mathematics

Article

The Permutation Flow Shop Scheduling Problem with Human Resources: MILP Models, Decoding Procedures, NEH-Based Heuristics, and an Iterated Greedy Algorithm

Victor Fernandez-Viagas [1,2,*], Luis Sanchez-Mediano [1], Alvaro Angulo-Cortes [1], David Gomez-Medina [1,2] and Jose Manuel Molina-Pariente [1,2]

1 Department of Industrial Organization and Business Management I, School of Engineering, University of Seville, 41091 Seville, Spain
2 Grupo de Informática de la Salud Computacional, Hospital Universitario Virgen del Rocío, 41013 Seville, Spain
* Correspondence: vfernandezviagas@us.es

Abstract: In this paper, we address the permutation flow shop scheduling problem with sequence-dependent and non-anticipatory setup times. These setups are performed or supervised by multiple servers, which are renewable secondary resources (typically human resources). Despite the real applications of this kind of human supervision and the growing attention paid in the scheduling literature, we are not aware of any previous study on the problem under consideration. To cover this gap, we start theoretically addressing the problem by: proposing three mixed-integer linear programming models to find optimal solutions in the problem; and proposing different decoding procedures to code solutions in approximated procedures. After that, the best decoding procedure is used to propose a new mechanism that generates 896 different dispatching rules, combining different measures, indicators, and sorting criteria. All these dispatching rules are embedded in the traditional NEH algorithm. Finally, an iterated greedy algorithm is proposed to find near-optimal solutions. By doing so, we provide academics and practitioners with efficient methods that can be used to obtain exact solutions of the problem; applied to quickly schedule jobs and react under changes; used for initialisation or embedded in more advanced algorithms; and/or easily updated and implemented in real manufacturing scenarios.

Keywords: scheduling; flow shop; MILP; decoding procedure; makespan; flow shop; human resources; multiple servers; sequence-dependent setups; iterated greedy

MSC: 90B35; 68M20; 90C59

Citation: Fernandez-Viagas, V.; Sanchez-Mediano, L.; Angulo-Cortes, A.; Gomez-Medina, D; Molina-Pariente, J.M. The Permutation Flow Shop Scheduling Problem with Human Resources: MILP Models, Decoding Procedures, NEH-Based Heuristics, and an Iterated Greedy Algorithm. *Mathematics* **2022**, *10*, 3446. https://doi.org/10.3390/math10193446

Academic Editor: Hsien-Chung Wu

Received: 27 July 2022
Accepted: 19 September 2022
Published: 22 September 2022

1. Introduction

Minimising the makepan in the Permutation Flow shop Scheduling Problem (PFSP) is one of the most addressed problems in the Operations Research literature [1]. In the PFSP, there are n jobs, each one with m operations, and a set of m machines. Each machine is exclusively responsible for processing an operation of each job and, therefore, each job must be processed on all machines following the same route. The goal of this scheduling problem is to find the best sequence (the same in all machines) to process the jobs according to a certain objective function, with the makespan being the most common one in the literature. In this problem, machines are typically classified as primary resources that remain busy throughout the processing of each job. In addition, in several situations, the processing of jobs may also require the use of other resources (such as, e.g., raw materials, human resources, or setup tools), denoted as 'secondary' ([2]). Among these types of secondary resources, the use of servers is very common in the real manufacturing industry [3]: examples can be found in flexible manufacturing systems [4]; in scheduling problems with versatile machines and assembly components [5]; in computer-controlled material handling systems [6]; in biomass truck scheduling problems in the context of supply chain optimisation [7]; and in container terminals [8]. Servers are secondary resources that are in charge of carrying out setups between jobs or families of jobs. In this sense, these servers could represent a robot (see, e.g., [4]), a human (see, e.g., [9]), or an automatic guided vehicle ([10]), among others. In the literature, this type of resource has also been denoted as a setup operator (see, e.g., [11,12]). In addition to the previous real implications of the problem, this scheduling problem is especially appropriate for the U-shaped manufacturing layout (see [13] for a definition of this production line layout), where machines are distributed in a flow-shop layout with a U shape, and human resources are located in the centre in charge of carrying out the changeovers.

Despite the practical applications of servers in real scenarios and the recent interest in the literature to these kind of scheduling problems with secondary resources responsible for setup times (see, e.g., [3,9,14–16]), literature addressing this topic is still very scarce and there are many open research questions. Most papers address the problem with a single server (see, e.g., [6,17]) under very simple layouts, mostly identical parallel machines (see, e.g., [18–20]). The case with multiple human resources has been tackled only in unrelated parallel machines, hybrid flow shop, open shop, and no-wait flow shop. Although these papers propose relevant algorithms for solving such scheduling problems, there is still a need to further improve the knowledge of multiple servers, especially solving and analysing its influence in the permutation flow shop scheduling problem, the most relevant and addressed layout in the literature. To cover this gap, we address the Permutation Flow shop Scheduling problem with Multiple Servers, denoted as PFSMS in the following. Regarding the setup times which have to carry out these servers, according to [21], they may be classified first as non-anticipatory and anticipatory: the former (also denoted as inseparable or attached) must be executed after the job arrival; the latter (also denoted as separable or detached) can be performed at the machine at any time after the completion of the previous job (i.e., they may be performed before the job arrival). Second, setup times can be classified as sequence-dependent or sequence-independent if this amount of time depends or not on the previous job executed on the same machine, respectively. Following the common approach used in the literature, in this paper, we address the PFSMS with non-anticipatory and sequence-dependent setup times. This scheduling problem, with the objective of minimising the makespan, is denoted by $Fm, Sr|prmu|C_{max}$, according to [18,22]. This problem is clearly NP-hard, since the same problem without considering the servers is already NP-hard [23].

To deal with this PFSMS for the first time, the contributions of this paper can be stated as follows:

- A new PFSP problem is defined with multiple servers or human resources in charge of carrying out setups.
- We identify efficient formulations to solve the proposed problem, by developing three different Mixed Integer Linear Programming (MILP) models.
- Another contribution is the development of decoding procedures. These procedures are important to obtain feasible schedules and to reduce the solution space of the problem. Both issues are essential to guarantee the efficiency of approximated algorithms developed for the problem.
- A new procedure to generate static dispatching rules is proposed. Using this procedure, we propose and compare 896 different dispatching rules.
- In addition, we propose different NEH-based constructive heuristics to efficiently solve the problem. Using the traditional NEH heuristic ([24]), we analyse the influence and efficiency of its first phase, embedding all previous dispatching rules. In this regard, note that heuristics have been traditionally proposed in the literature either to obtain high-quality solutions in short times (required when decisions should be made almost instantaneously or with high computational requirements) or to provide good initial solutions for more advanced algorithms (as metaheuristics). Recently, they are also relevant for quickly reacting to the changing environments of industry 4.0 ([25]).

- Finally, starting with the best constructive heuristic, we develop an iterated greedy metaheuristic to find near-optimal solutions in large-sized instances.

Note that with these contributions, we try to introduce future researchers and practitioners with the problem under study, providing them with tools that can be directly applied, combined or adapted to other related scheduling problems. It is also noteworthy to mention that, as most proposals in the literature, we propose deterministic approaches to solve the problem under consideration. Despite the inherent uncertainty in the supply chain ([26–28]), in many scheduling situations, deterministic approaches are very robust to stochasticity and uncertainty (see in this regard [25,29]).

The rest of the paper is organised as follows: the problem is described in Section 2. In this section, we also review the literature related to the problem under study. In Section 3, we propose several new formulations for the problem using MILP models. The decoding procedures are explained in Section 4. Regarding approximate algorithms, the proposed dispatching rules and constructive heuristics are detailed in Section 5, while the iterated greedy algorithm is explained in Section 6. The computational results of all previous methods are shown in Section 7. Finally, the conclusions and future research lines are discussed in Section 8.

2. Problem Description and Background

In the PFSP, there is a set $\mathcal{N}$ of n jobs that must be processed on a set $\mathcal{M}$ of m machines, following the same route of machines for each job. Each machine is always available and has to process jobs (one by one) following a certain sequence, $\Pi = (\pi_1, \dots, \pi_n)$, which is identical for all machines (permutation constraint). Let O_{ij} denote the operation of job j processed on machine i. Whenever it does not cause confusion, let $O_{i[k]}$ be the operation corresponding to the job in position k on machine i (i.e., $O_{i\pi_k}$). Each operation O_{ij} has a processing time p_{ij}. In addition, each job j requires a setup time s_{ilj}, when it is processed after job l on machine i. This setup is both non-anticipatory and sequence-dependent, and has to be carried out on that machine i by a worker w. Let $\mathcal{R}$ be the set of r identical workers that can perform all setups. Denoting by C_{ij} the completion time of job j on machine i, the goal of the problem is to find the best schedule that minimises the maximum completion time, C_{max}, where C_{max} is equal to $C_{m\pi_n}$.

In Figure 1, we show an example of $Fm, Sr|prmu, s_{ilj}|C_{max}$ with four jobs, four machines, and two workers. We can observe, for example, that the setup operation $O_{3[1]}$ cannot be performed before, as all the workers are unavailable. Similarly, some idle time is also forced on machines 3 and 4 by the setup of operations $O_{3[2]}$ and $O_{4[2]}$, which have to wait until a worker is available.

As mentioned in the previous section, due to its importance, the permutation flow shop scheduling problem is one of the most active problems in the operation literature with hundreds of contributions in the last years, addressing different variants and constraints in the classical problems. Recent examples solving the permutation flow shop can be found in [30–36], and addressing different setups configurations in [37–44]. For comprehensive reviews of the problem under different constraints, we refer the interested reader to [1,45–49]. Despite the extensive flow-shop-based literature, we are not aware of any previous reference to the problem under study (PFSMS) so far. Therefore, we focus this review of the literature on scheduling problems that deal with single or multiple servers. Regarding the single server literature, most focus on solving the parallel machine scheduling problem and applying it . In this regard, this problem has been solved for two machines, for example [3,4,50] considering sequence-independent setup times and using objective functions based on the makespan or total idle time. The problem with m machines has been solved by [6] sequence-dependent setup times and makespan minimisation and by [17] for sequence-dependent setup times to minimise total weighted earliness and tardiness. The complexity of this kind of problem is addressed by [10]. The problem with a single server has also been solved by [51] for the parallel dedicated machine scheduling problem with sequence-dependent setup times and makespan minimisation, while [20] have proposed a MILP model and two approximate algorithms to solve the unrelated parallel scheduling problem with sequence-dependent setup times and machine eligibility restrictions. Regarding other layouts in the literature with a single server, the problem has also been solved for the flow shop scheduling problem with two machines and minimisation of the makespan by [52]. The authors have addressed the problem of sequence-independent and anticipatory setup times. The complexity of this problem is addressed by [53], considering unit processing times, and, more generally, by [54]. Using the same configuration of setup times and traditional processing times, [55] have solved the no-wait variant of the problem to minimise the total completion time. Anticipatory and non-anticipatory setup times are addressed by [56], minimising the makespan, and also considering dismounting times. The problem with m machines and a single server has been solved by [5] for total completion times' minimisation, but in this case, the server is responsible for both processing and setup operations. The authors have addressed the problem with non-anticipatory and sequence-independent setup times.

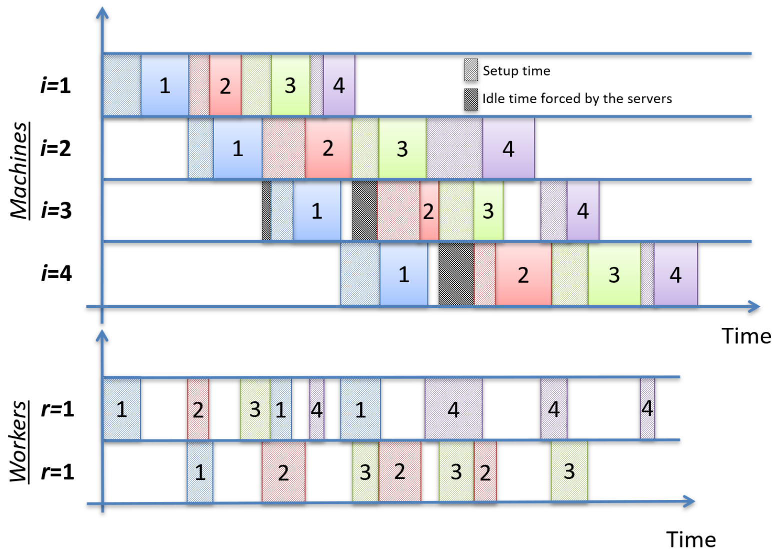

Figure 1. Example of the $Fm, Sr|prmu, s_{ij}|C_{max}$ problem.

Regarding the related scheduling problem with multiple servers, [18] have addressed the identical parallel machine scheduling problem with sequence-independent setup times. For the unrelated parallel machine layout, [15] have proposed a Grasp algorithm to solve the problem with sequence-dependent setup times for makespan minimisation, while [16] have proposed an iterated greedy-based algorithm to solve a bi-objective variant of the problem. Both researchers have addressed the case where a setup requires more than one server at the same time. Based on the ceramic tile manufacturing sector, [19] have solved a related unrelated parallel machine layout, where the setup times depend on the assignment of resources. The hybrid flow shop scheduling problem with sequence-independent setup times and multiple servers has recently been addressed by [9,14]. The former has proposed a backtracking search optimisation algorithm to solve the anticipatory variant, while the latter have proposed several constructive and composite heuristics to solve both the anticipatory and non-anticipatory cases. An example in the open shop layout with two machines can be found in [57] for non-anticipatory and sequence-independent setup times. Regarding the flow shop layout with multiple servers, the problem has been solved only by [58,59]. However, they addressed the no-wait variant using anticipatory and sequence-dependent setup times by proposing a genetic algorithm. A summary of the literature review is presented in Table 1.

Table 1. Summary of the literature review.

Paper	**#Machines**		**Layout**		**#Workers**		**Setup**				**Other**	**Objective**				**Approach**			
	$m = 2$	$m \geq 2$	**FS**	**O**	$r = 1$	$r \geq 1$	**SD**	**SI**	**A**	**NA**	**Constraint**	C_{max}	$\sum C_j$	**OM**	**MO**	**MILP**	**H**	**MH**	**O**
[3]	X			P	X			X	-	-		X				X			
[4]	X			P	X			X	-	-				X			X		
[5]		X	X		X			X		X	SPS		X			X		X	
[6]		X		P	X		X		-	-		X				X		X	
[9]		X		HF		X		X	X			X				X		X	
[14]		X		HF		X		X	X	X		X					X		
[15]		X		R		X	X		-	-		X				X	X	X	
[16]		X		R		X	X		-	-					X			X	
[50]		X		P	X			X	-	-		X		X			X		
[17]		X		P	X			X	-	-				X				X	
[51]		X		DP	X		X		-	-		X				X		X	
[20]		X		R	X		X		-	-	E	X				X		X	
[52]	X		X		X			X	X		CPT	X					X	X	
[55]	X		X		X			X	X				X				X		X
[56]	X		X		X			X	X	X	DT	X					X		
[57]	X			O		X		X		X		X					X		
[19]		X		R		X	X		-	-	STAR				X	X	X		
[58]		X	X			X	X		X		NW	X						X	
[59]		X	X			X	X		X		NW	X						X	
This		X	X			X	X			X		X				X	X	X	

Notation: FS, Flow shop; P, Identical Parallel Machines; DP, Dedicated Machines; R, Unrelated Parallel Machines; O, Other; NW, no-wait; STAR, setup times depending on the number of assigned resources; SPS, servers must process the jobs and carry out the setup; DT, Dismounting Times; CPT, Constant Processing Times; E, Eligibility; SD, Sequence dependent setup times; SI, sequence independent setup times; A, Anticipatory setup times; NA, Non-anticipatory setup times; OM, Other Mono-objective; MO, Multiobjective; H, Heuristic; MH, Metaheuristic.

3. MILP Models

In this section, we elaborate three different MILP models to exactly solve the problem under consideration. The parameters and variables used by the models are presented in Section 3.1. Then, a detailed description of each model is shown in Sections 3.2–3.4 for Models 1, 2, and 3, respectively.

