original PFSP and for several SMSPs) tend to be similar for high values of γ, δ and n, and for low values of m and β. In this section, we analyse the conditions that have to be approximately fulfilled so that the reduced SMSP is (roughly) equivalent to the original PFSP. First, let us assume that both problems are similar if the differences in the $ARPDs$ of the heuristics to solve both problems (i.e. $PF\_B(i)$ and $SM\_EB(i)$) are lesser than 0.5%. The experiments in this section are carried out for an exhaustive set of 608,000 instances (which contain some more values of β as compared to previous testbed):

- $n \in \{20, 50, 100, 200\}$.
- $m \in \{2, 5, 10, 20\}$.
- $\epsilon = 50$.
- $\beta \in \{0.04, 0.08, \ldots, 0.96, 1.00\}$.

- $\gamma \in \{0.00, 0.04, 0.08, \ldots, 2.96, 3.00\}$.
- $\delta \in \{0, 100\}$.

In this set, there are 40 instances with different values of n for each combination of m, β, γ and δ. Let us denote by $ARPD'_{m,\beta,\gamma,\delta}$ the average RPD of these 40 instances for each value of m, β, γ and δ as well as by $\gamma^*_{m,\beta,\delta}$, the first value of γ for which $ARPD'_{m,\beta,\gamma,\delta} < 0.5$ for the instances with parameters m, β and γ. Values of $\gamma^*_{m,\beta,\delta}$ are graphically shown in Figure 7 and 8 for makespan and flowtime minimisation respectively. On the left sides of both figures, values for $\delta = 0$ are shown while on the right sides values for $\delta = 100$ are shown. Additionally, for each value of m, a linear trend line is represented. Thereby, these lines represent approximately the boundary lines of both decision problems. On the one hand, given a number of machines m, instances with values of γ and β over the line represent the region where both scheduling problems are similar (i.e. the difference of $ARPD$ between the heuristics PF_B(i) and SM_EB(i) is lower than 0.5%). On the other hand, values of γ and β under the line represent instances which should be solved using heuristics specifically designed for the PFSP (i.e. the difference of $ARPD$ between both heuristics is higher than 0.5%). The R^2 of each trend line is mostly close to 0.99. By means of these trend lines, regions with relative similar $ARPD$ between heuristics to solve the PFSP and the reduced SMSP are shown in Table 4 and 5, for makespan and total flowtime minimisation respectively.

Note that these boundary lines are obviously exact over the proposed set of instances but they are an approximation for other benchmarks or for processing times following different distributions. Thereby, they can be useful for the decision makers to give an idea of solving their manufacturing layouts, since variables γ and β can be easily approximated by a sample of the processing times of the shop. Let $\bar{\mu}_1$ and $\bar{\mu}_2$ be the sample means of the processing times on the saturated machine and non-saturated machine respectively. Additionally, let $\bar{\sigma}^2_s$ be the unbiased sample variance. Then, using the definition of the mean and the variance for the uniform distribution, the following expressions approximate the parameters used in this study:

- $\bar{\mu}_1 \simeq \epsilon \longrightarrow \epsilon \simeq \bar{\mu}_1$
- $\bar{\mu}_2 \simeq \epsilon \cdot (1+\gamma) \longrightarrow \gamma \simeq \frac{\bar{\mu}_2}{\epsilon} - 1$
- $\bar{\sigma}^2_s \simeq \frac{(2\cdot\epsilon\cdot\beta+1)^2-1}{12} \longrightarrow \beta \simeq \frac{\sqrt{\bar{\sigma}^2_s\cdot 12+1}-1}{2\cdot\epsilon}$

By means of these expressions, given an instance, the decision maker can compute the corresponding values of ϵ, γ, and β, and use them to position the instance above or below the boundary line. This will be the decision maker a guess on whether this instance should or should not be solved using specific PFSP procedures.

