



**HAL**  
open science

## Flowshop scheduling problem with parallel semi-lines and final synchronization operation

Irce F.G. Guimarães, Yassine Ouazene, Mauricio C de Souza, Farouk Yalaoui

### ► To cite this version:

Irce F.G. Guimarães, Yassine Ouazene, Mauricio C de Souza, Farouk Yalaoui. Flowshop scheduling problem with parallel semi-lines and final synchronization operation. *Computers and Operations Research*, 2019, 108, pp.121-133. 10.1016/j.cor.2019.04.011 . hal-02311165

**HAL Id: hal-02311165**

**<https://utt.hal.science/hal-02311165>**

Submitted on 22 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

## Flowshop scheduling problem with parallel semi-lines and final synchronization operation

Irce F.G. Guimarães<sup>a</sup>, Yassine Ouazene<sup>b,\*</sup>, Mauricio C. de Souza<sup>a</sup>, Farouk Yalaoui<sup>b</sup>

<sup>a</sup>*DEP, Federal University of Minas Gerais  
Av. Antônio Carlos 6627, Belo Horizonte - MG  
31270 - 901, Brazil*

<sup>b</sup>*Laboratoire d'Optimisation des Systèmes Industriels  
Université de Technologie de Troyes, 12 rue Marie Curie  
CS 42060, 10004 Troyes, France*

---

### Abstract

This paper deals with a particular variant of the flowshop scheduling problem motivated by a real case configuration issued from an electro-electronic material industry. The shop floor environment is composed of two parallel semi-lines and a final synchronization operation. The jobs must follow the same technological order through the machines on each parallel semi-line. However, the operations on each semi-line are independent. The final synchronization operation, operated by a dedicated machine, can only start when the job is finished on both semi-lines. The objective is to determine a schedule that minimizes the makespan for a given set of jobs. Since this problem class is NP-hard in the strong sense, constructive heuristic procedures and metaheuristics methods are introduced to achieve optimal or near-optimal solutions. The performances of the proposed GRASP and the Simulated Annealing algorithms are evaluated and compared with the adaptation of two well-known heuristics. Computational experiments show that the proposed metaheuristics provide very good results in low computational times.

*Keywords:* flowshop scheduling; synchronization operation; heuristic and metaheuristic methods; GRASP algorithm; simulated annealing algorithm

---

\*Corresponding author  
Email address: [yassine.ouazene@utt.fr](mailto:yassine.ouazene@utt.fr) (Yassine Ouazene)

## 1. Introduction

In this paper, we consider a variant of the permutation flowshop scheduling problem. The classical permutation flowshop consists of  $n$  jobs to be processed among a set of  $m$  machines arranged in series. The jobs must follow the same technological order through the machines, and each job has a specific processing time in each machine. The goal is to determine, among all the possible  $n!$  sequences, one that optimizes a certain performance measure. The most commonly used is the minimization of the total completion time (makespan).

We analyze a variant of the permutation flowshop problem motivated by a practical application found in the welding sector of an electro-electronics industry. The shop floor environment is composed of two parallel semi-lines and a final synchronization operation. Each semi-line produces one of the halves of a job. These halves are assembled in the final synchronization operation. The halves of the respective jobs must be processed in the same order in each semi-line, which must be followed in the synchronization operation as well. Figure 1 shows a scheme of the studied environment. In this figure, the semi-lines process first the halves of job 2, then those of job 1, and finally those of job 3. This same order is followed in the synchronization operation. An operation in a machine of a semi-line does not need to start at the same time as an operation in the machine of the other semi-line. However, the final synchronization operation of a job can only start when its halves are completed in both semi-lines. The objective is to minimize the makespan.

We consider the cases where the semi-lines have (i) the same number and (ii) different number of machines. A practical example of a process with different number of machines in each semi-line is found in the production of circuit breakers, where one semi-line with two machines processes the contact blade, and the other semi-line with three machines processes the bimetal. When these two semi-products are completed, they are assembled in the synchronization

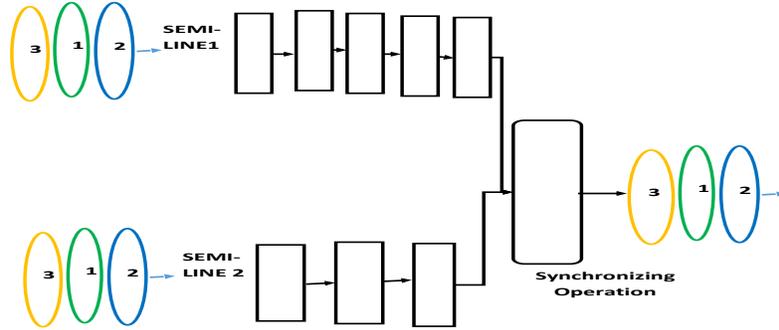


Figure 1: System under study.

operation.

30 *1.1. Illustrative examples*

Figure 2 shows the Gantt chart of an optimal sequence for a small illustrative example with 2 jobs, yellow and green, 2 machines in each semi-line  $l$ ,  $l = 1, 2$ , and the synchronization operation. We denote by  $i_k^l$ , the machine  $k = 1, 2$  of semi-line  $l$ , and by  $i_s$  the synchronization operation. Observe that a half of a job can start in a machine  $i_k^1$  independently of when the other half starts in a machine  $i_k^2$ . However, even when a half of a job is completed at semi-line  $l = 1$  it must wait for the other half to be completed at semi-line  $l = 2$  before the synchronization operation can start.