3.1. Notation: Variables and Common Parameters

The three proposed MILP models use the following common indices and parameters:

- i ($i \in \mathcal{M}$): machine index.
- j ($j \in \mathcal{N}$): job index.
- k ($k = 1, \ldots, n$): position index.
- w ($w \in \mathcal{R}$): worker index.
- $s_{ijj'}$ ($i \in \mathcal{M}$,j,$j' \in \mathcal{N}$): setup time of job j' on machine i when it is processed immediately after job j. If the job j' is the first job of the sequence, its setup time is defined by $s_{i,0,j'}$.
- p_{ij} ($i \in \mathcal{M}$,$j \in \mathcal{N}$): processing time of job j on machine i.

The proposed models use some of the following variables:

- X_{ijw} ($i \in \mathcal{M}$,$j \in \mathcal{N}$, $w \in \mathcal{R}$): 1 if the setup associated with the operation O_{ij} is carried out by the worker w.
- $\alpha_{jkj'}$ ($j, j', k \in \mathcal{N}$): 1 if job j is assigned to position k and is processed before job $j^{'}$, 0 otherwise. Additionally, let $\alpha_{0,0,j'} = 1$ if job $j^{'}$ is the first job in the sequence (0 otherwise).
- $\beta_{jj'}$ ($j, j^{'} \in \{0\} \cup \mathcal{N}$): 1 if job j is processed before job $j^{'}$, 0 otherwise.
- Z_{jk} ($j \in \mathcal{N}$, $k \in \mathcal{N}$): 1 if job j is assigned to position k, 0 otherwise.
- γ_{ikw} ($i \in \mathcal{M}$,$k \in \mathcal{N}$, $w \in \mathcal{R}$): 1 is setup of the job in position k is performed by worker w on machine i.
- C_{ij} ($i \in \mathcal{M}$, $j \in \mathcal{N}$): Completion time of job j on machine i.
- C_{max}: Makespan.
- $\hat{C}_{ik}$ ($\in \mathcal{M}$, $k \in \mathcal{N}$): Completion time of the job in position k on machine i.
- B_{ij} ($i \in \mathcal{M}$, $j \in \mathcal{N}$): Starting time of job j on machine i.
- $\hat{B}_{ik}$ ($i \in \mathcal{M}$, $k \in \mathcal{N}$): Starting time of the job in position k on machine i.
- $\eta_{wiji'j'}$ ($w =\in \mathcal{R}$, $i, i^{'} \in \mathcal{M}$, $j, j^{'} \in \mathcal{N}$): 1 if worker w performs the setup of operation O_{ij} immediately before operation $O_{i'j'}$. Furthermore, let $\eta_{w00ij} = 1$ if operation O_{ij} is the first operation in the worker w.
- $\lambda_{iji'j'}$ ($i, i^{'} \in \mathcal{M}$, $j, j^{'} \in \mathcal{N}$): if the setup for the operations O_{ij} and $O_{i'j'}$ are performed on the same machine, then it takes 1 if the setup of $O_{i'j'}$ is carried out after the setup of O_{ij}, and 0 otherwise.
- $\hat{\lambda}_{iki'k'}$ ($i, i' \in \mathcal{M}$, $k, k' \in \mathcal{N}$): if the setup for the operations $O_{i[k]}$ and $O_{i'[k']}$ is performed on the same machine, then it takes 1 if the setup of $O_{i'[k']}$ is carried out after setup of $O_{i[k]}$, and 0 otherwise.

3.2. Model 1

Our first proposal, denoted Model 1, emerges from the Wilson model, introduced by [60] in the classical $Fm|prmu|C_{max}$, which belongs to the Wagner family. This model used Z_{jk} to define the sequence of jobs in the shop. Then, this auxiliary variable is used to calculate the starting time of the job in position k ($\hat{B}_{ik}$). In our proposal, we introduce the setup times and workers that are defined by the decision variables $\alpha_{jkj'}$, γ_{ikw}, and $\hat{\lambda}_{iki'k'}$. Using the previous parameters and variables, Model 1 can be formulated as follows:

$$\text{Min } \hat{B}_{mn} + \sum_{j \in \mathcal{N}} p_{m,j} Z_{j,n}$$

subject to

$$\sum_{k=1}^{n} Z_{jk} = 1 \qquad j \in \mathcal{N} \tag{1}$$

$$\sum_{j \in \mathcal{N}} Z_{jk} = 1 \qquad k \in \mathcal{N} \tag{2}$$

$$\sum_{j' \in \mathcal{N}-\{j\}} \alpha_{jkj'} = Z_{jk} \qquad j \in \mathcal{N}, k \in \mathcal{N} - \{n\} \tag{3}$$

$$\sum_{j' \in \mathcal{N}-\{j\}} \alpha_{j',k-1,j} = Z_{jk} \qquad j \in \mathcal{N}, k \in \mathcal{N} - \{1\} \tag{4}$$

$$\alpha_{0,0,j} = Z_{j,1} \qquad j \in \mathcal{N} \tag{5}$$

$$\hat{B}_{1,k} + \sum_{j \in \mathcal{N}} p_{1,j} Z_{j,k} + \sum_{j \in \mathcal{N}} \sum_{j' \in \mathcal{N}-\{j\}} (s_{1,j,j'} \cdot \alpha_{jkj'}) \leq \hat{B}_{1,k+1} \qquad k \in \mathcal{N} - \{n\} \tag{6}$$

$$\hat{B}_{i,1} + \sum_{j \in \mathcal{N}} p_{i,j} Z_{j,1} + \sum_{j' \in \mathcal{N}} (s_{i+1,0,j'} \cdot \alpha_{0,0,j'}) \leq \hat{B}_{i+1,1} \qquad i \in \mathcal{M} - \{m\} \tag{7}$$

$$\hat{B}_{1,1} = \sum_{j' \in \mathcal{N}} (s_{1,0,j'} \cdot \alpha_{0,0,j'}) \tag{8}$$

$$\hat{B}_{i,k} + \sum_{j \in \mathcal{N}} p_{ij} Z_{jk} + \sum_{j \in \mathcal{N}} \sum_{j' \in \mathcal{N}-\{j\}} (s_{i+1,j,j'} \cdot \alpha_{j,k-1,j'}) \leq \hat{B}_{i+1,k} \qquad i \in \mathcal{M} - \{m\}, k \in \mathcal{N} - \{1\} \tag{9}$$

$$\hat{B}_{i,k} + \sum_{j \in \mathcal{N}} p_{ij} Z_{jk} + \sum_{j \in \mathcal{N}} \sum_{j' \in \mathcal{N}-\{j\}} (s_{ijj'} \cdot \alpha_{jkj'}) \leq \hat{B}_{i,k+1} \qquad i \in \mathcal{M} - \{1\}, k \in \mathcal{N} - \{n\} \tag{10}$$

$$\sum_{w \in \mathcal{R}} \gamma_{ikw} = 1 \qquad i \in \mathcal{M}, k \in \mathcal{N} \tag{11}$$

$$\hat{B}_{i'k'} - \hat{B}_{ik} + V(3 - \gamma_{ikw} - \gamma_{i'k'w} - \hat{\lambda}_{iki'k'}) \geq \sum_{j \in \mathcal{N}} \sum_{l \in \mathcal{N}-\{j\}} (s_{i'jj'} \alpha_{j,k'-1,j'}) \;, w \in \mathcal{R}, i^{'} \in \mathcal{M}, i > i^{'}, k \in \mathcal{N}, k^{'} > k, \tag{12}$$

$$\hat{B}_{ik} - \hat{B}_{i'k'} + V(2 - \gamma_{ikw} - \gamma_{i'k'w} + \hat{\lambda}_{iki'k'}) \geq \sum_{j \in \mathcal{N}} \sum_{j' \in \mathcal{N}-\{j\}} (s_{ijj'} \alpha_{j,k-1,j'}) \;, w \in \mathcal{R}, i' \in \mathcal{M}, i > i^{'}, k \in \mathcal{N}, k^{'} > k, \tag{13}$$

$$\hat{B}_{i'k'} - \hat{B}_{ik} + V(2 - \gamma_{ikw} - \gamma_{i'k'w}) \geq \sum_{j \in \mathcal{N}} \sum_{l \in \mathcal{N}-\{j\}} (s_{i'jj'} \alpha_{j,k'-1,j'}) \qquad , w \in \mathcal{R}, i \in \mathcal{M}, i^{'} \geq i, k \in \mathcal{N}, k^{'} > k, \tag{14}$$

$$\hat{B}_{i'k} - \hat{B}_{ik} + V(2 - \gamma_{ikw} - \gamma_{i'kw}) \geq \sum_{j \in \mathcal{N}} \sum_{l \in \mathcal{N}-\{j\}} (s_{i'jj'} \alpha_{j,k'-1,j'}) \qquad , w \in \mathcal{R}, i \in \mathcal{M}, i^{'} > i, k \in \mathcal{N}, \tag{15}$$

The set of constraints (1) ensures that every job is assigned to a unique position, while the constraints (2) achieve that exactly one job is assigned to each position. The set (3) establishes that each job has a successor, whereas the predecessors are defined in constraints (4) and (5). The starting time of every job on the first machine is defined in the set (6). Analogously, the starting time of the first job on every machine is calculated using constraints (7) and (8). Constraints (9) establish that a job must start after its previous operation is completed. The set (10) ensures that a job starts once its previous job in the same machine has been processed. Constraints (11) ensure that every job is assigned to a worker. The set of constraints (12) and (13) achieves that the same worker does not process two setups at the same time. Finally, on the one hand, constraints (14) establish that an operation $O_{i[k]}$ has to start before an operation $O_{i'[k']}$ if $k' > k$ and $i' \geq i$. On the other hand, (15) enforces that operation $O_{i[k]}$ must start before $O_{i'[k]}$ if $i' > i$.

3.3. Model 2

Our second proposal is denoted by Model 2. This model belongs to the Manne family of models ([60]), also denoted as a disjunctive formulation. Some examples of this formulation in flow-shop layouts can be found in the SGST model proposed by [60], in the MILP model proposed by [9] or in the MIP1 model proposed by [5]. This family starts by defining the sequence of jobs by variable $\beta_{jj'}$. In our case, we incorporate variable $\lambda_{iji'j'}$ for the sequences in the workers. In doing so, Model 3 can be formulated as follows:

$$\text{Min } C_{max}$$

subject to

$$\sum_{j \in \{0\} \cup \mathcal{N} - \{j'\}} \beta_{jj'} = 1 \qquad j' \in \mathcal{N} \tag{16}$$

$$\sum_{j' \in \mathcal{N} - \{j\}} \beta_{jj'} \leq 1 \qquad j \in \{0\} \cup \mathcal{N} \tag{17}$$

$$\sum_{w \in \mathcal{R}} X_{ijw} = 1 \qquad i \in \mathcal{M}, j \in \mathcal{N} \tag{18}$$

$$\lambda_{iji'j'} + \lambda_{i'j'ij} \geq X_{ijw} + X_{i'j'w} - 1 \qquad i \in \mathcal{M}, i^{'} \in \mathcal{M} - \{i\}, j \in \mathcal{N}, j^{'} \in \mathcal{N} - \{j\}, w \in \mathcal{R} \tag{19}$$

$$B_{ij'} \geq C_{ij} - V \cdot (1 - \beta_{jj'}) \qquad i \in \mathcal{M}, j \in \{0\} \cup \mathcal{N}, j^{'} \in \mathcal{N} - \{j\} \tag{20}$$

$$B_{i'j'} \geq B_{ij} + \sum_{j''\{0\} \cup \mathcal{N} - \{j\}} (s_{ij''j} \cdot \beta_{j''j}) - V \cdot (1 - \lambda_{iji'j'}) \qquad i \in \mathcal{M}, i^{'} \in \mathcal{M} - \{i\}, j \in \mathcal{N}, j^{'} \in \mathcal{N} - \{j\} \tag{21}$$

$$C_{ij} \geq B_{ij} + \sum_{j'\{0\} \cup \mathcal{N} - \{j\}} (s_{ij'j} \cdot \beta_{j'j}) + p_{ij} \qquad i \in \mathcal{M}, j \in \mathcal{N} \tag{22}$$

$$C_{i-1,j} \leq B_{ij} \qquad i \in \mathcal{M} - \{1\}, j \in \mathcal{N} \tag{23}$$

$$C_{max} \geq C_{m,j} \qquad j \in \mathcal{N} \tag{24}$$

The set of constraints (16) ensures that each job has a predecessor. Note that a dummy job is introduced to represent the predecessor of the first job in the sequence. Furthermore, each job (including the dummy job) has at most one successor, which is bounded by constraints (17). The set (18) enforces that the setup of operation O_{ij} is assigned to a unique worker. Constraints (19) define binary variables $\lambda_{iji'j'}$. The sets (20) and (21) ensure that an operation has to start processing after its predecessor in the machine and after its predecessor in the worker, respectively. The set of constraints (22) defines the completion time of an operation and (23) ensures that an operation is not processed after its previous operation (in the previous stage) is completed. Finally, the set (24) defines the makespan of the sequence.

3.4. Model 3

Our last MILP model arises from Model 2, but modifies how workers are treated. In this case, we introduce variable $\eta_{wiji'j'}$ to define the sequence followed by each worker (the use of a related decision variable can be found in the first MILP model proposed in [61]). The formulation of this model is presented below:

Min C_{max}

subject to

$$\sum_{j \in \{0\} \cup \mathcal{N} - \{j'\}} \beta_{jj'} = 1 \qquad j' \in \mathcal{N} \tag{25}$$

$$\sum_{j' \in \mathcal{N} - \{j\}} \beta_{jj'} \leq 1 \qquad j \in \{0\} \cup \mathcal{N} \tag{26}$$

$$C_{1,j} \geq p_{1,j} + \sum_{j' \{0\} \cup \mathcal{N} - \{j\}} (s_{1j'j} \cdot \beta_{j'j}) \qquad j \in \mathcal{N} \tag{27}$$

$$C_{ij} - C_{i-1,j} \geq p_{ij} + \sum_{j' \{0\} \cup \mathcal{N} - \{j\}} (s_{ij'j} \cdot \beta_{j'j}) \qquad i \in \mathcal{M} - \{1\}, j \in \mathcal{N} \tag{28}$$

$$C_{ij'} - C_{ij} + V \cdot (1 - \beta_{jj'}) \geq p_{ij'} + s_{ijj'} \qquad i \in \mathcal{M}, j \in \{0\} \cup \mathcal{N}, j' \in \mathcal{N} - \{j\} \tag{29}$$

$$\sum_{w \in \mathcal{R}} \left(\eta_{w00i'j'} + \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{N}} \eta_{wiji'j'} \right) = 1 \qquad i' \in \mathcal{M}, i \in \mathcal{M} - \{i'\}, j' \mathcal{N}, j \in \mathcal{N} - \{j'\} \tag{30}$$

$$\sum_{w \in \mathcal{R}} \sum_{i' \in \mathcal{M} - \{i\}} \sum_{j' \in \mathcal{N} - \{j\}} \eta_{wiji'j'} \leq 1 \qquad i \in \mathcal{M}, j \in \mathcal{N} \tag{31}$$

$$\sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{N}} \eta_{w00ij} = 1 \qquad w \in \mathcal{R} \tag{32}$$

$$\eta_{w00ij} + \sum_{i' \in \mathcal{M} - \{i\}} \sum_{j' \in \mathcal{N} - \{j\}} \eta_{wi'j'ij} \geq \sum_{i' \in \mathcal{M} - \{i\}} \sum_{j' \in \mathcal{N} - \{j\}} \eta_{wiji'j'} \qquad i \in \mathcal{M}, j\mathcal{N}, w \in \mathcal{R} \tag{33}$$

$$C_{i'j'} - p_{i'j'} + \dot{V}(1 - \eta_{wiji'j'}) \geq C_{ij} - p_{ij} + \sum_{j'' \in \{0\} \cup \mathcal{N} - j'} (s_{i'j''j'} \cdot \beta_{j''j'}) \qquad w \in \mathcal{R}, i \in \mathcal{M}, i' \in \mathcal{M} - \{i\}, j \in \mathcal{N}, j' \in \mathcal{N} - \{j\} \tag{34}$$

$$C_{ij} - p_{ij} + \dot{V}(1 - \eta_{w00ij}) \geq \sum_{j' \in \{0\} \cup \mathcal{N} - j} (s_{ij'j} \cdot \beta_{j'j}) \qquad w \in \mathcal{R}, i \in \mathcal{M}, j \in \mathcal{N} \tag{35}$$

$$C_{max} \geq C_{m,j} \qquad j \in \mathcal{N} \tag{36}$$

As in the previous model, constraints (25) and (26) fully define variables $\beta_{jj'}$. The completion times of the first machine are established in the set (27), while the constraints (28) and (29) bound the completion times of each job in the other machines. To do so, constraints (28) limit these completion times considering its previous machine, while constraints (29) consider its previous job. Variable $\eta_{wi'j'ij}$ is defined in Equations (30)–(33). More specifically, the set (30) ensures that the setup of each operation has either a predecessor in the same worker or a dummy setup. Constraints (31) achieve that each setup has at most a successor in a worker, while constraints (32) and (33) ensure that each worker starts with a setup. The completion times of the jobs and the precedence relationships between the operations are linked in constraints (34) and (35). Finally, the set of constraints (36) defines the makespan.