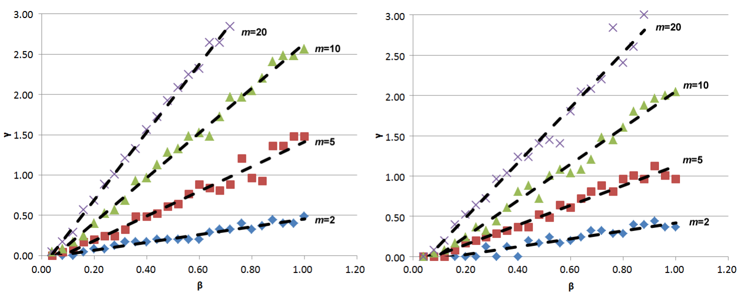

Figure 7: Boundary lines between the PFSP and the SMSP for makespan minimisation. On the left, no initial availability is considered and on the right an initial $\delta = 100$ is taken into account.

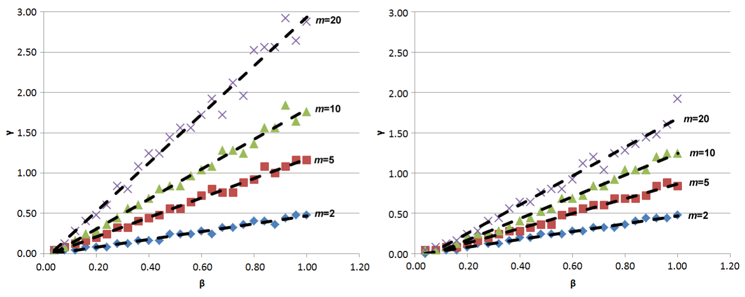

Figure 8: Boundary lines between the PFSP and the SMSP for flowtime minimisation. On the left, no initial availability is considered and on the right an initial $\delta = 100$ is taken into account.

Table 4: Regions where solving the PFSP is very similar to the SMSP for makespan minimisation

m	$\delta = 0$	$\delta = 100$
2	$\gamma \geq 0.48 \cdot \beta - 3.08 \cdot 10^{-2}$, $(R^2 = 0.962)$	$\gamma \geq 0.49 \cdot \beta - 7.12 \cdot 10^{-2}$, $(R^2 = 0.890)$
5	$\gamma \geq 1.53 \cdot \beta - 12.12 \cdot 10^{-2}$, $(R^2 = 0.960)$	$\gamma \geq 1.22 \cdot \beta - 9.88 \cdot 10^{-2}$, $(R^2 = 0.972)$
10	$\gamma \geq 2.79 \cdot \beta - 14.64 \cdot 10^{-2}$, $(R^2 = 0.994)$	$\gamma \geq 2.26 \cdot \beta - 20.36 \cdot 10^{-2}$, $(R^2 = 0.984)$
20	$\gamma \geq 4.22 \cdot \beta - 15.58 \cdot 10^{-2}$, $(R^2 = 0.997)$	$\gamma \geq 3.43 \cdot \beta - 20.26 \cdot 10^{-2}$, $(R^2 = 0.976)$

Table 5: Regions where solving the PFSP is very similar to the SMSP for flowtime minimisation

m	$\delta = 0$	$\delta = 100$
2	$\gamma \geq 0.48 \cdot \beta - 1.48 \cdot 10^{-2}$, $(R^2 = 0.974)$	$\gamma \geq 0.49 \cdot \beta - 1.56 \cdot 10^{-2}$, $(R^2 = 0.989)$
5	$\gamma \geq 1.21 \cdot \beta - 3.40 \cdot 10^{-2}$, $(R^2 = 0.991)$	$\gamma \geq 0.92 \cdot \beta - 5.08 \cdot 10^{-2}$, $(R^2 = 0.987)$
10	$\gamma \geq 1.86 \cdot \beta - 6.56 \cdot 10^{-2}$, $(R^2 = 0.989)$	$\gamma \geq 1.35 \cdot \beta - 10.12 \cdot 10^{-2}$, $(R^2 = 0.989)$
20	$\gamma \geq 3.01 \cdot \beta - 7.92 \cdot 10^{-2}$, $(R^2 = 0.984)$	$\gamma \geq 1.79 \cdot \beta - 10.76 \cdot 10^{-2}$, $(R^2 = 0.982)$