Let  $C^*$  be the optimum makespan of a given instance of the variant of the permutation flowshop under study, and let  $C^l$  be the optimum makespan of an instance of the classical permutation flowshop considering the machines of semi-line  $l$  along with the synchronization operation. In general we have that  $C^* > C^l$ ,  $l = 1, 2$ . Let us consider a small example with 3 jobs where each semi-

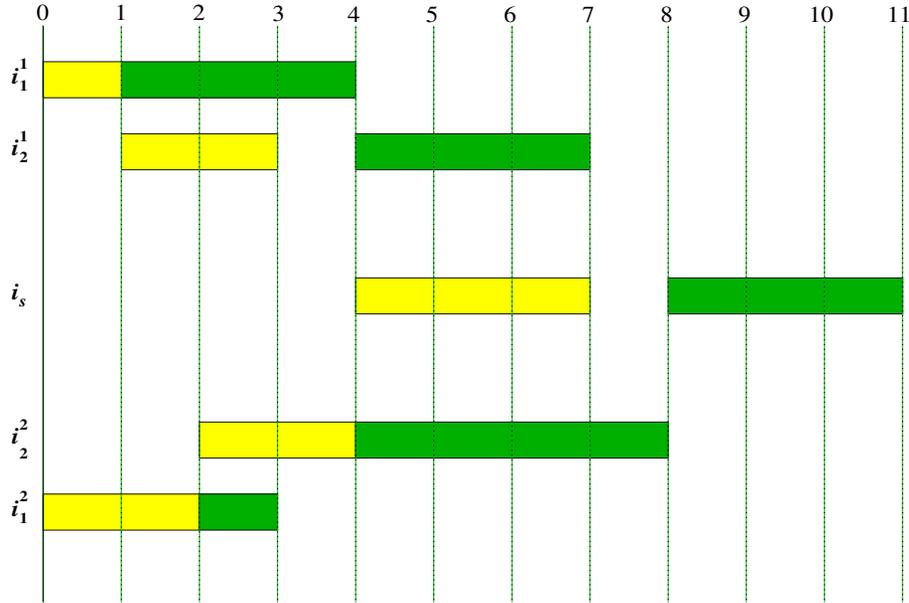


Figure 2: Gantt chart of a small example.

line  $l$  has 2 machines, plus the synchronization operation. Table 1 shows the  
 45 processing times of each job in each machine  $i_k^l$ ,  $k = 1, 2$ , and in the synchroniza-  
 tion machine  $i_s$ . The optimal makespan of the permutation flowshop instance  
 $i_1^1, i_2^1, i_s$  is 94, which can be obtained with the sequence  $j_1, j_3, j_2$ . The makespan  
 of this sequence in the problem with two semi-lines and the synchronization  
 operation is 96. Alternatively, the optimal makespan of the permutation flow-  
 50 shop instance  $i_1^2, i_2^2, i_s$  is 93, with the sequence  $j_2, j_1, j_3$ , and the makespan in  
 the problem with two semi-lines and the synchronization operation is 109. But  
 the optimal makespan of the permutation flowshop with the two semi-lines and  
 the synchronization operation is 95, which can be obtained with the sequence  
 $j_3, j_2, j_1$  (not optimal for any of the two instances given by the semi-lines con-  
 55 sidered independently).

This is also observed when the semi-lines have different number of machines.  
 Indeed, let us just consider the instance obtained by removing the first machine  
 $i_1^1$  from semi-line  $l = 1$  in Table 1, i.e., an instance with 1 machine in semi-line

	semi-line $l = 1$		semi-line $l = 2$		$i_s$
	$i_1^1$	$i_2^1$	$i_1^2$	$i_2^2$	
$j_1$	10	18	14	16	22
$j_2$	6	37	18	9	25
$j_3$	5	4	21	8	19

Table 1: An example with the same number of machines in each semi-line.

$l = 1$  and 2 machines in semi-line  $l = 2$ , plus the synchronization operation.  
60 The optimal makespan is still 95, which can be obtained with the same sequence  $j_3, j_2, j_1$ .

### 1.2. Related literature

Many approaches have been proposed for the flowshop problem since Johnson [1] presented the resolution for the flowshop considering two machines.  
65 Gupta and Stafford [2] provide a historical perspective of the research in the flowshop problem and its variants. The well-known algorithm of Johnson [1] finds in polynomial time an optimal sequence for a set of  $n$  jobs to be processed in  $m = 2$  machines. A major difficulty is to find an optimal solution when the number of machines is greater than two, since this problem is known to be strongly  
70 NP-Hard (Garey et al. [3]). Thus, studies have been developed in the literature of flowshop scheduling considering exact and heuristic techniques as well.

#### 1.2.1. Permutation flowshop

Tseng et al. [4] report a thorough empirical analysis to assess the effectiveness of mixed-integer linear programming (MIP) formulations for the permutation  
75 flowshop. We briefly give some more recent examples of the use of MIP models to address flowshop problems in the literature. Frach et al. [5] also present a MIP to solve flowshop problems with a limited number of intermediate buffers. Naderi et al. [6] propose a MIP to minimize the makespan, and the total tardiness in a flowshop environment. Ronconi and Bergin [7] address by MIP the problem

80 of minimizing the total earliness and tardiness of jobs for the flowshop problem with unlimited and also with zero buffer. Hnaien et al. [8] propose two MIP models for the two-machine flowshop scheduling problem with unavailability constraint in the first machine in order to minimize the makespan. The authors propose a branch and bound algorithm based on new lower bounds and heuristics  
85 that performs better than the two MIP models.