4. Proposed Decoding Procedures and Complete Enumeration

Due to the limitation of MILP models to solve medium-large sized instances, we also propose different approximate algorithms to tackle the problem (described in Section 5). Each of the proposed algorithms uses a sequence of jobs to easily represent the solutions, since each machine must process the jobs in the same order (permutation constraint, see Section 2). This sequence can be formally defined as follows:

- Sequence of jobs, $\Pi = (\pi_1, \ldots, \pi_k, \ldots, \pi_n)$: It represents the order that each machine has to follow to process the jobs, i.e., π_k has to be processed before job π_{k+1} in any machine i ($k \in \{1, \ldots, n-1\}$).

Obviously, for the problem under consideration, the same sequence of jobs could theoretically lead to different semi-active schedules depending on how and in which order the operations of the jobs are assigned to the workers. Therefore, in the proposed algorithms, it is necessary to include mechanisms to construct a unique schedule to represent the sequence. In this regard, the procedure to find a specific schedule (and consequently an objective function value) from a representation of solutions is denoted by decoding procedure . For the problem under consideration, a decoding procedure can be determined by establishing the following rules:

- Priority rule (PR): It is used to define which setup must be performed first by workers when there are several jobs waiting in the queue.
- Assignment rule (AR): It established the criterion for assigning a worker to a set-up. In this paper, as workers have the same skills, we use the FAW (First Available Worker) rule, which is the most common rule when workers/machines are identical (see [62]).

Therefore, given a sequence of jobs Π and the FAW rule, a different decoding procedure can be obtained by varying the priority rule. In this paper, we propose two families (each composed of six priority rules) to order the operations that are waiting to be processed (denoted as PR^S and PR^C). Each of these rules first selects an operation according to a certain criterion (either the operation which can start before, PR^S, or the operation to be completed before, PR^C), breaking ties using a specific mechanism. More specifically, we propose and compare the following priority rules:

- PR^S_J: This rule selects the operation whose setup can start before breaking the tie in favour of the operation in the lowest position of the sequence Π.
- PR^S_M: This rule selects the operation whose setup can start before breaking ties in favour of the operation that will be processed on the lowest machine index.
- PR^S_{SPI}: This rule selects the operation whose setup can start before breaking ties according to the operation with the lowest sum of setup and processing times.
- PR^S_{SPD}: This rule selects the operation whose setup can start before breaking ties in favour of the operation with the highest sum of setup and processing times.
- PR^S_{SI}: This rule selects the operation whose setup can start before breaking ties according to operation with the lowest setup time.
- PR^S_{SD}: This rule selects the operation whose setup can start before breaking ties in favour of the operation with the highest setup time.
- PR^C_J: This rule assigns the operation that can be completed before breaking ties in favour of the operation in the lowest position of the sequence Π.
- PR^C_M: This rule assigns the operation that can be completed before breaking ties in favour of the operation that will be processed on the lowest machine index.
- PR^C_{SPI}: This rule assigns the operation that can be completed before breaking ties according to the operation with the lowest sum of setup and processing times.
- PR^C_{SPD}: This rule assigns the operation that can be completed before breaking ties in favour of the operation with the highest sum of setup and processing times.
- PR^C_{SI}: This rule assigns the operation that can be completed before breaking ties according to the operation with the lowest setup time.
- PR^C_{SD}: This rule assigns the operation that can be completed before breaking ties in favour of the operation with the highest setup time.

Using a specific priority rule PR, the detailed procedure for decoding a sequence can be explained as follows. First, operation $O_{1[1]}$ (and its corresponding setup) is processed on the first machine. Once this operation is completed, an operation is selected according to the PR priority rule among the operations that are waiting to be processed (that is, $O_{2[1]}$ or $O_{1[2]}$). The procedure is repeated until there are no more operations available. In Figure 2, we show an example (following the previous example) of operations that are waiting to be processed ($O_{1[4]}$, $O_{2[3]}$, and $O_{3[2]}$) after the following operations have been completed: $O_{1[1]}$, $O_{1[2]}$, $O_{1[3]}$, $O_{2[1]}$, and $O_{2[2]}$. An example of the use of the PR^S_M priority rule to decode the sequence $(1, 2, 3, 4)$ is shown in Figure 1, while in Figure 3, the same sequence is decoded using the PR^S_J rule.

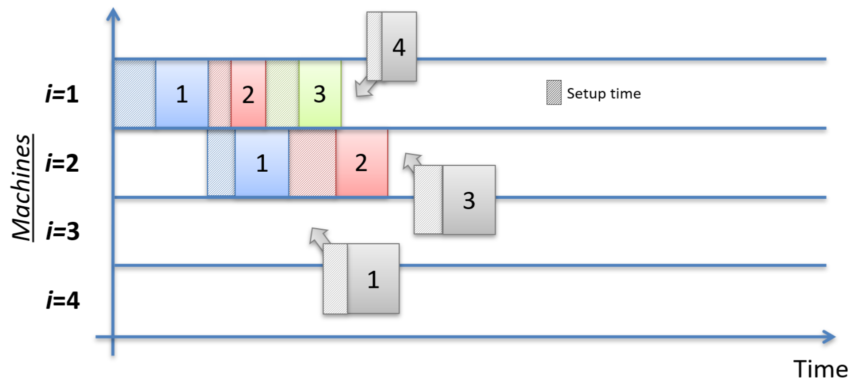

Figure 2. Example of operations that are waiting to be processed.

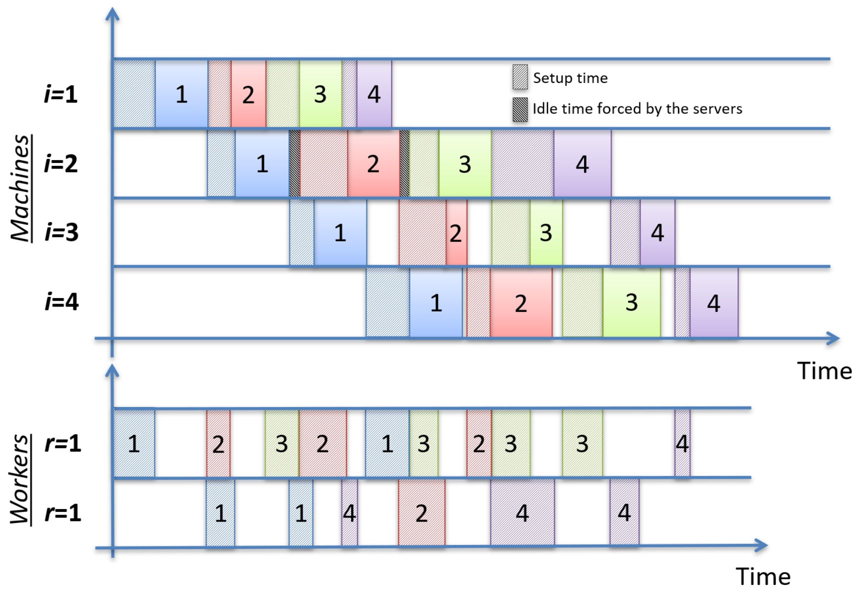

Figure 3. Example of using PR_J^S to decode a solution.

Obviously, the use of non-completed representations of the solutions with a specific decoding procedure (i.e., using rules to assign jobs to the worker instead of testing every possibility) does not guarantee the optimum of the problem, as many schedules are omitted. However, the solution space using these representations can be strongly reduced, which could be exploited by approximate algorithms to find solutions close to the optimum. To analyse the efficiency of each decoding procedure, we perform a complete enumeration by evaluating each solution in its solution space (composed by $n!$ sequences), whose computational results are shown in Section 7. Once all solutions are evaluated using a specific decoding procedure, the best objective function value found can be obtained.

5. Dispatching and Construction Procedures

In this section, we propose approximate algorithms to find fast and efficient solutions for the problem under consideration. As it is the first time that the problem is addressed, on the one hand, we start analysing the behaviour of very fast solutions (i.e., static dispatching rules). To this end, we propose a mechanism to generate 896 different dispatching rules in Section 5.1. On the other hand, we analyse the efficiency of the traditional NEH algorithm when it is adapted to the problem under consideration. The proposed variants of the NEH are explained in Section 5.2. All proposed approximate algorithms use the best decoding procedure found among the proposals, i.e., PR_{SI}^S (see Section 7.3).

5.1. Generation of Static Dispatching Rules

Dispatching rules are typically composed of an indicator value depending on the data of the instance and a sorting criterion to order the jobs. Generalising this idea, ref. [63] constructed and compared 176 different rules combining 22 indicators with 8 sorting criteria. All these indicators were constructed using functions of a unique measure of the original instance, the processing times of the jobs. In this paper, we extend this idea by also changing the starting measure. Thereby, we generate dispatching rules by combining the following issues:

- Measure , χ_{ij}: It represents the contribution of the operation O_{ij} to the dispatching rules and is denoted by χ_{ij}. χ_{ij} is obtained from the instance data.
- Indicator, ψ_j: Using a specific measure for each operation O_{ij}, we can define an indicator to characterise each job j, denoted by ψ_j. Then, ψ_j can be constructed as a function of χ_{ij}.
- Sorting criterion, Ω: Once an indicator for each job is generated (i.e., ψ_j), the sorting criterion established the procedure to order the jobs according to indicator ψ_j. Let Ω be the sequence obtained after applying such a sorting criterion.

As measures, we consider seven different variants:

- Measure P: It directly uses the processing times to define χ_{ij}: $\chi_{ij} = p_{ij}$.
- Measure S: The average setup time is used for each operation O_{ij}: $\chi_{ij} = \frac{\sum_{j'=1}^{n} s_{ij'j}}{n}$.
- Measure MS: For each operation, it considers the maximal setup time that the operation could need: $\chi_{ij} = \max_{\forall j'} s_{ij'j}$.
- Measure mS: For each operation, it considers the minimal setup time that the operation could need: $\chi_{ij} = \min_{\forall j'} s_{ij'j}$.
- Measure PS: In this case, each operation is defined by its processing time plus its average setup time: $\chi_{ij} = p_{ij} + \frac{\sum_{j'=1}^{n} s_{ij'j}}{n}$.
- Measure PMS: Each operation is defined by its processing time plus its maximum setup time: $\chi_{ij} = p_{ij} + \max_{\forall j'} s_{ij'j}$.
- Measure PmS: Each operation is defined by its processing time plus its minimal setup time: $\chi_{ij} = p_{ij} + \min_{\forall j'} s_{ij'j}$.

As indicators, based on [63], we consider 16 different variants:

- Indicator SUM: $\psi_j = \sum_{i=1}^{m} \chi_{ij}$.
- Indicator WSUM: $\psi_j = \sum_{i=1}^{m} (m - i + 1)\chi_{ij}$.
- Indicator ABS: $\psi_j = \sum_{i=1}^{m} \sum_{j'=1}^{n} |\chi_{ij} - \chi_{ij'}|$.
- Indicator WABS: $\psi_j = \sum_{i=1}^{m} \sum_{j'=1}^{n} (m - i + 1)|\chi_{ij} - \chi_{ij'}|$.
- Indicator SRA: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} |r_{ijj'}|$, where $r_{ijj'} = \chi_{ij} - \chi_{i-1,j'}$.
- Indicator WSRA: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (m - i + 1)|r_{ijj'}|$.
- Indicator SRS: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (r_{ijj'})^2$.
- Indicator WSRS: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (m - i + 1)(r_{ijj'})^2$.
- Indicator SRN: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} |\min(r^*_{ijj'}, 0)|$, where $r^*_{ijj'} = \chi_{ij} - \chi_{i-1,j'} + \min\{r_{i-1,j,j'}\}$.
- Indicator WSRN: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (m - i + 1)|\min(r^*_{ijj'}, 0)|$.
- Indicator SRAN: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} |r^*_{ijj'}|$.
- Indicator WSRAN: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (m - i + 1)|r^*_{ijj'}|$.
- Indicator SRSN: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (r^*_{ijj'})^2$.
- Indicator WSRSN: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (m - i + 1)(r^*_{ijj'})^2$.
- Indicator SRAN2: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (\max(r_{ijj'}, 0) + 2|\min(r_{ijj'}, 0)|)$.
- Indicator WSRAN2: $\psi_j = \sum_{\forall j' \in \mathcal{N}-j} \sum_{i=2}^{m} (m - i + 1)(\max(r_{ijj'}, 0) + 2|\min(r_{ijj'}, 0)|)$.

As sorting criteria (Ω), we use the following eight variants taken from [63]:

- Sorting Criterion C: Jobs are ordered in non-decreasing order of ψ_j. Let $\Omega^C = (\omega_1^C, \ldots, \omega_n^C)$ denote such a sequence.
- Sorting Criterion D: Jobs are ordered according to non-increasing order of ψ_j. Let $\Omega^D = (\omega_1^D, \ldots, \omega_n^D)$ denote the obtained order.
- Sorting Criterion H: The jobs are sorted as a 'hill' of their ψ_j values. Denoting this sequence by $\Omega^H = (\omega_1^H, \ldots, \omega_n^H)$, it can be calculated by $\Omega^H = (\omega_1^C, \omega_3^C, \ldots, \omega_4^C, \omega_2^C)$.
- Sorting Criterion V: Jobs are sorted as a 'valley' of their ψ_j values. Denoting this sequence by $\Omega^V = (\omega_1^V, \ldots, \omega_n^V)$, it can be calculated by $\Omega^H = (\omega_1^D, \omega_3^D, \ldots, \omega_4^D, \omega_2^D)$.
- Sorting Criterion HIH: Jobs are sorted by first choosing the job with maximal ψ_j, and then the job with minimal ψ_j. The procedure is repeated until all jobs are selected. Denoting by Ω^{HIH} the sequence obtained, it can be calculated by $\Omega^{HIH} = (\omega_1^D, \omega_n^D, \omega_2^D, \omega_{n-1}^D, \ldots,)$.
- Sorting Criterion HIL: Similarly as in HIH, jobs are sorted by firstly choosing the job with minimal ψ_j, and then the job with maximal ψ_j. Denoting this sequence by Ω^{HIL}, it can be calculated by $\Omega^{HIL} = (\omega_n^D, \omega_1^D, \omega_{n-1}^D, \omega_2^D, \ldots,)$.
- Sorting Criterion LOH: Jobs sorted by reverse sequence Ω^{HIH}, that is, $\Omega^{LOH} = (\omega_1^{LOH}, \ldots, \omega_n^{LOH}) = (\omega_n^{HIH}, \ldots, \omega_1^{HIH})$.
- Sorting Criterion LOL: Jobs sorted reversing sequence Ω^{HIL}, i.e., $\Omega^{LOL} = (\omega_1^{LOL}, \ldots, \omega_n^{LOL}) = (\omega_n^{HIL}, \ldots, \omega_1^{HIL})$.