Table 6: *ARPD* of the heuristics in structured problems

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Avg.
SM_R(MK)	5.54	2.82	2.25	1.91	1.60	1.42	1.26	1.26	1.15	1.09	0.98	1.94
SM_EB(MK)	4.93	2.15	1.50	1.19	0.89	0.72	0.64	0.56	0.48	0.41	0.37	1.26
PF_B(MK)	0.01	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.01
PF_W(MK)	15.36	7.66	5.47	4.58	3.74	3.33	2.88	2.76	2.48	2.31	2.13	4.79
M_B(MK)	4.09	2.16	1.49	1.34	0.96	0.95	0.78	0.79	0.67	0.62	0.58	1.31
SM_B(FT)	6.11	3.51	2.49	2.16	1.77	1.54	1.40	1.31	1.13	1.09	0.99	2.14
SM_EB(FT)	5.37	2.81	1.80	1.46	1.07	0.87	0.69	0.66	0.51	0.44	0.40	1.46
SM_W(FT)	9.84	6.06	5.01	4.44	3.94	3.64	3.35	3.26	3.06	2.97	2.69	4.39
PF_B(FT)	0.02	0.01	0.03	0.03	0.04	0.05	0.07	0.05	0.06	0.08	0.07	0.05
PF_W(FT)	16.43	9.44	7.45	6.48	5.77	5.20	4.75	4.52	4.21	4.03	3.66	6.54
M_B(FT)	4.46	2.59	1.92	1.79	1.40	1.28	1.13	1.10	1.00	0.91	0.91	1.68

4.6 Structured Problems. Set of instances of Watson et al. (2002)

A number of conclusions similar to those in previous sections are found when we analyse the heuristics over the set of instances of Watson et al. (2002), which uses a parameter α to establish the correlation of the processing times. Results in terms of *ARPD* for each value of α are shown in Table 6. When α increases, heuristics designed for the SMSP are closer to the heuristics implemented for the original PFSP, indicating that both problems are similar. Thereby, for $\alpha = 0.0$ the difference between SM_EB(MK) and PF_B(MK) is 4.92 while for $\alpha = 1.0$ it is 0.36. Similar results are obtained for SM_EB(FT) and PF_B(FT) as the difference is 5.35 for $\alpha = 0.0$ and 0.33 for $\alpha = 1.0$ and also with SM_W(FT) and PF_W(FT). These results also confirm the conclusion found by Watson et al. (2002) that the problems tend to be very easy with the increase of α for the $Fm|prmu|C_{max}$ problem as shown by the high decrease in the difference of *ARPD* between PF_B(MK) and PF_W(MK) (it goes from 15.35 at the beginning to 2.12 at the end). In fact, for high values of α, the small distances between both heuristics is probably an effect of the similarity between the $Fm|prmu|C_{max}$ problem and the $1||C_{max}$ problem, where each solution is optimal. Thereby, from $\alpha \gtrsim 0.4$ for makespan and $\alpha \gtrsim 0.5$ for total flowtime, the *ARPD* found by the heuristics designed to solve the equivalent SMSP (SM_EB(MK) and SM_EB(FT)) are lower than 1.0. Graphically, *ARPD* values as a function of α are shown in Figure 9 for makespan and in Figure 10 for total flowtime (for clarity, M_B(i) is not shown in the figures).

5 Conclusions

In this paper, several properties and dominance rules have been presented to analyse the relation between the PFSP and the SMSP for makespan and total flowtime minimisation depending on the processing times of the jobs. Additionally, in order to empirically compare the problems, 11 algorithms (5 for makespan

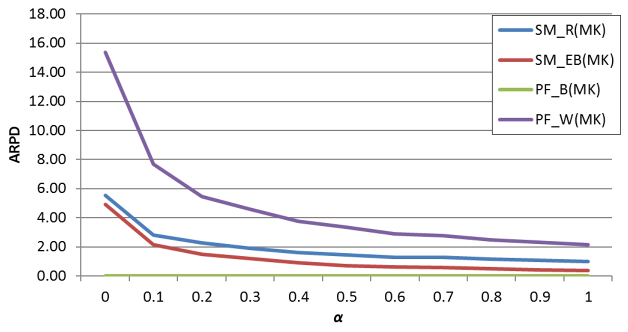

Figure 9: *ARPD* of heuristics for makespan minimisation using the set of instances of Watson et al. (2002).