Heuristics have been proposed in the literature to obtain good solutions in a short computational time, see, for instance, Mainieri and Ronconi [9], Nawaz et al. [10], Rad et al. [11], and Widmer and Hertz [12]. Johnson's algorithm was adapted by Allahverdi et al. [13] to minimize the total completion time in two-  
90 machines flowshop with limited and random processing times. Pan et al. [14] tackle the flowshop problem with zero buffer. In that study, the heuristic of Nawaz et al. [10] exploit specific characteristics of the problem to find good solutions with little computational effort. Allaoui and Artiba [15] aim to minimize the makespan in a two stage hybrid flowshop with a single machine in the first  
95 stage and  $m$  machines in the second stage. Fernandez-Viagas and Framinan [16] propose efficient tie-breaking mechanisms to be used in the heuristic of Nawaz et al. [10] when dealing with total tardiness.

Metaheuristic approaches have also been proposed in the literature to solve large instances in reasonable computational time. Simulated annealing has been  
100 applied by Low et al. [17] and by Nearchou [18] to minimize the makespan in the flowshop problem, and by Mirsanei et al. [19] and by Santosa and Rofiq [20] to the hybrid flowshop problem with  $m$ -machines in each stage. GRASP has been applied by Prabhakaran et al. [21]. Shahul Hamid Khan et al. [22] address with GRASP a bicriteria flowshop where the objective is to minimize the  
105 weighted sum of makespan and maximum tardiness. Their algorithm was able to outperform a simulated annealing previously proposed by Chakravarthy and Rajendran [23] for the same problem. On the other hand, in the computational experiments conducted by Sivasankaran et al. [24] simulated annealing outperformed GRASP for a single-stage scheduling problem.

110 *1.2.2. Distributed permutation flowshop*

In 2010, Naderi and Ruiz [25] introduced a new generalization of the regular permutation flowshop scheduling problem referred to as the distributed permutation flowshop scheduling problem or (DPFSP). This new version assumes that there are a total of  $F$  identical factories or shops, each one composed  
115 of  $m$  machines disposed in series. The available jobs have to be distributed among the different factories or shops and then a processing sequence has to be derived for the jobs assigned to each factory. The author characterized the DPFSP and proposed different mixed integer linear programming models to solve the problem. They also discussed several heuristics based on dispatching  
120 rules, effective constructive heuristics and variable neighborhood descent methods. This problem has attracted considerable interest over the last few years. Many authors have presented new optimization methods to solve this problem. For example, Bargaoui et al. [26] used an artificial chemical reaction metaheuristic which objective is to minimize the maximum completion time to  
125 solve the distributed permutation flowshop scheduling problem with makespan criterion. In the proposed metaheuristic, the NEH heuristic was adapted to generate the initial population of molecules. Furthermore, a One-Point crossover and a greedy strategy were embedded in the algorithm in order to ameliorate the solution quality. Ruiz et al. [27] proposed Iterated Greedy algorithms based  
130 on improved initialization, construction and destruction procedures, along with a local search procedure.

Hatemi et al. [28] presented for the first time a combination of DPFSP and the Assembly Flow Shop Scheduling Problem referred to distributed assembly permutation flowshop scheduling problem (DAPFSP). DAPFSP contains two  
135 phase, production and assembly. At the first phase, it is to produce manufacture parts, just like in the regular DPFSP. The second phase is to assemble parts to make products in one assembly factory with only an assembly machine. The authors proposed two simple constructive algorithms to solve the problem. They also provided two sets of instances to test the model and algorithms.

140 The literature about the distributed permutation flowshop Problem is relatively small. Li et al. [29] addressed the same problem. They proposed four versions of genetic algorithm by combining the classical genetic algorithm with enhanced crossover and several local searches to solve the problem. Recently, Hatemi et al. [30] addressed the same problem with the additional consid-  
145 eration of sequence dependent setup times at both production and assembly stages. They proposed two simple heuristics and two metaheuristics to solve it. The performance of their proposed methods was compared through extensive computational and statistical experiments study. A hybrid biogeography-based optimization algorithm that integrates several novel heuristics is proposed by  
150 Lin and Zhang[31] to solve the DAPFSP with the objective of minimizing the makespan. The performance of their approach was evaluated based on two sets of benchmark instances issued from the literature. Compared to the best-known results, new best solutions for 71 small-sized instances and 91 large-sized instances were found.

155 To the best of our knowledge, Sang et al. [32] addressed in 2019 the first paper attempt to minimize total flowtime for the distributed assembly permutation flowshop scheduling problem. They proposed three variants of the discrete invasive weed optimization approach to solve the problem. To test the proposed algorithms, they carried out a comprehensive experiment based on 810 instances  
160 issued from literature. Numerical results and statistical analysis show that the presented algorithms perform substantially better than the other algorithms in for solving the DAPFSP with the total flowtime criterion.

For a complete review of the literature of the different variants of deterministic assembly scheduling problems, reader can refer the excellent work of  
165 Framinan et al. [33]. The authors reviewed a large number of papers in order to provide a comprehensive overview on scheduling with assembly operations. They proposed a unified notation for assembly scheduling models that encompass all concurrent-type scheduling problems. Using this notation, the existing contributions are reviewed and classified into a single framework, so a comprehensive, unified picture of the field is addressed.  
170

### 1.3. Structure of the paper

Based on the studies aforementioned, we propose, for the variant of the flowshop analyzed in this paper: a MIP model; heuristics based on the algorithms proposed by Johnson [1] and by Nawaz et al. [10]; and GRASP and simulated annealing metaheuristics. We report computational experiments on random instances generated according to procedures used to generate benchmark instances found in the literature for the permutation flowshop. This work is an extension of the preliminary results in [34]. The paper is organized as follows. In Section 2 the mathematical model is presented. The proposed heuristics and metaheuristics are presented in Section 3 and Section 4, respectively. Computational experiments are reported in Section 5, and concluding remarks drawn in Section 6.