Combining all previous measures, indicators, and sorting criteria (i.e., by a fixed $\{\chi_{ij}, \psi_j, \Omega\}$), we can construct 896 different sequences, which are evaluated in Section 7.4. Many of them are very common static dispatching rules proposed in the literature, such as, e.g., Shortest Processing Time (SPT) rule ([64]) by using $\{P, SUM, C\}$; Longest Processing Time (LPT) rule ([65]) by using $\{P, SUM, D\}$; SPT-LPT ([66]) by using $\{P, SUM, H\}$; Shortest Setup Time (SST) rule ([21]) by using $\{S, SUM, H\}$; or the dispatching rule applied by [67] using $\{P, WSUM, C\}$. In the following section, these dispatching rules are included as an initialisation of more advanced heuristics.

5.2. Constructive Heuristics: NEHV

In the flow-shop literature, the NEH heuristic (which was first proposed by [24], and adapted for the PFSP with sequence-dependent setup times by [68]) is clearly the cornerstone heuristic to solve flow-shop scheduling problems, which can be explained by the following two issues: (1) many of the state-of-the-art heuristics nowadays for this kind of layout are variants of the classical proposal (see, e.g., [32,69,70] among many others); (2) it is also the base of more complex metaheuristics (see e.g., [39,71–73]). The original NEH heuristic is a greedy approximate algorithm composed of two phases. In the first phase, the jobs are ordered according to the LPT rule ($\Pi^{LPT} := \{\pi_1^{LPT}, \ldots, \pi_n^{LPT}\}$). Then, a partial sequence, denoted by Π^P, is generated with only the first element of Π^{LPT}, i.e., $\Pi^P := (\pi_1^{LPT})$. Examples modifying this phase in the literature can be found, e.g., in [74–79]. The second phase is an insertion procedure to iteratively construct a complete sequence. This phase starts by testing the second job of the initial rule (π_2^{LPT}) in all available positions of Π^P. The position that yields the minimum makespan is chosen to insert π_2^{LPT}, and the new partial sequence overwrites Π^P. This insertion step is repeated iteratively with the rest of the jobs in Π^{LPT} (i.e., with π_k^{LPT} with $k \in \{3, n\}$). Examples modifying this phase in the literature can be found, e.g., in [69,80–83].

In this paper, we propose different variants of this classical NEH algorithm, denoted by NEHV($\{\chi_{ij}, \psi_j, \Omega\}$), by modifying its initial phase. The idea is twofold: first, to analyse the limit and influence of the initial order in the classical NEH heuristic; and secondly to determine the best initial sequence to start more advanced algorithms. To deal with these issues, our proposal is focused on the first phase of the algorithm (see [63,74,77] with similar approaches that also deal with the first phase of the NEH algorithms in the literature). More specifically, we propose different NEHV heuristics by replacing the first phase of the NEH algorithm for the following steps:

1. Define a measure χ_{ij} to identify each operation.

2. Define an indicator ψ_j to identify each job.
3. Define a specific sorting criterion Ω.
4. Denote by Π^{ini} the sequence obtained by ordering the jobs according to $\{\chi_{ij}, \psi_j, \Omega\}$.
5. Denote by Π^{P} the initial partial sequence composed of job π_1^{ini}

Combining all the measures, indicators, and sorting criteria from the previous section, we can propose 896 different NEHV. A general pseudo-code of these proposals is described in Algorithm 1.

Algorithm 1: Pseudo-code of the proposed NEH algorithm.

```
Procedure NEHV($\{\chi_{ij}, \psi_j, \Omega\}$)
    //Phase I
    $\Pi^{ini}$ := Jobs ordered by using dispatching rule $\{\chi_{ij}, \psi_j, \Omega\}$;
    $\Pi^{P}$ := $\{\pi_1^{ini}\}$;
    //Phase II
    for $k = 2$ to $n$ do
        Test job $\pi_k^{ini}$ in any position of $\Pi^{P}$.
        $\Pi^{P}$ := partial sequence obtained by inserting $\pi_k^{ini}$ in the position of $\Pi^{P}$
          with the lowest makespan;
    end
end
```

6. Iterated Greedy Algorithm, IGV

In this section, we detail the proposed iterated-greedy-based metaheuristic. The iterated greedy algorithm is a single-solution-based metaheuristic, which iteratively removes several jobs from a sequence (destruction phase) and re-inserts them using a greedy procedure (construction phase). Traditionally, this destruction-construction procedure is followed by a local search and a simulated annealing procedure. This metaheuristic was first proposed by [84] and since then is one of the best performing metaheuristics in flow-shop-based scheduling problems. Examples of state-of-the-art iterated greedy algorithms can be found in [42,85–90].

The proposed iterated greedy algorithm, denoted as IGV, is composed of the following steps:

STEP 1: *Initial Solution*. We employ a constructive heuristic that explores the reduced solution space obtained by PR_{SI}^{S}. More specifically, we use the best NEHV proposed, i.e., NEHV({PMS,ABS,D}) (see Section 7.6 for more details).

STEP 2: *Local search applying PR_{SI}^{S}* (denoted as LS). Once a sequence of jobs is obtained by NEHV({PMS,ABS,D}), a traditional insertion local search is implemented, using the PR_{SI}^{S} procedure. This search removes a job of the sequence and re-inserts it in the best position of the sequence. This procedure is repeated until all jobs are tested. Additionally, the insertion local search is completely repeated until no improvement is found.

STEP 3: *Iterations*. The following steps are repeated until the stopping criterion is reached:

(a) *Destruction phase*. In this phase, we removed d random jobs from the iteration sequence.

(b) *Construction phase*. After the previous phase is completed and a partial sequence is obtained without the destructed jobs, these destructed jobs are iteratively re-inserted in the best position of that sequence, until a complete sequence is constructed.

(c) *Local search applying PR_{SI}^{S}*. The LS method is applied to the previous sequence to find a local optimum, denoted as $\Pi^{'}$ in the solution space obtained using the PR_{SI}^{S} procedure.

(d) *Intensive local search fixing the sequence* (denoted as ILS). Each time a local optimum has been found in the previous local search, we explore the neighbourhood of this solution by changing the decoding procedure. More specifically, we obtain the order of operations that have been processed in the shop to obtain this local optimum. Then, each operation (following a random order) is re-inserted and evaluated in all available positions, i.e., without altering the sequence of jobs in each machine and fulfilling the route of operations.

(e) *Simulated annealing procedure*. The last step is a simple simulated annealing procedure to determine the reference sequence for the next iteration. We directly apply the original procedure proposed by ([84]), which depends on the parameter T. Basically, the sequence obtained in Step 3c is kept if $random \leq exp\{(Cmax - Cmax^r)/Temperature\}$ (being $random$ a randomly generated number between 0 and 1, $Cmax$ the makespan of sequence Π', while $Cmax^r$ is the reference makespan). Finally, $Temperature$ depends on parameter T according to the following formula:

$$Temperature = T \times \frac{\sum_{\forall n}\sum_{\forall m} p_{ij}}{n \times m \times 10}$$

A complete pseudo-code of the proposed algorithm is shown in Algorithm 2.

Algorithm 2: Proposed iterated greedy algorithm.

```
Procedure IGV
    //Initial Solution
    $\Pi^i$ = NEHV({PMS,ABS,D});
    //Local Search LS
    $\Pi^{ii}$ = LS($\Pi^i$);
    $\Pi^r := \Pi^{ii}$;
    $C^r_{max} := C_{max}(\Pi^r)$;
    $\Pi^B := \Pi^{ii}$;
    $C^B_{max} := C_{max}(\Pi^r)$;
    while stopping criterion is not met do
        //Destruction Phase
        $\Pi$:= Sequence obtained after removing $d$ random jobs on $\Pi^r$;
        //Construction Phase
        for $i = 1$ to $d$ do
            $\Pi :=$ Sequence obtained after inserting the $i$th removed job in the best
              position of $\Pi$;
        end
        //Local Search LS
        $\Pi' = $ LS($\Pi$);
        //Simple simulated annealing procedure and intensive local search
        if $C_{max}(\Pi') < C^r_{max}$ then
            $\Pi^r := \Pi'$;
            $C^r_{max} := C_{max}(\Pi')$;
            $\Pi'' :=$ ILS($\Pi^r$);
            if $C_{max}(\Pi'') < C^B_{max}$ then
                $C^B_{max} := C_{max}(\Pi'')$;
                $\Pi^B := \Pi''$;
            end
        else if $random \leq exp\{-(C_{max}(\Pi') - C^r_{max})/Temperature\}$ then
            $\Pi^r := \Pi'$;
            $C^r_{max} := C_{max}(\Pi')$;
        end
    end
end
```

7. Computational Results

In this section, we present the computational evaluations carried out in our study. A total of four experiments have been performed: first, we compare the performance of the proposed MILP models in Section 7.2; second, the proposed decoding procedures are compared in Section 7.3 by analysing all their complete solutions (complete enumeration procedure); thirdly, the efficiency of each proposed dispatching rule is tested in Section 7.4; fourthly, the constructive heuristics proposed are analysed in Section 7.6; then, the iterated greedy algorithm is calibrated and its performance is evaluated in Section 7.6. To perform all these experiments, we generate three sets of benchmarks explained in Section 7.1. All procedures tested have been compared using the same computer conditions: on the same Intel Core i7-3770 with 3.4 GHz and 16 GB RAM; under the same programming languages (C# and Gurobi 9.5.0); and by the same person and using the same common functions and libraries. Finally, a sensitivity analysis is performed in Section 7.7 to study the impact of each factor on the problem.

7.1. Benchmarks

In this section, we describe three sets of instances generated to test the performance of the proposals. The first set, denoted by β_1, is specifically constructed to compare exact methods, i.e., the proposed MILP models (see Section 3) and the complete enumerations for each of the proposed decoding procedure (see Section 7.3). The second set, denoted by β_2 is constructed to compare approximate methods, i.e., the proposed dispatching rules (see Section 5.1), the proposed NEHV (see Section 5.2), and to evaluated the performance of the proposed IGV (see Section 6). Finally, a last benchmark, denoted by β_3, is generated to calibrated IGV.

- β_1 for comparison of exact methods. As the complete enumeration procedure has a high computational requirement (due to the evaluation of all solutions), the set is composed of 540 small-sized instances with processing and setup times following uniform distributions from $[1,99]$ and $[1,\gamma]$, respectively ([14,91,92]). γ is a parameter of the testbed in the range $\gamma \in \{25,50,100\}$ ([14]). The other parameters are generated for the following values: $n \in \{4,5,6,7,8,9\}$, $m \in \{3,4,5\}$, and $r \in \{1,2\}$. Finally, five instances are constructed for each combination of the parameters.
- β_2 for comparison of approximate methods: β_2 is a set of medium and large instances. This benchmark is generated following a procedure similar to β_1. Thereby, p_{ij} and s_{ilj} follow the uniform distributions $[1,99]$ and $[1,\gamma]$, respectively. Regarding the parameters of the testbed, five instances are generated for each combination of the following parameters: $\gamma \in \{25,50,100\}$, $n \in \{50,100,150,200\}$, $m \in \{10,20\}$, and $r = r' \cdot m$, with $r' \in \{0.1,0.2,0.3,0.4,0.5\}$.
- β_3 for calibration of the iterated greedy algorithm. Similarly as in β_2, we generate two instances for each combination of the following parameters: $\gamma \in \{25,50,100\}$, $n \in \{50,100,150,200\}$, $m \in \{10,20\}$, and $r = r' \cdot m$, with $r' \in \{0.1,0.2,0.3,0.4,0.5\}$.

7.2. Computational Evaluation: Comparison of MILP Models

In this section, we test the performance of the MILP models proposed in Section 3. The proposed models are solved using Gurobi 9.5.0 with 500 seconds as the stopping criterion in each instance of the β_1 set. The models are then compared in terms of the quality of the solution by the Relative Percentage Deviation (RPD_{pi}) indicator (see Equation (37)). RPD_{pi} measures (in percentage) the difference between the objective function found by procedure p in instance i (OF_{hi}) and the best solution found in this instance (denoted $best_i$). In addition, to compare the computational effort required by the MILP models, we use the average CPU time. Both indicators are summarised in Table 2, grouped by n, m, r, and γ. Finally, in Table 3, we also present the number of optimal, feasible, and no solutions found.

$$RPD_{pi} = \frac{OF_{hi} - best_i}{best_i} 100 \tag{37}$$

According to these computational results, we can observe that the best performance is found in Model 1. This model clearly outperforms the other MILP models in terms of average RPD (denoted ARPD) and CPU times. In this regard, Model 1 has an ARPD value of 0.01 (in the whole benchmark), while Models 2 and 3 found 0.14 and 0.56, respectively. This difference is statistically significant with a p-value of 0.004, using a non-parametric Mann-Whitney test between Model 1 and 2. With respect to average CPU times, we can observe the same trend. Model 1 requires a lower average CPU (61.61 seconds) as compared to Model 2 (84.78) and Model 3 (214.48). Similarly, Models 1 and 2 clearly outperform Model 3 in terms of number of optimal solutions, both finding a total of 487 optimal solutions versus 332 by Model 3.

Table 2. Computational results of the proposed MILP models: ARPD values and CPU times (in seconds).

Parameter	Value	ARPD			CPU Time (s)		
		Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
n	4	0.00	0.00	0.00	0.24	0.27	2.03
	5	0.00	0.00	0.00	2.54	0.66	19.90
	6	0.00	0.00	0.35	19.18	6.24	75.67
	7	0.00	0.00	1.01	63.08	22.66	272.94
	8	0.04	0.21	2.68	112.70	86.66	421.57
	9	0.02	0.61	3.65	171.94	392.16	494.75
m	3	0.00	0.00	0.00	6.85	50.19	134.54
	4	0.00	0.06	0.25	56.17	84.77	221.45
	5	0.03	0.35	1.60	121.82	119.36	287.44
r	1	0.01	0.26	0.55	91.13	104.58	213.69
	2	0.01	0.01	0.57	32.10	64.97	215.26
γ	25	0.00	0.00	0.02	5.30	71.31	204.16
	50	0.01	0.06	0.42	54.59	79.43	207.71
	100	0.02	0.35	1.18	124.96	103.59	231.55
Average		0.01	0.14	0.56	61.61	84.78	214.48

Table 3. Computational results of the proposed MILP models: optimal solutions (#O), feasible solutions (#F), and no solution is found (#N).

Parameter	Value	Model 1			Model 2			Model 3		
		#O	#F	#N	#O	#F	#N	#O	#F	#N
n	4	90	0	0	90	0	0	90	0	0
	5	90	0	0	90	0	0	88	2	0
	6	87	3	0	90	0	0	85	5	0
	7	82	8	0	88	2	0	47	15	28
	8	73	17	0	84	6	0	19	5	66
	9	65	25	0	45	45	0	3	10	77
m	3	179	1	0	176	4	0	138	4	38
	4	164	16	0	164	16	0	108	9	63
	5	144	36	0	147	33	0	86	24	70
r	1	229	41	0	232	38	0	166	23	81
	2	258	12	0	255	15	0	166	14	90
γ	25	180	0	0	165	15	0	114	2	64
	50	165	15	0	166	14	0	114	10	56
	100	142	38	0	156	24	0	104	25	51
All		487	53	0	487	53	0	332	37	171

7.3. Computational Evaluation: Complete Enumeration Applying Different Decoding Procedures

In this section, using the same set of instances β_1, we compare the decoding procedures proposed in Section 4. Each procedure is embedded in a complete enumeration procedure and the best solution found is compared using the RPD indicator (see Equation (37), where, again, $best_i$ is the best solution found in instance i, i.e., considering both MILP models and the complete enumeration procedures). The computational results are demonstrated in Table 4. The best ARPD (2.32) is found using the PR^S_{SI} procedure of the PR^S family. In this family, the tie-breaking mechanisms have a high influence on reducing the ARPD value from 3.76 (PR^S_{SD} procedure) to 2.32, while these mechanisms are not as relevant in the PR^C family, oscillating the ARPD from 7.65 to 7.76. Despite the excellent performance found by PR^S_{SI}, it is not statistically significant with PR^S_M (the second best proposal), according to the non-parametric Mann-Whitney test (p-value equals to 0.221). However, with the exception of PR^S_M and PR^S_{SPD} (whose ARPD is similar to 2.64 and 2.65, respectively), it is statistically significant with all other decoding procedures (in this regard, when compared with the PR^S_{SPI} procedure, the p-value found in the Mann-Whitney test is 0.000).