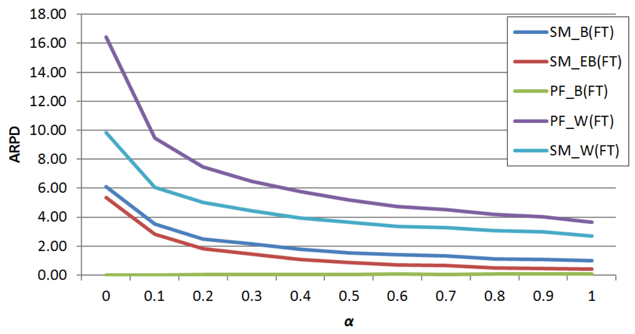

Figure 10: *ARPD* of heuristics for total flowtime minimisation using the set of instances of Watson et al. (2002).

and 6 for total flowtime) are tested on an extensive testbed with more than 600,000 instances designed for the PFSP with machine correlated processing times. Four algorithms have been designed to solve the instances of the PFSP. Five of them reduce each PFSP instance to an equivalent SMSP considering only the saturated machine, whereas the other 2 algorithms solve a reduced PFSP without considering the machines before the most saturated one. Results show that the algorithms designed for the PFSP and for the SMSP tend to be very similar for several values of the parameters of the testbed.

The goal of this paper is to prove the intuition that, when in a PFSP there is a machine with larger processing times, then the problem should be similar to the equivalent SMSP considering only this machine. Thereby, this paper intends to explore the theoretical and empirical boundaries between these two problems. On the one hand, theoretical results shown in the paper prove that both problems are equivalent under several conditions. Although these conditions are hardly present in a real manufacturing environment (particularly in shops with several machines), they are sufficient but not necessary conditions and they only give an idea of the relationship between both problems. On the other hand, the empirical comparison carried out in this paper stresses the relationship between both problems. It has been shown that increasing γ (which is related to the dispersion of the processing times) and n, and decreasing β (which is related to the predominance of the most loaded machine) and m makes the PFSP to be more similar to a SMSP. In fact, for low values of β and/or m, procedures for the equivalent SMSP are able to find similar or even better solutions than the heuristics to solve the original PFSP. In order to empirically establish the frontier between both problems, an extensive set of instances with 608,000 instances has been generated. Thus, we have obtained several boundary lines depending on the number of machines in the shop. For a configuration in the shop (number of machines, initial machine availabilities, objective to be solved, length or variation of the processing times on the machines), these lines show the values of γ causing the difference of $ARPD$ between the heuristics of both problems to be less than 0.5% on the analysed set of instances.

The relation between both scheduling problems shown in this paper highlights the importance of the pre-processing of the processing times of the problems, as well as the importance of the right choice of the scheduling problem to be solved, which does not necessary match the original machine environment of the shop. Additionally, it explains the behaviour found in the papers of Watson et al. (2002) and of Perez-Gonzalez and Framinan (2009) where the $Fm|prmu|C_{max}$ problem has been found to be easily solvable for structured instances and for machines with initial availabilities, respectively.

Regarding future research lines, although the presented paper represents an advance in the study of the relationship between the PFSP and the SMSP, the boundary lines between both problems are not yet completely defined. Further enhancements may focus on the following issues:

- The variance of the processing times on the saturated machine probably plays an important role in the relationship between both problems.
- The present study uses a uniform distribution for the processing times. Further analyses can use of different distributions, extending the boundary lines between the problems.
- The presented analysis may probably be extended to other scheduling layouts.
- The PFSP has been compared with the SMSP of the saturated machine. Further analysis may compare the PFSP with a SMSP combining the processing times of different machines.

References

Carlier, J. (1978). Ordonnancements a contraintes disjonctives. *RAIRO Recherche Operationnelle*, 12(4):333–350.

Cepek, O., Okada, M., and Vlach, M. (2002). Nonpreemptive flowshop scheduling with machine dominance. *European Journal of Operational Research*, 139(2):245–261.