## 2. Mathematical formulation

We propose a MIP formulation based on the previous works developed by Wagner [35] and by Stafford [36]. According to the study conducted by Tseng et al. [4], such kind of formulation is the best for permutation flowshop. Let  $n$  be the number of jobs, and  $m_l$  be the number of machines in semi-line  $l = 1, 2$  (before the synchronizing operation). In our model, a unique permutation has to be chosen for two flowshop problems with  $m_l + 1$  machines each subject to the additional constraint that the completion times of each job on machines  $m_1 + 1$  and  $m_2 + 1$  are the same. Let  $p_{ki}^l$  be the processing time of job  $i$  on machine  $k$  of the problem  $l$  ( $p_{(m_1+1)i} = p_{(m_2+1)i}$ ). The model can be generalized for an arbitrary number  $L$  of semi-lines. Thus, assuming, without loss of generality,  $l = 1, 2$  flowshop problems with  $m_l + 1$  machines each, the variables are defined as follows:

- $z_{ij}$  binary variable specifies if job  $i$  is assigned to the  $j^{\text{th}}$  position of the permutation (common two both  $l = 1, 2$  flowshop problems);
- $x_{jk}^l$  idle time on machine  $k$  of the problem  $l$  before the job starts in the  $j^{\text{th}}$  position of the permutation;
- $y_{jk}^l$  idle time of job in the  $j^{\text{th}}$  position of the permutation after finishing the processing on machine  $k$  of problem  $l$ , while waiting for the machine  $k + 1$  of problem  $l$  to become available;
- $C_j^l$  completion time in problem  $l$  of the job in the  $j^{\text{th}}$  position of the permutation.

The makespan is given by the completion time of the job in the last position of the permutation. The model is written as follows:

$$\min C_n^1 \quad (1)$$

$$\sum_{j=1}^n z_{ij} = 1; i = 1, \dots, n \quad (2)$$

$$\sum_{i=1}^n z_{ij} = 1; j = 1, \dots, n \quad (3)$$

$$\sum_{i=1}^n p_{ri}^l z_{ij+1} + y_{j+1r}^l + x_{j+1r}^l = y_{jr}^l + \sum_{i=1}^n p_{r+1i}^l z_{ij} + x_{j+1r+1}^l; l = 1, 2, j = 1, \dots, n-1; r = 1, \dots, m_l \quad (4)$$

$$\sum_{r=1}^{k-1} \sum_{i=1}^n p_{ri}^l z_{i1} = x_{1k}^l; l = 1, 2, k = 2, \dots, m_l \quad (5)$$

$$\sum_{r=1}^{m_v} \sum_{i=1}^n p_{ri}^v z_{i1} \leq x_{1m_l+1}^l; l, v = 1, 2 \quad (6)$$

$$y_{1k}^l = 0; l = 1, 2, k = 1, \dots, m_l - 1 \quad (7)$$

$$x_{1m_l+1}^l - \left( x_{1m_l}^l + \sum_{i=1}^n p_{m_l i} z_{i1} \right) = y_{1m_l}^l; l = 1, 2 \quad (8)$$

$$\sum_{u=1}^j \sum_{i=1}^n p_{m_l+1i}^l z_{iu} + \sum_{u=1}^j x_{um_l+1}^l = C_j^l; l = 1, 2, j = 1, \dots, n \quad (9)$$

$$C_j^1 = C_j^2; j = 1, \dots, n \quad (10)$$

$$z_{ij} \in \{0, 1\}; j = 1, \dots, n, i = 1, \dots, n \quad (11)$$

$$y_{jk}^l, x_{jk}^l \geq 0; l = 1, 2, j = 1, \dots, n; k = 1, \dots, m_l + 1 \quad (12)$$

200 The objective function (1) minimizes the makespan. Constraints (2) and (3) assign one job to exactly one position in the permutation. Constraint (4) is the so called job-adjacency, machine linkage constraint [37]. Constraint (4) ensures equal time-slices on adjacent machines for each pair of consecutive jobs in the sequence. A time-slice between the completion of job in position  $j$  on machine  $r$  and the start of job in position  $j + 1$  on machine  $r + 1$  is analysed. 205 On the left side is computed the idle time on machine  $r$  before starting job in position  $j + 1$ , its processing time, and the idle time of the job in case machine  $r + 1$  is not free. On the right side is computed the idle time of job in position  $j$  before starting in machine  $r + 1$ , its processing time, and the idle time on 210 machine  $r + 1$  while waiting for job in position  $j + 1$  to finish in machine  $r$ . The computations on each side must occur in the same time-slice. Constraint (5) computes, from the second machine on, the idle time in each machine of each semi-line while waiting for the first job. Constraint (6) ensures that the idle time on the synchronization machine waiting for the first job is equal to 215 the larger total processing time on each semi-line. Constraint (7) ensures that there is no idle time for the job assigned to the first position in each machine of each semi-line, but the first job may wait to be processed on the synchronization machine, which is ensured by constraint (8). Constraints (9) and (10) ensure the synchronization. Constraints (11) and (12) impose the variation domain of 220 the variables.

The differences that have to be introduced in the model with respect to the classical permutation flowshop are due to the fact that we have synchronize each problem  $l$  at machines  $m_{l+1}$  (the synchronization operation duplicated to be the last machine in each problem  $l$ ). To do this we have to compute in (9) the completion time  $C_j^l$  for each job  $j$  in each flowshop problem  $l$ , instead of just the makespan, and impose them to be equal in (10).