Table 4. Computational results in small-sized instances: ARPD values of the proposed decoding procedures, the best NEHV (i.e., NEHV(PMS,ABS,D)), and IGV.

Parameter	Value	PR^S_J	PR^S_M	PR^S_{SPI}	PR^S_{SPD}	PR^S_{SI}	PR^S_{SD}	PR^C_J	PR^C_M	PR^C_{SPI}	PR^C_{SPD}	PR^C_{SI}	PR^C_{SD}	NEHV	IGV
n	4	3.02	2.89	3.49	2.21	2.17	3.41	6.54	6.49	6.49	6.49	6.54	6.49	3.86	1.92
	5	4.08	2.96	3.85	3.11	2.57	4.33	8.04	8.04	8.15	7.96	8.01	8.07	5.53	2.02
	6	4.08	2.80	3.76	2.96	2.60	4.31	7.87	7.85	7.85	7.83	7.86	7.87	6.35	1.97
	7	4.06	2.61	3.47	2.62	2.39	3.62	7.78	7.81	7.83	7.74	7.83	7.75	7.61	1.42
	8	3.35	2.36	3.12	2.68	2.20	3.54	7.82	7.83	7.96	7.68	7.82	7.87	7.80	1.48
	9	3.56	2.24	2.92	2.31	1.98	3.34	8.20	8.27	8.27	8.20	8.17	8.27	9.21	1.27
m	3	3.39	1.67	2.41	2.00	1.43	3.03	4.73	4.74	4.78	4.69	4.69	4.78	6.15	1.04
	4	3.73	2.91	3.47	2.76	2.47	3.95	7.64	7.69	7.68	7.65	7.64	7.66	6.80	1.85
	5	3.95	3.36	4.43	3.19	3.06	4.30	10.74	10.72	10.82	10.61	10.78	10.71	7.23	2.16
r	1	5.94	4.15	5.33	4.26	3.33	6.26	12.43	12.47	12.51	12.37	12.41	12.49	8.13	2.52
	2	1.44	1.14	1.54	1.04	1.31	1.26	2.99	2.96	3.01	2.93	3.00	2.95	5.32	0.85
γ	25	2.50	1.83	2.81	1.49	1.40	2.78	4.45	4.50	4.49	4.44	4.45	4.50	4.36	0.88
	50	4.09	2.90	3.79	2.85	2.47	4.18	7.26	7.23	7.32	7.17	7.25	7.26	6.61	1.73
	100	4.49	3.20	3.70	3.60	3.10	4.31	11.40	11.42	11.47	11.34	11.42	11.40	9.20	2.43
Average		3.69	2.64	3.43	2.65	2.32	3.76	7.71	7.72	7.76	7.65	7.71	7.72	6.73	1.68

7.4. Computational Evaluation: Dispatching Rules

In this section, we compare the dispatching rules generated following the procedure described in Section 5.1. Using this procedure, we construct 896 (potentially) different sequences of jobs in each instance, based on the specific indicator, measure, and sorting criterion applied. To generate a schedule (and consequently to obtain an objective function value), we apply the best decoding procedure (i.e., the PR^S_{SI} procedure) to each proposed dispatching rule as mentioned previously. Then, each dispatching rule is tested in the set of instances β_2 and its performance is evaluated by the RPD indicator (see Equation (37)). In the same way, $best_i$ is the best solution found in instance i, considering both dispatching rules and NEHV variants. The computational results are shown in Tables 5–7. All ARPD values for the proposed dispatching rules are obtained between 23.97 and 28.75. Regarding the best results, they are found by dispatching rules using the WSUM indicator, the C sorting criterion, and any measure that considers the processing times of the operations (i.e., P, PS, PMS, PmS). Thereby, the ARPD values of this group of dispatching rules are 24.01, 23.97, 24.00, and 24.10 (for {P,WSUM,C}, {PS,WSUM,C}, {PMS,WSUM,C}, and {PmS,WSUM,C}, respectively). In fact, there is a statistically significant difference between this group of rules and {P,SUM,C} (the following best rule with an ARPD of 25.28), finding a p-value of 0.000 (using the non-parametric Mann-Whitney test).

Table 5. Computational results for the dispatching rules: ARPD values I.

Measure	Indicator	Sorting							
		C	**D**	**H**	**V**	**HIH**	**HIL**	**LOH**	**LOL**
P	SUM	26.42	26.30	25.28	27.25	26.48	26.35	26.34	26.20
P	WSUM	24.01	28.64	25.60	27.24	26.47	26.20	26.32	26.44
P	ABS	26.40	26.23	26.23	26.57	26.34	26.28	26.37	26.27
P	WABS	26.02	26.52	26.43	26.62	26.31	26.23	26.24	26.38
P	SRA	26.46	26.40	26.42	26.44	26.27	26.21	26.45	26.37
P	WSRA	25.98	26.58	26.30	26.40	26.29	26.28	26.27	26.34
P	SRS	26.43	26.11	26.31	26.30	26.34	26.22	26.30	26.34
P	WSRS	25.93	26.65	26.22	26.31	26.28	26.38	26.24	26.31
P	SRN	26.22	27.25	26.62	26.42	26.49	26.61	26.40	26.50
P	WSRN	28.13	26.13	26.55	26.92	26.13	26.26	26.50	26.47
P	SRAN	26.15	27.17	26.33	26.50	26.45	26.47	26.27	26.33
P	WSRAN	27.34	26.53	26.15	26.89	26.01	25.96	26.78	26.89
P	SRSN	26.23	27.16	26.23	26.51	26.46	26.42	26.32	26.30
P	WSRSN	27.31	26.55	26.46	27.12	25.92	26.06	26.90	26.92
P	SRAN2	25.85	27.21	26.58	26.11	26.49	26.42	26.26	26.25
P	WSRAN2	27.39	25.86	26.45	26.67	26.21	26.22	26.75	26.70
S	SUM	26.33	26.34	26.25	26.23	26.16	26.21	26.32	26.34
S	WSUM	26.08	26.33	26.36	26.38	26.25	26.12	26.21	26.27
S	ABS	26.32	26.34	26.29	26.38	26.30	26.32	26.34	26.28
S	WABS	26.40	26.14	26.28	26.40	26.18	26.22	26.21	26.23
S	SRA	26.24	26.22	26.27	26.32	26.22	26.30	26.36	26.34
S	WSRA	26.28	26.35	26.26	26.39	26.17	26.12	26.27	26.46
S	SRS	26.24	26.35	26.31	26.19	26.20	26.18	26.41	26.34
S	WSRS	26.34	26.45	26.30	26.33	26.29	26.21	26.27	26.51
S	SRN	26.21	26.35	26.16	26.24	26.34	26.25	26.20	26.06
S	WSRN	26.26	26.19	26.18	26.37	26.26	26.28	26.28	26.28
S	SRAN	26.39	26.33	26.34	26.37	26.20	26.32	26.29	26.28
S	WSRAN	26.38	26.13	26.20	26.36	26.43	26.29	26.38	26.48
S	SRSN	26.37	26.18	26.33	26.51	26.30	26.33	26.45	26.35
S	WSRSN	26.26	26.21	26.20	26.27	26.24	26.30	26.32	26.35
S	SRAN2	26.26	26.20	26.15	26.23	26.14	26.21	26.37	26.23
S	WSRAN2	26.23	26.33	26.22	26.24	26.32	26.28	26.31	26.28
MS	SUM	26.30	26.18	26.32	26.37	26.33	26.24	26.21	26.34
MS	WSUM	26.20	26.29	26.25	26.38	26.35	26.36	26.36	26.30
MS	ABS	26.24	26.29	26.29	26.12	26.28	26.22	26.21	26.20
MS	WABS	26.24	26.33	26.41	26.12	26.36	26.25	26.28	26.26
MS	SRA	26.23	26.27	26.29	26.18	26.33	26.34	26.23	26.33
MS	WSRA	26.33	26.44	26.29	26.10	26.18	26.23	26.13	26.15
MS	SRS	26.30	26.34	26.34	26.06	26.27	26.43	26.16	26.16
MS	WSRS	26.31	26.27	26.27	26.11	26.33	26.27	26.14	26.23
MS	SRN	26.33	26.29	26.37	26.32	26.26	26.34	26.29	26.19
MS	WSRN	26.28	26.31	26.30	26.14	26.25	26.19	26.28	26.28
MS	SRAN	26.29	26.17	26.23	26.14	26.34	26.33	26.21	26.30
MS	WSRAN	26.32	26.23	26.31	26.11	26.20	26.38	26.20	26.30
MS	SRSN	26.29	26.35	26.21	26.19	26.31	26.25	26.27	26.27
MS	WSRSN	26.18	26.30	26.36	26.12	26.20	26.08	26.20	26.25
MS	SRAN2	26.23	26.28	26.31	26.19	26.27	26.36	26.23	26.20
MS	WSRAN2	26.24	26.37	26.32	26.12	26.21	26.34	26.25	26.27

Table 6. Computational results for the dispatching rules: ARPD values II.

Measure	Indicator	Sorting							
		C	D	H	V	HIH	HIL	LOH	LOL
mS	SUM	26.46	26.51	26.27	26.39	26.37	26.38	26.22	26.18
mS	WSUM	26.31	26.40	26.29	26.36	26.39	26.29	26.20	26.29
mS	ABS	26.39	26.40	26.24	26.28	26.30	26.36	26.23	26.28
mS	WABS	26.52	26.32	26.34	26.22	26.32	26.27	26.18	26.33
mS	SRA	26.42	26.30	26.24	26.34	26.39	26.26	26.07	26.11
mS	WSRA	26.44	26.39	26.23	26.29	26.22	26.17	26.18	26.30
mS	SRS	26.37	26.45	26.14	26.25	26.34	26.29	26.20	26.30
mS	WSRS	26.45	26.41	26.29	26.30	26.43	26.31	26.22	26.38
mS	SRN	26.35	26.16	26.29	26.25	26.33	26.26	26.36	26.38
mS	WSRN	26.33	26.26	26.29	26.30	26.18	26.25	26.30	26.27
mS	SRAN	26.44	26.33	26.32	26.40	26.26	26.33	26.23	26.21
mS	WSRAN	26.39	26.34	26.28	26.31	26.28	26.37	26.29	26.28
mS	SRSN	26.32	26.34	26.22	26.33	26.24	26.35	26.30	26.22
mS	WSRSN	26.40	26.36	26.28	26.23	26.34	26.37	26.17	26.23
mS	SRAN2	26.38	26.38	26.27	26.35	26.43	26.33	26.19	26.27
mS	WSRAN2	26.54	26.30	26.18	26.22	26.33	26.29	26.10	26.34
PS	SUM	26.58	26.33	25.36	27.17	26.53	26.34	26.47	26.42
PS	WSUM	23.97	28.75	25.46	27.20	26.63	26.25	26.43	26.44
PS	ABS	26.38	26.31	26.30	26.59	26.16	26.40	26.31	26.30
PS	WABS	26.15	26.67	26.25	26.45	26.23	26.33	26.21	26.34
PS	SRA	26.38	26.34	26.37	26.38	26.24	26.33	26.26	26.34
PS	WSRA	26.01	26.64	26.17	26.30	26.39	26.28	26.28	26.29
PS	SRS	26.46	26.19	26.31	26.28	26.14	26.29	26.32	26.31
PS	WSRS	25.93	26.58	26.34	26.42	26.39	26.41	26.21	26.23
PS	SRN	26.24	27.41	26.70	26.43	26.42	26.49	26.32	26.47
PS	WSRN	28.01	26.13	26.73	27.11	26.28	26.17	26.47	26.57
PS	SRAN	26.07	27.26	26.53	26.61	26.44	26.54	26.19	26.29
PS	WSRAN	27.26	26.49	26.22	27.01	25.88	26.16	26.90	26.82
PS	SRSN	26.15	27.21	26.37	26.60	26.54	26.53	26.26	26.28
PS	WSRSN	27.31	26.60	26.24	27.16	25.85	26.07	26.96	27.00
PS	SRAN2	25.92	27.27	26.63	26.07	26.47	26.38	26.29	26.39
PS	WSRAN2	27.34	25.87	26.59	26.91	26.13	26.18	26.58	26.71

Table 7. Computational results for the dispatching rules: ARPD values III.

Measure	Indicator	Sorting							
		C	D	H	V	HIH	HIL	LOH	LOL
PMS	SUM	26.36	26.28	25.46	27.44	26.42	26.41	26.31	26.19
PMS	WSUM	24.00	28.66	25.55	27.26	26.57	26.18	26.42	26.49
PMS	ABS	26.42	26.29	26.16	26.43	26.26	26.37	26.40	26.38
PMS	WABS	26.03	26.59	26.18	26.38	26.26	26.23	26.32	26.32
PMS	SRA	26.55	26.31	26.50	26.46	26.16	26.24	26.28	26.24
PMS	WSRA	26.02	26.71	26.25	26.31	26.25	26.28	26.20	26.31
PMS	SRS	26.35	26.03	26.37	26.45	26.25	26.36	26.38	26.33
PMS	WSRS	26.00	26.60	26.17	26.20	26.32	26.33	26.23	26.25
PMS	SRN	26.21	27.42	26.74	26.40	26.51	26.58	26.48	26.49
PMS	WSRN	28.14	26.14	26.58	26.96	26.17	26.21	26.52	26.42
PMS	SRAN	26.14	27.18	26.41	26.63	26.36	26.51	26.28	26.18
PMS	WSRAN	27.31	26.46	26.34	26.99	25.98	26.08	26.89	26.93
PMS	SRSN	26.27	27.12	26.40	26.55	26.41	26.47	26.21	26.30
PMS	WSRSN	27.39	26.53	26.36	27.22	25.98	26.06	26.81	26.88
PMS	SRAN2	25.91	27.09	26.62	26.21	26.47	26.43	26.34	26.31
PMS	WSRAN2	27.30	25.94	26.57	26.67	26.13	26.24	26.67	26.76
PmS	SUM	26.47	26.30	25.39	27.33	26.51	26.43	26.34	26.27

Table 7. *Cont.*

Measure	Indicator	Sorting							
		C	**D**	**H**	**V**	**HIH**	**HIL**	**LOH**	**LOL**
PmS	WSUM	24.10	28.63	25.59	27.37	26.64	26.16	26.23	26.53
PmS	ABS	26.40	26.24	26.21	26.59	26.29	26.43	26.20	26.36
PmS	WABS	26.12	26.44	26.44	26.48	26.32	26.32	26.28	26.30
PmS	SRA	26.50	26.24	26.35	26.26	26.29	26.30	26.25	26.23
PmS	WSRA	26.03	26.76	26.27	26.39	26.27	26.26	26.34	26.29
PmS	SRS	26.33	26.20	26.32	26.41	26.24	26.30	26.19	26.26
PmS	WSRS	26.07	26.59	26.27	26.38	26.29	26.43	26.20	26.28
PmS	SRN	26.15	27.26	26.78	26.37	26.43	26.50	26.46	26.48
PmS	WSRN	28.12	26.05	26.55	26.89	26.08	26.27	26.49	26.46
PmS	SRAN	26.10	27.06	26.51	26.63	26.39	26.38	26.27	26.31
PmS	WSRAN	27.28	26.54	26.16	26.89	25.95	26.01	26.80	26.89
PmS	SRSN	26.21	27.06	26.31	26.54	26.28	26.50	26.36	26.33
PmS	WSRSN	27.35	26.61	26.37	27.08	25.97	26.07	26.79	26.83
PmS	SRAN2	25.81	27.09	26.69	26.19	26.41	26.38	26.35	26.32
PmS	WSRAN2	27.34	25.94	26.56	26.61	26.19	26.28	26.73	26.71

7.5. Computational Evaluation: NEHV Variants

In this section, we analyse the performance of the proposed NEHV variants and compare it with the best proposals in the related literature. These 896 proposed heuristics are tested in the set of instances β_2. The computational results are shown in Tables 8–10. In addition, computational results comparing the best proposal with the most promising NEH variants in the literature are shown in Table 11 (RA, NM, KK1, KK2, AD, and PR_{SKE} are the NEH heuristics that apply the initial order proposed by [74–79], respectively, while NEH represents the traditional NEH ordering the jobs in non-increasing sum of processing times). As in the previous section, the quality of the solution of the proposals is evaluated using the RPD indicator. In this case, the ARPDs of the proposals range between 2.58 and 3.30. Despite this difference, there are several variants whose ARPD is very similar, close to the best-obtained value. In this regard, the best value is found using the {PMS,ABS,D} rule as an initial sequence (with an ARPD equal to 2.58), while, e.g., the {PS,WABS,H}, {P,SRA,H}, {P,WSRS,H}, {P,WABS,HIH}, and {PS,ABS,D} found an ARPD value equal to 2.61. In this case, no statistically significant differences (with the Mann-Whitney test) were found when comparing { PMS, ABS, D} with any of the previous variants. However, the proposed NEHV({PMS,ABS,D} clearly outperforms every previous proposal in the literature. In this regard, we obtain p values equal to 0.031 and 0.004 (using the non-parametric Mann-Whitney test) comparing our best proposal with RA and with the traditional NEH algorithm, whose ARPD values are 2.73 and 2.82, respectively.Finally, in Table 4 we demonstrate the ARPD values obtained by NEHV({PMS,ABS,D}) in the small-sized instances, β_1. The global ARPD obtained is 6.73, which outperforms the best results obtained by several decoding procedures. Note that the best ARPD achievable using the PR_{SI}^{S} decoding procedure is 2.32.