Cheng, M., Sun, S., and Yu, Y. (2007). A note on flow shop scheduling problems with a learning effect on no-idle dominant machines. *Applied Mathematics and Computation*, 184(2):945–949.

Demirkol, E., Mehta, S., and Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141.

Dong, X., Chen, P., Huang, H., and Nowak, M. (2013). A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time. *Computers and Operations Research*, 40(2):627–632.

Dudek, R. and Teuton, O. (1964). Development of m stage decision rule for scheduling n jobs through m machines. *Operations Research*, 12:471.

Easwaran, G., Parten, L., Moras, R., and Uhlig, P. (2010). Makespan minimization in machine dominated flowshop. *Applied Mathematics and Computation*, 217(1):110–116.

Fernandez-Viagas, V. and Framinan, J. (2015a). A new set of high-performing heuristics to minimise flowtime in permutation flowshops. *Computers and Operations Research*, 53:68–80.

Fernandez-Viagas, V. and Framinan, J. (2015b). NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. *Computers and Operations Research*, 60:27–36.

Fernandez-Viagas, V. and Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers and Operations Research*, 45(0):60 – 67.

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.

Framinan, J. and Leisten, R. (2006). A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness. *International Journal of Production Economics*, 99(1-2):28–40.

Framinan, J., Leisten, R., and Ruiz-Usano, R. (2002). Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European Journal of Operational Research*, 141(3):559–569.

Framinan, J., Leisten, R., and Ruiz-Usano, R. (2005). Comparison of heuristics for flowtime minimisation in permutation flowshops. *Computers and Operations Research*, 32(5):1237–1254.

Gajpal, Y. and Rajendran, C. (2006). An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. *International Journal of Production Economics*, 101(2):259–272.

Garey, M., Johnson, D., and Sethi, R. (1976). Complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129.

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.

Heller, J. (1960). Some numerical experiments for an m x j flow shop and its decision-theoretical aspects. *Operations Research*, 8(2):178–184.

Ho, J. and Gupta, J. (1995). Flowshop scheduling with dominant machines. *Computers and Operations Research*, 22(2):237–246.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

Johnson, S. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68.

Leisten, R. and Rajendran, C. (2014). Variability of completion time differences in permutation flow shop scheduling. *Computers and Operations Research*, 54:155–167.

M'Hallah, R. (2014). An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. *International Journal of Production Research*, 52(13):3802–3819.

Monma, C. L. and Rinnooy Kan, A. H. G. (1983). A concise survey of efficiently solvable special cases of the permutation flow-shop problem. *RAIRO - Operations Research - Recherche Opérationnelle*, 17(2):105–119.

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

Pan, Q.-K. and Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers and Operations Research*, 40(1):117–128.

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. (2012). *Scheduling: Theory, algorithms, and systems: Fourth edition.* Springer US, New York.

Rad, S. F., Ruiz, R., and Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *OMEGA, The International Journal of Management Science*, 37(2):331–345.

Reeves, C. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 22(1):5–13.

Reza Hejazi, S. and Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research*, 43(14):2895–2929.

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.

Schaller, J. and Valente, J. (2013). A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness. *International Journal of Production Research*, 51(3):772–779.

Sun, Y., Zhang, C., Gao, L., and Wang, X. (2011). Multi-objective optimization algorithms for flow shop scheduling problem: A review and prospects. *International Journal of Advanced Manufacturing*

Technology, 55(5-8):723–739.

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

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

Vallada, E., Ruiz, R., and Framinan, J. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240:666–677.

Vallada, E., Ruiz, R., and Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers and Operations Research*, 35(4):1350–1373.

Wang, J.-B., Shan, F., Jiang, B., and Wang, L.-Y. (2006). Permutation flow shop scheduling with dominant machines to minimize discounted total weighted completion time. *Applied Mathematics and Computation*, 182(1):947–954. cited By 9.

Watson, J., Barbulescu, L., Whitley, L., and Howe, A. (2002). Contrasting structured and random permutation flow-shop scheduling problems: Search-space topology and algorithm performance. *INFORMS Journal on Computing*, 14(2):98–123.