This implies (i) that the  $m_{l+1}$  machine in problem  $l$  may have to wait for the first job in the sequence more than the sum of the processing times of such job on the previous machines (which is assumed to be equal in modeling the classical permutation flowshop). Thus, in (6) the waiting time on the  $m_{l+1}$  machine for the first job in each problem  $l$  must be greater or equal than the processing times in the previous machines of all problems, i.e., the two halves of a job must be completed before start the synchronization operation. And (ii) that the first job of the sequence may have to wait in problem  $l$  before starts its operation at the  $m_{l+1}$  machine (which is assumed to not occur in modeling the classical permutation flowshop). Indeed, we compute in (8) the waiting time of the first job after being completed in the  $m_l$  machine in problem  $l$  because machine  $m_{l+1}$  has to be synchronized.

Let us illustrate with the example in Figure 2. The synchronization operation is the third machine in each problem  $l$ . We have  $x_{1,3}^1 = 4$ , which is greater than the time spent by job yellow to be processed in the first two machines of problem  $l = 1$ , since the third machine of problem  $l = 1$  has to wait job yellow to be processed in the first two machines of problem  $l = 2$ . We also have  $y_{1,2}^1 = 1$ , which is the time job yellow has to wait before start processing at the third machine due to synchronization.

### 3. Heuristics

In this section, we propose adaptations of the NEH heuristic by Nawaz et al. [10] and also adaptations of the algorithm of Johnson [1] to obtain feasible solutions for the flowshop scheduling problem with parallel semi-lines and the

250 synchronization operation.

### 3.1. Adaptations of the NEH heuristic

The NEH heuristic for the classical permutation flowshop starts by sorting the  $n$  jobs in decreasing order of the sums of processing times on all the  $m$  machines. Then, a partial scheduling consists of the first two jobs of this order  
255 in a sequence that minimizes the makespan. The other jobs, from the third, are inserted (one at a time) in the position of the partial scheduling leading to the smallest makespan. The relative positions between jobs already inserted in the partial scheduling do not change. We develop three adaptations of the NEH algorithm :  $\text{NEH}_{\text{av}}$  - the average of the processing times  $p_{ki}^1$  and  $p_{ki}^2$  for each  
260  $k = 1, \dots, m_l$  and for each job  $i$ ,  $\text{NEH}_{\text{hi}}$  - the highest processing time between  $p_{ki}^1$  and  $p_{ki}^2$  for each  $k = 1, \dots, m_l$  and for each job  $i$ , and  $\text{NEH}_{\text{sep}}$  - where each semi-line is considered separately including the synchronization operation.

The  $\text{NEH}_{\text{av}}$  and the  $\text{NEH}_{\text{hi}}$  heuristics require the same number of machines in each semi-line, i.e,  $m = m_1 = m_2$ . Let  $f$  designate the final synchronization  
265 machine. The general principle is to reduce the two semi-lines to a single line and apply the NEH heuristic. We do this by replacing at each stage the corresponding machines in each semi-line for a a single machine, as illustrated in Figure 3.

In the  $\text{NEH}_{\text{av}}$  heuristic, for each job  $j = 1, \dots, n$ , we compute  $\bar{p}_{kj} = (p_{kj}^1 +$   
270  $p_{kj}^2)/2$ , where  $k = 1, \dots, m$  is the  $k^{\text{th}}$ -machine in each semi-line. At this point we have an instance of the classical permutation flowshop with  $m + 1$  machines where, for each job  $j$ ,  $\bar{p}_{kj}$  is the processing time on machine  $k = 1, \dots, m$  and  $p_{fj}$  is the processing time on the last machine, and apply the NEH heuristic to obtain a sequence  $\text{seq}_{\text{av}}$ . Finally, we compute the makespan incurred by the  
275 sequence  $\text{seq}_{\text{av}}$  in the whole system with the actual processing times  $p_{kj}^l$  for each semi-line  $l = 1, 2$ .

The  $\text{NEH}_{\text{hi}}$  heuristic works in a similar manner. For each job  $j = 1, \dots, n$ , we compute  $\bar{p}_{kj} = \max\{p_{kj}^1, p_{kj}^2\}$ , where  $k = 1, \dots, m$  is the  $k^{\text{th}}$ -machine in each semi-line. Analogously to  $\text{NEH}_{\text{av}}$ , we obtain by the NEH heuristic a sequence

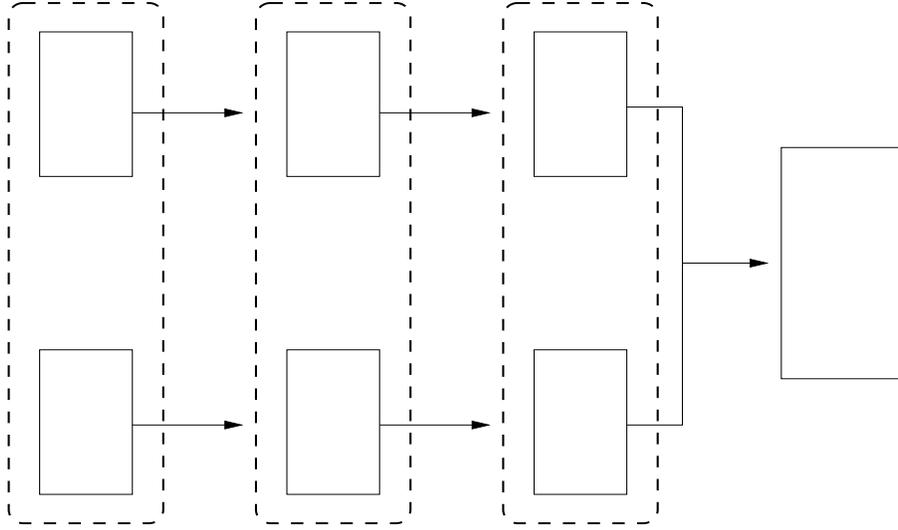


Figure 3: Reducing the semi-lines to a single line,  $m = 3$ .