Table 8. Computational results for the NEH algorithm: ARPD values I.

Measure	Indicator	Sorting							
		C	D	H	V	HIH	HIL	LOH	LOL
P	SUM	2.80	2.82	2.74	2.79	2.83	2.79	2.73	2.68
P	WSUM	2.79	3.19	2.77	2.88	2.70	2.66	2.70	2.80
P	ABS	2.86	2.73	2.68	2.80	2.75	2.78	2.73	2.82
P	WABS	2.84	2.65	2.73	2.70	2.61	2.71	2.84	2.67
P	SRA	2.85	2.74	2.61	2.79	2.73	2.72	2.74	2.77
P	WSRA	2.78	2.67	2.68	2.78	2.71	2.76	2.76	2.74
P	SRS	2.86	2.67	2.68	2.73	2.71	2.64	2.74	2.75
P	WSRS	2.80	2.72	2.61	2.81	2.73	2.76	2.75	2.70
P	SRN	2.86	3.01	2.80	2.73	2.76	2.76	2.73	2.76
P	WSRN	3.14	2.90	2.86	2.88	2.75	2.73	2.76	2.76
P	SRAN	2.78	2.91	2.76	2.80	2.79	2.86	2.63	2.64
P	WSRAN	3.01	2.71	2.77	2.81	2.72	2.67	2.83	2.80
P	SRSN	2.77	2.88	2.71	2.79	2.81	2.78	2.65	2.61
P	WSRSN	3.01	2.81	2.79	2.81	2.64	2.67	2.80	2.79
P	SRAN2	2.73	2.89	2.76	2.83	2.74	2.80	2.72	2.68
P	WSRAN2	3.02	2.75	2.85	2.86	2.69	2.76	2.75	2.77
S	SUM	2.75	2.66	2.74	2.76	2.67	2.71	2.77	2.75
S	WSUM	2.68	2.76	2.62	2.77	2.71	2.79	2.80	2.70
S	ABS	2.83	2.67	2.64	2.69	2.72	2.68	2.71	2.71
S	WABS	2.74	2.70	2.69	2.74	2.76	2.74	2.72	2.69
S	SRA	2.74	2.69	2.75	2.73	2.75	2.73	2.73	2.78
S	WSRA	2.72	2.72	2.73	2.75	2.75	2.73	2.74	2.69
S	SRS	2.72	2.74	2.72	2.78	2.74	2.74	2.70	2.69
S	WSRS	2.77	2.75	2.72	2.73	2.78	2.71	2.74	2.75
S	SRN	2.70	2.78	2.76	2.78	2.70	2.74	2.68	2.71
S	WSRN	2.76	2.73	2.68	2.77	2.72	2.70	2.75	2.72
S	SRAN	2.72	2.70	2.72	2.66	2.74	2.77	2.72	2.71
S	WSRAN	2.68	2.75	2.73	2.73	2.82	2.68	2.71	2.75
S	SRSN	2.63	2.68	2.71	2.70	2.79	2.76	2.69	2.68
S	WSRSN	2.68	2.71	2.69	2.71	2.74	2.75	2.68	2.66
S	SRAN2	2.69	2.76	2.78	2.72	2.72	2.77	2.70	2.69
S	WSRAN2	2.73	2.72	2.64	2.74	2.69	2.72	2.78	2.72
MS	SUM	2.77	2.76	2.74	2.75	2.73	2.71	2.71	2.72
MS	WSUM	2.70	2.70	2.69	2.77	2.75	2.69	2.67	2.80
MS	ABS	2.72	2.79	2.70	2.71	2.70	2.74	2.70	2.73
MS	WABS	2.72	2.70	2.78	2.70	2.71	2.71	2.71	2.72
MS	SRA	2.78	2.73	2.69	2.78	2.71	2.72	2.71	2.65
MS	WSRA	2.75	2.76	2.78	2.67	2.72	2.70	2.65	2.79
MS	SRS	2.81	2.74	2.72	2.73	2.67	2.78	2.71	2.69
MS	WSRS	2.72	2.77	2.79	2.67	2.73	2.79	2.72	2.68
MS	SRN	2.71	2.68	2.73	2.76	2.76	2.71	2.76	2.68
MS	WSRN	2.72	2.73	2.75	2.79	2.67	2.76	2.77	2.64
MS	SRAN	2.76	2.70	2.72	2.70	2.73	2.70	2.69	2.73
MS	WSRAN	2.77	2.73	2.74	2.72	2.77	2.70	2.74	2.71
MS	SRSN	2.74	2.72	2.70	2.75	2.79	2.73	2.68	2.80
MS	WSRSN	2.75	2.76	2.70	2.77	2.71	2.71	2.74	2.72
MS	SRAN2	2.77	2.70	2.74	2.75	2.75	2.69	2.72	2.65
MS	WSRAN2	2.72	2.77	2.74	2.63	2.71	2.72	2.77	2.70

Table 9. Computational results for the NEH algorithm: ARPD values II.

Measure	Indicator	Sorting							
		C	D	H	V	HIH	HIL	LOH	LOL
mS	SUM	2.73	2.65	2.72	2.75	2.76	2.79	2.80	2.73
mS	WSUM	2.67	2.70	2.67	2.72	2.78	2.65	2.76	2.69
mS	ABS	2.70	2.75	2.71	2.73	2.67	2.81	2.74	2.75
mS	WABS	2.76	2.77	2.70	2.70	2.78	2.66	2.75	2.76
mS	SRA	2.72	2.75	2.77	2.76	2.71	2.75	2.73	2.73
mS	WSRA	2.66	2.73	2.70	2.72	2.76	2.74	2.70	2.70
mS	SRS	2.71	2.69	2.77	2.77	2.77	2.78	2.71	2.75
mS	WSRS	2.75	2.72	2.67	2.78	2.76	2.67	2.73	2.69
mS	SRN	2.73	2.73	2.75	2.71	2.67	2.77	2.68	2.73
mS	WSRN	2.73	2.66	2.75	2.69	2.65	2.68	2.68	2.76
mS	SRAN	2.74	2.75	2.72	2.70	2.73	2.70	2.68	2.74
mS	WSRAN	2.71	2.72	2.75	2.74	2.70	2.77	2.69	2.78
mS	SRSN	2.67	2.75	2.80	2.76	2.76	2.76	2.71	2.74
mS	WSRSN	2.70	2.77	2.72	2.75	2.75	2.75	2.77	2.71
mS	SRAN2	2.67	2.75	2.78	2.67	2.73	2.77	2.76	2.77
mS	WSRAN2	2.69	2.74	2.67	2.75	2.74	2.79	2.69	2.73
PS	SUM	2.77	2.80	2.69	2.67	2.78	2.76	2.75	2.71
PS	WSUM	2.79	3.24	2.80	2.89	2.65	2.66	2.79	2.76
PS	ABS	2.85	2.61	2.67	2.92	2.73	2.75	2.78	2.70
PS	WABS	2.90	2.63	2.61	2.84	2.76	2.71	2.74	2.70
PS	SRA	2.88	2.74	2.62	2.78	2.68	2.70	2.76	2.70
PS	WSRA	2.82	2.65	2.65	2.80	2.71	2.81	2.67	2.74
PS	SRS	2.81	2.69	2.63	2.82	2.66	2.67	2.75	2.71
PS	WSRS	2.83	2.70	2.68	2.80	2.77	2.75	2.73	2.68
PS	SRN	2.90	2.95	2.72	2.77	2.76	2.81	2.68	2.66
PS	WSRN	3.15	2.95	2.88	2.89	2.67	2.76	2.70	2.78
PS	SRAN	2.87	2.92	2.74	2.78	2.80	2.79	2.72	2.75
PS	WSRAN	2.97	2.75	2.81	2.87	2.76	2.70	2.78	2.76
PS	SRSN	2.86	2.88	2.76	2.80	2.78	2.78	2.75	2.68
PS	WSRSN	3.00	2.82	2.77	2.78	2.68	2.72	2.78	2.77
PS	SRAN2	2.75	2.87	2.77	2.81	2.76	2.75	2.74	2.66
PS	WSRAN2	3.03	2.74	2.67	2.79	2.69	2.74	2.76	2.86

Table 10. Computational results for the NEH algorithm: ARPD values III.

Measure	Indicator	Sorting							
		C	D	H	V	HIH	HIL	LOH	LOL
PMS	SUM	2.84	2.76	2.70	2.85	2.80	2.80	2.71	2.71
PMS	WSUM	2.84	3.30	2.80	2.91	2.70	2.70	2.78	2.85
PMS	ABS	2.81	2.58	2.72	2.87	2.71	2.78	2.79	2.74
PMS	WABS	2.89	2.64	2.72	2.79	2.66	2.75	2.83	2.73
PMS	SRA	2.82	2.67	2.62	2.87	2.69	2.71	2.83	2.68
PMS	WSRA	2.76	2.64	2.68	2.82	2.74	2.75	2.72	2.71
PMS	SRS	2.73	2.66	2.70	2.79	2.68	2.72	2.76	2.66
PMS	WSRS	2.81	2.70	2.64	2.78	2.74	2.72	2.79	2.74
PMS	SRN	2.82	3.04	2.78	2.80	2.78	2.82	2.69	2.71
PMS	WSRN	3.18	2.97	2.82	2.85	2.78	2.71	2.75	2.76
PMS	SRAN	2.82	2.81	2.66	2.85	2.79	2.70	2.70	2.70
PMS	WSRAN	3.04	2.75	2.83	2.78	2.67	2.67	2.76	2.74
PMS	SRSN	2.77	2.93	2.75	2.83	2.71	2.77	2.77	2.69
PMS	WSRSN	3.10	2.82	2.79	2.82	2.66	2.71	2.79	2.82
PMS	SRAN2	2.81	2.89	2.72	2.82	2.80	2.82	2.75	2.71

Table 10. *Cont.*

Measure	Indicator	Sorting							
		C	D	H	V	HIH	HIL	LOH	LOL
PMS	WSRAN2	3.10	2.81	2.82	2.77	2.74	2.72	2.76	2.79
PmS	SUM	2.84	2.78	2.72	2.81	2.78	2.73	2.81	2.71
PmS	WSUM	2.80	3.20	2.80	2.90	2.64	2.67	2.72	2.76
PmS	ABS	2.78	2.68	2.64	2.83	2.73	2.75	2.76	2.71
PmS	WABS	2.85	2.63	2.68	2.78	2.66	2.68	2.78	2.66
PmS	SRA	2.86	2.68	2.73	2.84	2.72	2.66	2.74	2.67
PmS	WSRA	2.76	2.69	2.62	2.80	2.77	2.70	2.80	2.72
PmS	SRS	2.79	2.65	2.70	2.81	2.71	2.70	2.79	2.72
PmS	WSRS	2.87	2.66	2.62	2.81	2.73	2.71	2.82	2.77
PmS	SRN	2.82	3.00	2.75	2.77	2.81	2.78	2.69	2.73
PmS	WSRN	3.14	2.92	2.80	2.84	2.68	2.70	2.72	2.79
PmS	SRAN	2.88	2.88	2.75	2.76	2.73	2.81	2.72	2.70
PmS	WSRAN	2.98	2.74	2.81	2.81	2.71	2.67	2.80	2.82
PmS	SRSN	2.78	2.92	2.76	2.84	2.79	2.83	2.77	2.68
PmS	WSRSN	3.02	2.77	2.80	2.80	2.68	2.70	2.78	2.77
PmS	SRAN2	2.80	2.85	2.77	2.77	2.76	2.78	2.69	2.75
PmS	WSRAN2	3.11	2.72	2.82	2.77	2.66	2.68	2.83	2.77

Table 11. Computational results for the NEH algorithm: comparison with the most promising proposals in the literature.

Parameter	Value	Our	RA	NM	KK1	KK2	AD	PR_{SKE}	NEH
n	50	3.38	3.58	3.71	3.69	3.59	3.70	3.74	3.46
	100	2.71	2.90	2.86	2.82	3.06	2.62	2.86	2.95
	150	2.25	2.34	2.55	2.50	2.55	2.48	2.34	2.54
	200	2.00	2.09	2.41	2.13	2.34	2.27	2.09	2.33
m	10	2.96	3.10	3.28	3.27	3.25	3.15	3.18	3.20
	20	2.21	2.36	2.49	2.30	2.52	2.39	2.34	2.44
r'	0.1	3.19	3.12	3.39	3.39	3.41	3.47	2.88	3.38
	0.2	3.05	3.19	3.54	3.55	3.48	3.45	2.93	3.40
	0.3	2.54	2.79	3.02	2.93	2.92	2.89	2.89	2.93
	0.4	2.25	2.40	2.37	2.30	2.43	2.24	2.66	2.40
	0.5	1.89	2.14	2.10	1.75	2.17	1.78	2.41	1.98
γ	25	2.07	2.18	2.27	2.12	2.28	2.06	2.50	2.08
	50	2.54	2.82	2.90	2.81	2.90	2.73	2.80	2.86
	100	3.15	3.17	3.48	3.42	3.47	3.51	2.97	3.51
Average		2.58	2.73	2.78	2.88	2.88	2.77	2.76	2.82

7.6. Computational Evaluation: IGV

In this section, we analyse the performance of the proposed iterated greedy algorithm, IGV. To do so, we first compare it with both the MILP models and the complete enumeration procedures, on small-sized instances (β_1). Second, we compare our proposal with some of the most promising iterated greedy algorithms proposed in the related literature, on medium-large sized instances (β_2). To deal with this last issue, the following metaheuristics are re-implemented:

- IGP: The iterated greedy algorithm proposed by [93] for the mixed no-idle permutation flow shop scheduling problem and makespan minimisation.
- IGF: The iterated greedy algorithm proposed by [94] for the classical permutation flow shop scheduling problem with total tardiness minimisation.
- IGL: The iterated greedy algorithm proposed by [95] for the distributed permutation flow shop with makespan minimisation.

- IGH: The iterated greedy algorithm proposed by [96] for the distributed assembly permutation flow shop scheduling problem with sequence-dependent setup times.
- IGD: The iterated greedy algorithm proposed by [97] for the classical permutation flow shop scheduling problem with makespan minimisation.
- IGV: The proposed iterated greedy algorithm. Regarding the parameters of the algorithm, we calibrate them based on [97]. More specifically, we vary parameter d between 3 and 8, setting T to 0.4. This parameter has not been found statistically significant according to the non-parametric Kruskal-Wallis (p-value equal to 0.609). A similar result is found for parameter T when we vary it in $\{0.2, 0.3, 0.4, 0.5, 0.6\}$, setting d to 4. In this case, we obtain a p-value equal to 0.776. In Figure 4, we show the 95% confidence intervals obtained in both calibrations. Due to the fact that there are no statistically significant differences in d and T and the difference between the levels is very narrow, we use $d = 4$ and $T = 0.4$ as applied in the traditional algorithm by [84].

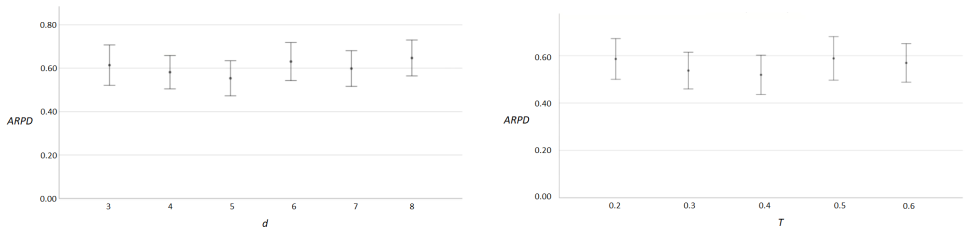

Figure 4. Calibration of d and T.

To adapt these metaheuristics to the problem under consideration, we apply the best decoding procedure (i.e., PR^S_{SI}), and initialise them with the best NEH algorithm (i.e., NEHV($\{$PMS,ABS,D$\}$)). As stopping criterion, we use $n \times m \times m \times 10/1000$, which depends of the size of the problem.

The computational results obtained by IGV on the small-sized instances are shown in Table 4, while the comparison on medium-large sized instance is shown in Table 12. Regarding the former, the ARPD obtained by IGV is 1.68 compared to 0.01 found by the best Model 1, requiring only an average computational time of 1.08 seconds, instead of 61.61 seconds required by Model 1. Note that the proposed iterated greedy algorithm explores the space of solutions of PR^S_{SI}, whose optimum is 2.32, and incorporates a local search to try to capture the global optimum of the problem. Therefore, this intensive local search helps to efficiently escape from the local optimum. The computational results also demonstrate the efficiency of the proposed mechanism, as IGV clearly outperforms the best solutions obtained by any of the proposed complete enumeration procedures. Regarding the comparison with other metaheuristics from the related literature, we again compare the performance of the metaheuristics using the RPD indicator (see Equation (37)), with $best_i$ as the best solution found in instance i by any of the iterated greedy algorithms. We also include in the comparison the best obtained NEH algorithm. The following conclusions can be derived from the results:

- IGV obtains an ARPD of 0.49, clearly outperforming all other metaheuristics. This conclusion is also statistically confirmed by a non-parametric Mann-Whitney test comparing IGV and IGD (which obtains an ARPD of 0.61). A statistically significant difference has been found with the p-value equal to 0.003.
- The constructive heuristic (NEHV($\{$PMS,ABS,D$\}$)) improves its performance when n increases compared to the metaheuristics. This fact can be explained by the hardness of the problem. When n increases, the local search included in the metaheuristics requires much more computational time, and the number of global iterations decreases compared to small instances.

- Despite the excellent performance found by some of the metaheuristics in related scheduling problems, their effectiveness in the problem under consideration clearly decreases, as e.g., IGL and IGH whose ARPD values are 1.05 and 0.84, respectively.

Table 12. Computational results for IGV: comparison with the most promising proposals in the literature.

Parmeter	Value	NEHV	IGP	IGF	IGL	IGH	IGD	IGV
n	50	6.49	0.92	1.07	1.31	1.49	0.84	0.59
	100	4.39	0.84	0.82	1.23	0.94	0.71	0.62
	150	2.88	0.59	0.53	1.00	0.60	0.57	0.48
	200	1.86	0.32	0.31	0.66	0.32	0.31	0.27
m	10	4.26	0.77	0.76	1.25	0.99	0.71	0.47
	20	3.55	0.56	0.61	0.85	0.68	0.50	0.51
r'	0.1	5.24	0.61	0.73	1.18	1.10	0.77	0.71
	0.2	4.56	0.93	0.89	1.19	1.13	0.71	0.55
	0.3	3.68	0.72	0.79	1.18	0.86	0.62	0.41
	0.4	3.11	0.64	0.55	0.86	0.59	0.57	0.39
	0.5	2.94	0.44	0.46	0.84	0.50	0.37	0.40
γ	25	3.22	0.57	0.58	0.86	0.68	0.50	0.39
	50	3.80	0.69	0.69	1.06	0.88	0.58	0.48
	100	4.70	0.74	0.79	1.23	0.95	0.74	0.60
Average		3.91	0.67	0.68	1.05	0.84	0.61	0.49

7.7. Sensitivity Analysis

In this section, we address a sensitivity analysis to study the influence of the parameters of the problem (that is, n, m, r, and γ). We analyse this influence both in the original problem under study by using the best MILP model (i.e., Model 1), and in the reduced problem obtained by applying the best decoding procedure (i.e., PR_{SI}^{S}, whose 'optimal' solution can be reached by the complete enumeration). To this end, we address two analysis of variance (ANOVA), using the previous parameters as independent variables and analysing their influence in the objective function of the problem (i.e., makespan). Table 13 reports the results with two-way interactions considering Model 1. We can observe that all the parameters significantly influence the makespan (with p-value equal to 0.000 in each case). In this regard, the makespan clearly increases when n, m, and γ increase or when r decreases. Regarding the second-order interactions, only $n * m$ is not statistically significant (with p-value equal to 0.85), i.e., the effect in the makespan of the setup time and server parameters (r and γ) changes with each other parameter. A similar trend can be found by analysing the space of solutions obtained by PR_{SI}^{S} in both the main factors and the second-order interactions. The results obtained using PR_{SI}^{S} are shown in Table 14.

Table 13. Output from ANOVA using Model 1.

Source	Type III Sum of Squares	gl	Mean Square	F	p-Value
Corrected model	36,385,933.71	107	340,055.455	77.588	0.000
Interaction	339,824,334.491	1	339,824,334.491	77,535.603	0.000
n	8,703,858.765	5	1,740,771.753	397.181	0.000
m	6,704,935.715	2	3,352,467.857	764.912	0.000
r	1,751,838.980	1	1,751,838.980	399.706	0.000
γ	15,876,334.004	2	7,938,167.002	1811.202	0.000
$n * m$	73,289.730	10	7328.973	1.672	0.085
$n * r$	83,672.854	5	16,734.571	3.818	0.002
$n * \gamma$	472,032.707	10	47,203.271	10.770	0.000
$m * r$	238,123.670	2	119,061.835	27.166	0.000
$m * \gamma$	895,119.607	4	223,779.902	51.058	0.000
$r * \gamma$	1,036,832.737	2	518,416.369	118.284	0.000
Error	1,893,376.800	432	4382.817		
Total	378,103,645.000	540			
Corrected total	38,279,310.509	539			
R Squared	0.951		Adjusted R Squared	0.938	

Table 14. Output from ANOVA using PR^S_{SI}.

Source	Type III Sum of Squares	gl	Mean Square	F	p-Value
Corrected model	32,806,374.954	107	306,601.635	71.406	0.000
Interaction	321,406,900.046	1	321,406,900.046	74,853.703	0.000
n	8,511,868.787	5	1,702,373.757	396.472	0.000
m	5,781,238.981	2	2,890,619.491	673.208	0.000
r	1,072,095.780	1	1,072,095.780	249.685	0.000
γ	14,316,475.837	2	7,158,237.919	1667.110	0.000
$n * m$	68,005.219	10	6800.522	1.584	0.109
$n * r$	82,400.787	5	16,480.157	3.838	0.002
$n * \gamma$	447,155.963	10	44,715.596	10.414	0.000
$m * r$	194,368.381	2	97,184.191	22.634	0.000
$m * \gamma$	795,412.952	4	198,853.238	46.312	0.000
$r * \gamma$	953,082.770	2	476,541.385	110.984	0.000
Error	1,854,922.000	432	4293.801		
Total	356,068,197.000	540			
Corrected total	34,661,296.954	539			
R Squared	0.951		Adjusted R Squared	0.938	

8. Conclusions

This paper tackled the permutation flow shop scheduling problem with multiple servers or human resources for the first time. The purpose of our study is twofold: first, to implement the multiple server constraint in one of the most relevant real-world-based layout; and, secondly, to analyse and propose efficient methods to solve this problem. To cover this purpose, the work developed in this paper can be stated as follows:

1. We formulated three different MILP models to exactly solve the proposed problem. The formulations were based on three different efficient families of models from the literature. The computational results identified the proposed Model 1 as the best exact formulation to solve the problem under consideration.
2. We proposed 12 different decoding procedures to explore the solutions spaces of future approximate algorithms. These procedures were integrated into a complete enumeration algorithm to analyse their efficiency. The computational results demonstrated that it is more efficient to always select the operations that can start before (i.e., PR^S family). Among them, the best result was obtained by breaking ties according to the operation with the lowest setup time (i.e., PR^S_{SI}).

3. We analyse the efficiency of dispatching rules in the problem under consideration by combining several different measures, indicators, and sorting criteria. A total of 896 different dispatching rules were proposed and compared. The best values were found by ordering the jobs according to a non-decreasing order of the processing times with or without setup times and giving more importance to its values in the first machines than to the last ones (WSUM indicator, which is based on the Johnson algorithm idea), i.e., dispatching rules {P,WSUM,C}, {PS,WSUM,C}, {PMS,WSUM,C}, and {PmS,WSUM,C}.
4. We proposed an efficient constructive heuristic to find fast solutions for the problem under consideration. The proposed heuristic is based on the classical NEH algorithm. We modified this algorithm by testing all previous dispatching rules in its first phase. The computational results demonstrate that the proposed NEHV({PMS,ABS,D}) is the best heuristic for the problem, outperforming both the classical NEH algorithm and the related NEH-based heuristics.
5. Finally, we develop a new iterated greedy metaheuristic to obtain near-optimal solutions in medium-large sized instances. The proposed IGV explores different solutions spaces by changing the decoding procedures between phases. In doing so, the metaheuristic outperforms the most promising iterated greedy algorithms from the related literature.

By this work, we try to introduce future academics with the problem, providing efficient fast heuristics, decoding procedures, and MILP models either to initialise and improve the space of solutions of future proposals (typically metaheuristics) or to embed them in more advanced approaches (as, e.g., in matheuristics). Furthermore, we provide practitioners with efficient methods (dispatching rules and fast constructive heuristics) that can be easily updated and implemented in related real manufacturing scenarios, since they are not complex to implement and are appropriate for quick decision-making.

Regarding future research lines, a number of open research issues could be conducted from this work. First, further advances should come in the development of time-consuming approaches to the problem under consideration, which can be compared with the proposed iterated greedy algorithm and initialised with the proposed methods or used some modification of the proposals in its intermediate phases (e.g., in matheuristics relaxing assignment or scheduling constraints). Second, the technological advances recently brought about by Industry 4.0 offer decision makers the ability to integrate information in real time. Preliminary studies have been conducted in this direction; however, we recommend exploring this research line to improve the efficiency of servers and machines in the problem under consideration. Finally, although the methods proposed in this paper have been developed to be easily adapted to the real manufacturing scenario, future studies could be carried out to validate and implement such methods either in a real shop or considering additional real constraints (such as, e.g., deteriorating job, green scheduling, learning effect,. . .).

Author Contributions: Conceptualization, V.F.-V. and J.M.M.-P.; methodology, V.F.-V., A.A.-C., D.G.-M., L.S.-M. and J.M.M.-P.; sofware, V.F.-V., A.A.-C. and D.G.-M.; validation, V.F.-V., A.A.-C., D.G.-M. and L.S.-M.; formal analysis, J.M.M.-P.; investigation, V.F.-V. and J.M.M.-P.; resources, V.F.-V.; data curation, V.F.-V. and L.S.-M.; visualization, L.S.-M. and J.M.M.-P.; writing—original draft preparation, V.F.-V.; writing—review and editing, V.F.-V.; supervision, J.M.M.-P.; project administration, V.F.-V. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the Junta de Andalucía (I+D+i FEDER Andalucia 2014–2020) under the project "DEMAND" with reference P18-FR-1149, and by the Ministerio de Ciencia e Innovación (Spain), under the project "ASSORT" with reference PID2019-108756RB-I00.

Data Availability Statement: Computational results are included as on-line material.

Acknowledgments: The authors wish to thank the referees for their comments on the earlier versions of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PFSP	Permutation Flow shop Scheduling Problem
PFSMS	Permutation Flow shop Scheduling problem with Multiple Servers
MILP	Mixed Integer Linear Programming model
NEHV	Proposed constructive heuristics based on the classical NEH algorithm

References

1. Fernandez-Viagas, V.; Ruiz, R.; Framinan, J.M. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *Eur. J. Oper. Res.* **2017**, *257*, 707–721. [CrossRef]
2. Kempf, K.; Uzsoy, R.; Wang, C.S. Scheduling a Single Batch Processing Machine with Secondary Resource Constraints. *J. Manuf. Syst.* **1998**, *17*, 37–51. [CrossRef]
3. Elidrissi, A.; Benmansour, R.; Benbrahim, M.; Duvivier, D. Mathematical formulations for the parallel machine scheduling problem with a single server. *Int. J. Prod. Res.* **2021**, *59*, 6166–6184. [CrossRef]
4. Koulamas, C. Scheduling two parallel semiautomatic machines to minimize machine interference. *Comput. Oper. Res.* **1996**, *23*, 945–956. [CrossRef]
5. Li, X.; Baki, M.; Aneja, Y. Flow shop scheduling to minimize the total completion time with a permanently present operator: Models and ant colony optimization metaheuristic. *Comput. Oper. Res.* **2011**, *38*, 152–164. [CrossRef]
6. Kim, M.Y.; Lee, Y. MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. *Comput. Oper. Res.* **2012**, *39*, 2457–2468. [CrossRef]
7. Torjai, L.; Kruzslicz, F. Mixed integer programming formulations for the Biomass Truck Scheduling problem. *Cent. Eur. J. Oper. Res.* **2016**, *24*, 731–745. [CrossRef]
8. Bish, E. A multiple-crane-constrained scheduling problem in a container terminal. *Eur. J. Oper. Res.* **2003**, *144*, 83–107. [CrossRef]
9. Costa, A.; Fernandez-Viagas, V.; Framinan, J. Solving the hybrid flow shop scheduling problem with limited human resource constraint. *Comput. Ind. Eng.* **2020**, *146*, 106545. [CrossRef]
10. Hall, N.; Potts, C.; Sriskandarajah, C. Parallel machine scheduling with a common server. *Discret. Appl. Math.* **2000**, *102*, 223–243. [CrossRef]
11. Seeanner, F.; Meyr, H. Multi-stage simultaneous lot-sizing and scheduling for flow line production. *OR Spectr.* **2013**, *35*, 33–73. [CrossRef]
12. Tempelmeier, H.; Copil, K. Capacitated lot sizing with parallel machines, sequence-dependent setups, and a common setup operator. *OR Spectr.* **2016**, *38*, 819–847. [CrossRef]
13. Miltenburg, J. U-shaped production lines: A review of theory and practice. *Int. J. Prod. Econ.* **2001**, *70*, 201–214. [CrossRef]
14. Fernandez-Viagas, V.; Costa, A.; Framinan, J. Hybrid flow shop with multiple servers: A computational evaluation and efficient divide-and-conquer heuristics. *Expert Syst. Appl.* **2020**, *153*, 113462. [CrossRef]
15. Yepes-Borrero, J.; Villa, F.; Perea, F.; Caballero-Villalobos, J. GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Syst. Appl.* **2020**, *141*, 112959. [CrossRef]
16. Yepes-Borrero, J.; Perea, F.; Ruiz, R.; Villa, F. Bi-objective parallel machine scheduling with additional resources during setups. *Eur. J. Oper. Res.* **2021**, *292*, 443–455. [CrossRef]
17. Sivrikaya-Serifoglu, F.; Ulusoy, G. Parallel machine scheduling with earliness and tardiness penalties. *Comput. Oper. Res.* **1999**, *26*, 773–787. [CrossRef]
18. Werner, F.; Kravchenko, S. Scheduling with multiple servers. *Autom. Remote Control* **2010**, *71*, 2109–2121. [CrossRef]
19. Ruiz, R.; Andres-Romano, C. Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *Int. J. Adv. Manuf. Technol.* **2011**, *57*, 777–794. [CrossRef]
20. Bektur, G.; Sarac, T. A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. *Comput. Oper. Res.* **2019**, *103*, 46–63. [CrossRef]
21. Framinan, J.; Leisten, R.; Ruiz, R. *Manufacturing Scheduling Systems: An Integrated View on Models, Methods, and Tools*; Springer: Berlin/Heidelberg, Germany, 2014.
22. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Ann. Discret. Math.* **1979**, *5*, 287–326.
23. Rinnooy Kan, A.H.G. *Machine Scheduling Problems: Classification, Complexity and Computations*; Martinus Nijhoff: The Hague, The Netherlands, 1976.
24. Nawaz, M.; Enscore, J.E.E.; Ham, I. A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. *OMEGA Int. J. Manag. Sci.* **1983**, *11*, 91–95. [CrossRef]
25. Fernandez-Viagas, V.; Framinan, J. Exploring the benefits of scheduling with advanced and real-time information integration in Industry 4.0: A computational study. *J. Ind. Inf. Integr.* **2022**, *27*, 100281. [CrossRef]
26. Qu, S.; Li, Y.; Ji, Y. The mixed integer robust maximum expert consensus models for large-scale GDM under uncertainty circumstances. *Appl. Soft Comput.* **2021**, *107*, 107369. [CrossRef]