280  $\text{seq}_{\text{hi}}$  for an instance of the classical permutation flowshop with  $m + 1$  machines where, for each job  $j$ ,  $\bar{p}_{kj}$  is the processing time on machine  $k = 1, \dots, m$  and  $p_{fj}$  is the processing time on the last machine. We then compute the makespan incurred by the sequence  $\text{seq}_{\text{hi}}$  in the whole system with the actual processing times  $p_{kj}^l$  for each semi-line  $l = 1, 2$ .

285 In the  $\text{NEH}_{\text{sep}}$  heuristic we generate two instances of the classical permutation flowshop.  $\text{NEH}_{\text{sep}}$  does not require the same number of machines in each semi-line. Each instance is composed of the machines of one of the semi-lines along with the synchronization operation, as illustrated in Figure 4. We apply, to obtain a sequence  $\text{seq}_{\text{sep}}^l$ , the NEH heuristic to each instance  $l = 1, 2$  with  
 290  $m_l + 1$  machines where, for each job  $j$ ,  $p_{kj}^l$  is the processing time on machine  $k = 1, \dots, m_l$  and  $p_{fj}$  is the processing time on the last machine. We adopt the sequence  $\text{seq}_{\text{sep}}^l$ ,  $l = 1, 2$ , leading to the smallest makespan in the whole system with the two semi-lines and the synchronization operation.

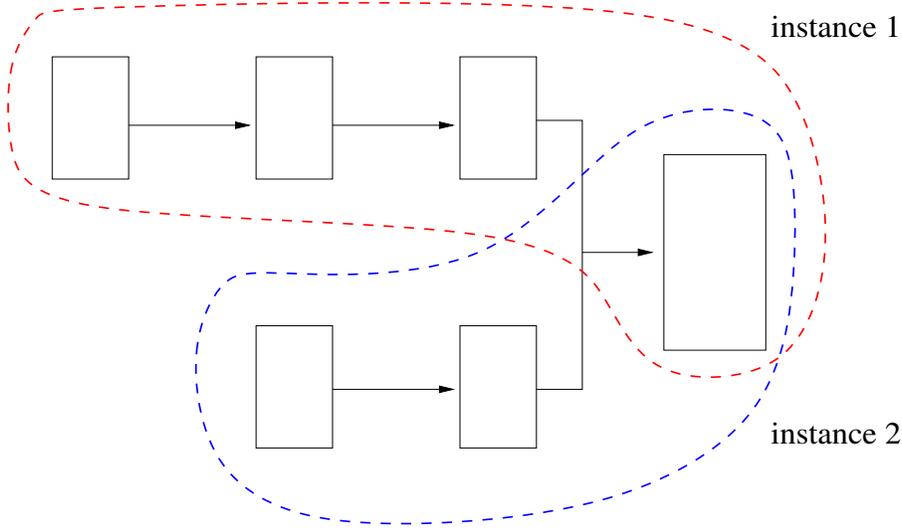


Figure 4: Generating two instances of the classical permutation flowshop.

### 3.2. Adaptations of Johnson's algorithm

295 Johnson's algorithm obtains a sequence that minimizes the makespan for the flowshop problem with two machines. The optimal sequence begins with the jobs having the processing time on the first machine smaller than the processing time on the second machine sorted in increasing order of processing times on the first machine, and ends with the remaining jobs in decreasing order of  
 300 processing times on the second machine. We develop two adaptations of the Johnson's algorithm. The general principle is to consider the studied system as a flowshop with two machines by setting the synchronization operation as the second machine.

In the adaptation denoted by  $\text{Joh}_{\text{av}}$ , for each job  $j = 1, \dots, n$ , the average processing time  $\frac{\sum_{k=1}^{m_1} p_{kj}^1 + \sum_{k=1}^{m_2} p_{kj}^2}{m_1 + m_2}$  in the machines of the two semi-lines is used as the processing time on the first machine. The processing time of the synchronization operation is the processing time on the second machine. We then apply Johnson's algorithm, and compute, for the sequence obtained, the makespan in the whole system with the actual processing times  $p_{kj}^l$  for each  
 310 semi-line  $l = 1, 2$ . The adaptation denoted by  $\text{Joh}_{\text{hi}}$  works in a similar manner,

but the largest processing time  $\max_{l=1,2,k=1,\dots,m_l} \{p_{kj}^l\}$  among the machines of the semi-lines is used as the processing time of job  $j$  on the first machine.

#### 4. Metaheuristics

We propose to use the previously described heuristics in the construction phase of a GRASP algorithm. GRASP was successfully applied to the permutation flowshop by Prabhakaran et al. [21], and to a multi-objective variant by Arroyo and de Souza Pereira [38]. Alternatively, we propose a simulated annealing algorithm, which was also successfully applied to the permutation flowshop by Hurkała and Hurkała [39], and to a multi-objective variant by Jarosław et al. [40].

##### 4.1. GRASP

GRASP is a multistart metaheuristic, see, for instance, Resende and Ribeiro [41]. A GRASP iteration consists basically of two phases: a construction phase that builds a feasible solution using a randomized greedy heuristic, followed by a local search phase. We propose two versions of GRASP to try to find optimal or near-optimal solutions for the flowshop problem with parallel semi-lines and the synchronization operation. These variants differ in the heuristic employed in the construction phase. One version uses  $\text{NEH}_{\text{sep}}$  and the other uses  $\text{Joh}_{\text{av}}$ , since these were the procedures to obtain the best results in our computational experiments, c.f., Section 5.

Figure 5 presents the pseudo-code of the construction phase of  $\text{GRASP}_{\text{NEH}_{\text{sep}}}$ . The procedure is written for a number  $L$  of semi-lines. The main loop in lines 1 to 10 treats, as  $\text{NEH}_{\text{sep}}$ , each semi-line  $l$  along with the synchronization operation as an independent instance of the classical permutation flowshop. In line 2, for each job  $j$ ,  $p_j$  is the sum of processing times on all the  $m_l + 1$  machines. The inner loop in lines 3 to 8 applies a randomized version of  $\text{NEH}_{\text{sep}}$ . Instead of taking each time a job in decreasing order of  $p_j$ , a job  $q$  is randomly chosen from the Restricted Candidate List  $\text{RCL}_l$ . Given a percentage  $\alpha$  of the total

number  $n$  of jobs, at each time,  $RCL_l$  contains the  $\alpha n$  jobs with the largest  
340  $p_j$  not yet added to the sequence (if it remains lesser jobs than  $\alpha n$ , then all  
remaining jobs are inserted at  $RCL_l$ ). The insertion of job  $q$  randomly selected  
from  $RCL_l$  in the sequence under construction is done according to the NEH  
heuristic. In line 9, the makespan  $mk^l$  incurred by  $seq^l$  in the whole system  
with the two semi-lines and the synchronization operation is computed. At  
345 each GRASP iteration the construction phase returns, in line 11, the sequence  
 $seq^l$ ,  $l = 1, \dots, L$ , leading to the smallest makespan.

**Procedure Construction Phase GRASP\_NEH<sub>sep</sub>**

- 1 **For**  $l = 1$  **to**  $L$  **do**
- 2     Let  $J$  be the set of  $n$  jobs, and compute  $p_j = \sum_{k=1}^{m_l} p'_{kj} + p_{fj}$  for each job  $j \in J$ .
- 3     **For**  $t = 1$  **to**  $n$  **do**
- 4         Let  $RCL_l \subseteq J$  be the set of the  $\min\{\alpha n, n - t + 1\}$  jobs with the largest  $p_j$ .
- 5         Take at random a job  $q \in RCL_l$ .
- 6         Insert  $q$  in the best of the  $t$  possible positions in the partial sequence  $seq^l$
- 7         Let  $J = J - \{q\}$
- 8     **End-For**
- 9     Compute be the makespan  $mk^l$  incurred by  $seq^l$  in the whole system.
- 10 **End-For**
- 11 Return the sequence  $seq^l$ ,  $l = 1, \dots, L$ , leading to the smallest  $mk^l$ .

**End-Procedure**

Figure 5: Pseudo-code of the construction phase of GRASP\_NEH<sub>sep</sub>.

Figure 6 presents the pseudo-code of the construction phase of GRASP\_Joh<sub>av</sub>.  
The procedure uses two Restricted Candidate Lists:  $RCL_u$  for the upward part,  
and  $RCL_d$  for the downward part of the sequence  $seq^{Jav}$ . In line 1, for each  
350 job  $j$ ,  $p_j$  is the average processing time on all the machines of the  $L$  semi-lines  
(the processing time of job  $j$  reducing the semi-lines to the first machine, the  
synchronization operation being the second machine). As in Johnson's algo-  
rithm, in line 2 the jobs are partitioned in two subsets  $J_u$  and  $J_d$  by comparing  
 $p_j$  to  $p_{fj}$ . Note that  $n_u$  and  $n_d$  are set in line 3, and do not change along the  
355 procedure. The first loop in lines 4 to 9 builds with a randomized version of

Johnson's algorithm the upward part of  $\text{seq}^{\text{Jav}}$ . Instead of taking each time a job in increasing order of  $p_j$ , a job  $q$  is randomly chosen from  $\text{RCL}_u$ . Given a percentage  $\alpha$  of  $n_u$ , at each time,  $\text{RCL}_u$  contains the  $\alpha n_u$  jobs with the smallest  $p_j$  not yet added to the sequence (if it remains lesser jobs than  $\alpha n_u$ , then all remaining jobs in  $J_u$  are inserted at  $\text{RCL}_u$ ). As in Johnson's algorithm, job  $q$  is inserted in the last position of  $\text{seq}^{\text{Jav}}$ . The second loop in lines 10 to 14 builds in an analogous manner the downward part of  $\text{seq}^{\text{Jav}}$ , as each time a job  $q$  is randomly chosen from  $\text{RCL}_d$  which contains the  $\alpha n_d$  jobs with the largest  $p_j$  not yet added to the sequence. The makespan incurred by  $\text{seq}^{\text{Jav}}$  in the whole system with the two semi-lines and the synchronization operation is computed in line 16, and  $\text{seq}^{\text{Jav}}$  is returned by the construction phase at each GRASP iteration in line 17.