27. Ji, Y.; Li, H.; Zhang, H. Risk-Averse Two-Stage Stochastic Minimum Cost Consensus Models with Asymmetric Adjustment Cost. *Group Decis. Negot.* **2022**, *31*, 261–291. [CrossRef]
28. Larsen, R.; Pranzo, M. A framework for dynamic rescheduling problems. *Int. J. Prod. Res.* **2019**, *57*, 16–33. [CrossRef]
29. Framinan, J.; Perez-Gonzalez, P. On heuristic solutions for the stochastic flowshop scheduling problem. *Eur. J. Oper. Res.* **2015**, *246*, 413–420. [CrossRef]
30. Alawad, N.; Abed-alguni, B. Discrete Jaya with refraction learning and three mutation methods for the permutation flow shop scheduling problem. *J. Supercomput.* **2022**, *78*, 3517–3538. [CrossRef]
31. Fathollahi-Fard, A.; Woodward, L.; Akhrif, O. Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept. *J. Ind. Inf. Integr.* **2021**, *24*, 100233. [CrossRef]
32. Fernandez-Viagas, V.; Talens, C.; Framinan, J. Assembly flowshop scheduling problem: Speed-up procedure and computational evaluation. *Eur. J. Oper. Res.* **2022**, *299*, 869–882. [CrossRef]
33. Fernandez-Viagas, V.; Prata, B.; Framinan, J. A critical-path based iterated local search for the green permutation flowshop problem. *Comput. Ind. Eng.* **2022**, *169*, 108276. [CrossRef]
34. Lee, J.H.; Kim, H.J. Reinforcement learning for robotic flow shop scheduling with processing time variations. *Int. J. Prod. Res.* **2022**, *60*, 2346–2368.
35. Morais, M.; Ribeiro, M.; da Silva, R.; Mariani, V.; Coelho, L. Discrete differential evolution metaheuristics for permutation flow shop scheduling problems. *Comput. Ind. Eng.* **2022**, *166*, 107956. [CrossRef]
36. Abu Doush, I.; Al-Betar, M.; Awadallah, M.; Alyasseri, Z.; Makhadmeh, S.; El-Abd, M. Island neighboring heuristics harmony search algorithm for flow shop scheduling with blocking. *Swarm Evol. Comput.* **2022**, *74*, 101127. [CrossRef]
37. Cheng, C.Y.; Pourhejazy, P.; Ying, K.C.; Liao, Y.H. New benchmark algorithms for No-wait Flowshop Group Scheduling Problem with Sequence-Dependent Setup Times. *Appl. Soft Comput.* **2021**, *111*, 107705. [CrossRef]
38. Rifai, A.; Mara, S.; Sudiarso, A. Multi-objective distributed reentrant permutation flow shop scheduling with sequence-dependent setup time. *Expert Syst. Appl.* **2021**, *183*, 115339. [CrossRef]
39. Ribas, I.; Companys, R.; Tort-Martorell, X. An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times. *Expert Syst. Appl.* **2021**, *184*, 115535. [CrossRef]
40. Meng, L.; Gao, K.; Ren, Y.; Zhang, B.; Sang, H.; Chaoyong, Z. Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. *Swarm Evol. Comput.* **2022**, *71*, 101058. [CrossRef]
41. Sharma, M.; Sharma, M.; Sharma, S. Desert sparrow optimization algorithm for the bicriteria flow shop scheduling problem with sequence-independent setup time. *Oper. Res.* **2022**, *22*, 4353–4396. [CrossRef]
42. Miyata, H.; Nagano, M. An iterated greedy algorithm for distributed blocking flow shop with setup times and maintenance operations to minimize makespan. *Comput. Ind. Eng.* **2022**, *171*, 108366. [CrossRef]
43. Zhang, Z.Q.; Qian, B.; Hu, R.; Jin, H.P.; Wang, L.; Yang, J.B. A matrix-cube-based estimation of distribution algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. *Expert Syst. Appl.* **2022**, *205*, 117602. [CrossRef]
44. Qiao, Y.; Wu, N.; He, Y.; Li, Z.; Chen, T. Adaptive genetic algorithm for two-stage hybrid flow-shop scheduling with sequence-independent setup time and no-interruption requirement. *Expert Syst. Appl.* **2022**, *208*, 118068. [CrossRef]
45. González-Neira, E.; Montoya-Torres, J.; Barrera, D. Flow-shop scheduling problem under uncertainties: Review and trends. *Int. J. Ind. Eng. Comput.* **2017**, *8*, 399–426. [CrossRef]
46. Miyata, H.; Nagano, M. The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Syst. Appl.* **2019**, *137*, 130–156. [CrossRef]
47. Neufeld, J.; Gupta, J.; Buscher, U. A comprehensive review of flowshop group scheduling literature. *Comput. Oper. Res.* **2016**, *70*, 56–74. [CrossRef]
48. Rossit, D.; Tohmé, F.; Frutos, M. The Non-Permutation Flow-Shop scheduling problem: A literature review. *Omega* **2018**, *77*, 143–153. [CrossRef]
49. Yenisey, M.; Yagmahan, B. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega* **2014**, *45*, 119–135. [CrossRef]
50. Kravchenko, S.; Werner, F. Parallel machine scheduling problems with a single server. *Math. Comput. Model.* **1997**, *26*, 1–11. [CrossRef]
51. Huang, S.; Cai, L.; Zhang, X. Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Comput. Ind. Eng.* **2010**, *58*, 165–174. [CrossRef]
52. Lim, A.; Rodrigues, B.; Wang, C. Two-machine flow shop problems with a single server. *J. Sched.* **2006**, *9*, 515–543. [CrossRef]
53. Ling, S.; Xue-guang, C. On a Two-machine Flow-shop Scheduling Problem with a Single Server and Unit Processing Times. *J. Appl. Math. Bioinform.* **2011**, *1*, 33–38.
54. Brucker, P.; Knust, S.; Wang, G. Complexity results for flow-shop problems with a single server. *Eur. J. Oper. Res.* **2005**, *165*, 398–407. [CrossRef]
55. Su, L.H.; Lee, Y.Y. The two-machine flowshop no-wait scheduling problem with a single server to minimize the total completion time. *Comput. Oper. Res.* **2008**, *35*, 2952–2963. [CrossRef]
56. Cheng, T.; Wang, G.; Sriskandarajah, C. One-operator-two-machine flowshop scheduling with setup and dismounting times. *Comput. Oper. Res.* **1999**, *26*, 715–730. [CrossRef]

57. Oulamara, A.; Rebaine, D.; Serairi, M. Scheduling the two-machine open shop problem under resource constraints for setting the jobs. *Ann. Oper. Res.* **2013**, *211*, 333–356. [CrossRef]
58. Samarghandi, H. A no-wait flow shop system with sequence dependent setup times and server constraints. *IFAC-PapersOnLine* **2015**, *28*, 1604–1609. [CrossRef]
59. Samarghandi, H. Studying the effect of server side-constraints on the makespan of the no-wait flow-shop problem with sequence-dependent set-up times. *Int. J. Prod. Res.* **2015**, *53*, 2652–2673. [CrossRef]
60. Stafford Jr., E.; Tseng, F.; Gupta, J. Comparative evaluation of MILP flowshop models. *J. Oper. Res. Soc.* **2005**, *56*, 88–101. [CrossRef]
61. Naderi, B.; Gohari, S.; Yazdani, M. Hybrid flexible flowshop problems: Models and solution methods. *Appl. Math. Model.* **2014**, *38*, 5767–5780. [CrossRef]
62. Fernandez-Viagas, V.; Perez-Gonzalez, P.; Framinan, J. Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective. *Comput. Oper. Res.* **2019**, *109*, 77–88. [CrossRef]
63. Framinan, J.M.; Leisten, R.; Rajendran, C. Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *Int. J. Prod. Res.* **2003**, *41*, 121–148. [CrossRef]
64. Conway, R.W. Priority Dispatching and Work-in-Process Inventory in a Job Shop. *J. Ind. Eng.* **1965**, *16*, 123–130.
65. Graham, R.L. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* **1969**, *17*, 416–429. [CrossRef]
66. Pinedo, M. *Scheduling: Theory, Algorithms and Systems*; Springer: Berlin/Heidelberg, Germany, 2012.
67. Rajendran, C. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *Int. J. Prod. Econ.* **1993**, *29*, 65–73. [CrossRef]
68. Rios-Mercado, R.; Bard, J. Heuristics for the flow line problem with setup costs. *Eur. J. Oper. Res.* **1998**, *110*, 76–98. [CrossRef]
69. Fernandez-Viagas, V.; Molina-Pariente, J.M.; Framinan, J.M. New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Syst. Appl.* **2018**, *114*, 345–356. [CrossRef]
70. Perez-Gonzalez, P.; Fernandez-Viagas, V.; Framinan, J. Permutation flowshop scheduling with periodic maintenance and makespan objective. *Comput. Ind. Eng.* **2020**, *143*, 106369. [CrossRef]
71. Ruiz, R.; Pan, Q.K.; Naderi, B. Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega* **2019**, *83*, 213–222. [CrossRef]
72. Rossi, F.; Nagano, M. Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times. *Comput. Ind. Eng.* **2021**, *157*, 107337. [CrossRef]
73. Li, Y.Z.; Pan, Q.K.; Ruiz, R.; Sang, H.Y. A referenced iterated greedy algorithm for the distributed assembly mixed no-idle permutation flowshop scheduling problem with the total tardiness criterion. *Knowl.-Based Syst.* **2022**, *239*, 108036. [CrossRef]
74. Nagano, M.; Moccellin, J. A high quality solution constructive heuristic for flow shop sequencing. *J. Oper. Res. Soc.* **2002**, *53*, 1374–1379. [CrossRef]
75. Liu, W.; Jin, Y.; Price, M. A new improved NEH heuristic for permutation flowshop scheduling problems. *Int. J. Prod. Econ.* **2017**, *193*, 21–30. [CrossRef]
76. Dong, X.; Huang, H.; Chen, P. An improved NEH-based heuristic for the permutation flowshop problem. *Comput. Oper. Res.* **2008**, *35*, 3962–3968. [CrossRef]
77. Kalczynski, P.J.; Kamburowski, J. An improved NEH heuristic to minimize makespan in permutation flow shops. *Comput. Oper. Res.* **2008**, *35*, 3001–3008. [CrossRef]
78. Kalczynski, P.J.; Kamburowski, J. An empirical analysis of the optimality rate of flow shop heuristics. *Eur. J. Oper. Res.* **2009**, *198*, 93–101. [CrossRef]
79. Ribas, I.; Companys, R.; Tort-Martorell, X. Comparing three-step heuristics for the permutation flow shop problem. *Comput. Oper. Res.* **2010**, *37*, 2062–2070. [CrossRef]
80. Fernandez-Viagas, V.; Framinan, J.M. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Comput. Oper. Res.* **2014**, *45*, 60–67. [CrossRef]
81. Ying, K.C.; Lin, S.W. A high-performing constructive heuristic for minimizing makespan in permutation flowshops. *J. Ind. Prod. Eng.* **2013**, *30*, 355–362. [CrossRef]
82. Rad, S.F.; Ruiz, R.; Boroojerdian, N. New high performing heuristics for minimizing makespan in permutation flowshops. *OMEGA Int. J. Manag. Sci.* **2009**, *37*, 331–345. [CrossRef]
83. Kalczynski, P.J.; Kamburowski, J. On the NEH heuristic for minimizing the makespan in permutation flow shops. *OMEGA Int. J. Manag. Sci.* **2007**, *35*, 53–60. [CrossRef]
84. Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **2007**, *177*, 2033–2049. [CrossRef]
85. Huang, Y.Y.; Pan, Q.K.; Huang, J.P.; Suganthan, P.; Gao, L. An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem. *Comput. Ind. Eng.* **2021**, *152*, 107021. [CrossRef]
86. Mao, J.Y.; Pan, Q.K.; Miao, Z.H.; Gao, L. An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. *Expert Syst. Appl.* **2021**, *169*, 114495. [CrossRef]

87. Zou, W.Q.; Pan, Q.K.; Tasgetiren, M. An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop. *Appl. Soft Comput.* **2021**, *99*, 106945. [CrossRef]
88. Missaoui, A.; Ruiz, R. A parameter-Less iterated greedy method for the hybrid flowshop scheduling problem with setup times and due date windows. *Eur. J. Oper. Res.* **2022**, *303*, 99–113. [CrossRef]
89. Lu, C.; Liu, Q.; Zhang, B.; Yin, L. A Pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop. *Expert Syst. Appl.* **2022**, *204*, 117555. [CrossRef]
90. Wang, Z.Y.; Pan, Q.K.; Gao, L.; Wang, Y.L. An effective two-stage iterated greedy algorithm to minimize total tardiness for the distributed flowshop group scheduling problem. *Swarm Evol. Comput.* **2022**, *74*, 101143. [CrossRef]
91. Shao, Z.; Pi, D.; Shao, W. A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. *Swarm Evol. Comput.* **2018**, *40*, 53–75. [CrossRef]
92. Vallada, E.; Ruiz, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *Eur. J. Oper. Res.* **2011**, *211*, 612–622. [CrossRef]
93. Pan, Q.K.; Ruiz, R. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. *Omega* **2014**, *44*, 41–50. [CrossRef]
94. Fernandez-Viagas, V.; Valente, J.; Framinan, J. Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. *Expert Syst. Appl.* **2018**, *94*, 58–69. [CrossRef]
95. Lin, S.W.; Ying, K.C.; Huang, C.Y. Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. *Int. J. Prod. Res.* **2013**, *51*, 5029–5038. [CrossRef]
96. Hatami, S.; Ruiz, R.; Andrés-Romano, C. Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *Int. J. Prod. Econ.* **2015**, *169*, 76–88. [CrossRef]
97. Dubois-Lacoste, J.; Pagnozzi, F.; Stützle, T. An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem. *Comput. Oper. Res.* **2017**, *81*, 160–166. [CrossRef]