**Procedure Construction Phase GRASP\_Joh<sub>av</sub>**

- 1 Let  $J$  be the set of  $n$  jobs, and compute  $p_j = \frac{\sum_{l=1}^L \sum_{k=1}^{m_l} p_{kj}^l}{\sum_{l=1}^L m_l}$  for each job  $j \in J$ .
- 2 Let  $J_u \subseteq J$  (resp.  $J_d \subseteq J$ ) be the set of jobs such that  $p_j \leq p_{fj}$  (resp.  $p_j > p_{fj}$ ).
- 3 Set  $n_u = |J_u|$  and  $n_d = |J_d|$ .
- 4 **For**  $t = 1$  **to**  $n_u$  **do**
- 5     Let  $\text{RCL}_u \subseteq J_u$  be the set of the  $\min\{\alpha n_u, n_u - t + 1\}$  jobs with the smallest  $p_j$ .
- 6     Take at random a job  $q \in \text{RCL}_u$ .
- 7     Insert  $q$  in the  $t$ -th position in the partial sequence  $\text{seq}^{\text{Jav}}$
- 8     Let  $J_u = J_u - \{q\}$
- 9 **End-For**
- 10 **For**  $t = 1$  **to**  $n_d$  **do**
- 11     Let  $\text{RCL}_d \subseteq J_d$  be the set of the  $\min\{\alpha n_d, n_d - t + 1\}$  jobs with the largest  $p_j$ .
- 12     Take at random a job  $q \in \text{RCL}_d$ .
- 13     Insert  $q$  in the  $(n_u + t)$ -th position in the partial sequence  $\text{seq}^{\text{Jav}}$
- 14     Let  $J_d = J_d - \{q\}$
- 15 **End-For**
- 16 Compute the makespan incurred by  $\text{seq}^{\text{Jav}}$  in the whole system.
- 17 Return the sequence  $\text{seq}^{\text{Jav}}$ .

**End-Procedure**

Figure 6: Pseudo-code of the construction phase of GRASP\_Joh<sub>av</sub>.

The local search for both versions of GRASP is based on swap moves. Let  $\pi$  be a permutation of the  $n$  jobs. The local search procedure evaluates all the possible  $O(n^2)$  moves swapping pairs of jobs at two different positions in  $\pi$ . Initially,  $\pi$  is set as the permutation returned by the construction phase. The local search uses the strategy of best improvement move. Let  $\pi'$  be the solution with lowest cost in the neighborhood of  $\pi$ . If the cost of  $\pi'$  is lower than the cost of  $\pi$ , then  $\pi$  is updated to  $\pi'$  and the search resumes. Otherwise  $\pi$  is returned as the local optimum.

#### 4.2. Simulated annealing

We also use  $\text{NEH}_{\text{sep}}$  and  $\text{Joh}_{\text{av}}$  as initial solutions to a Simulated Annealing (SA) algorithm. Given a current solution  $s$ , SA proceeds generating at each iteration a neighbor solution  $s'$  with a swap move at random. If  $s'$  improves the makespan of  $s$ , then the current solution is updated. To prevent getting stuck in a local optima, the algorithm allows some worsening solutions. This is done by respecting a probability of allowance in relation to a temperature  $T$ . Let  $\Delta$  be the difference in the makespan between  $s'$  and  $s$ . The worsening solution is accepted if a randomly chosen value between 0 and 1 is lower than  $e^{-\Delta/T}$ . The algorithm stopping criterion is determined by the slow cooling of the initial temperature. After a number of iterations with the same temperature without improvement, the temperature is updated to  $\alpha T$ ,  $\alpha \in (0, 1)$ . Note that GRASP and SA exploit the same neighborhood. At the end of SA, as an intensification strategy, we apply the local search based on swap moves to the solution returned by SA.

### 5. Computational experiments

The computational experiments were structured into three comparative settings: comparison of the heuristics proposed in Section 3 in terms of solution quality, tuning parameters of the metaheuristics proposed in Section 4, and evaluation of the effectiveness of the best metaheuristic configurations with respect to optimal or lower bounds obtained with the MIP model proposed in Section 2.

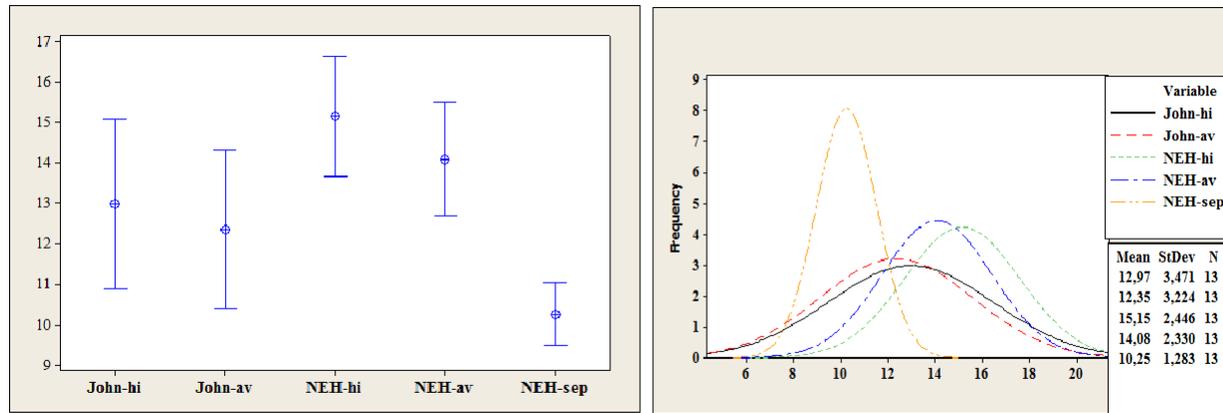
For such purpose, we generated two set of instances according to the guidelines introduced by Taillard [42]. In the first set of instances the semi-lines have the same number of machines each, while in the second set they have different  
400 number of machines. A total of 230 instances were generated with 10, 20, and 50 jobs, and 3, 5, 7, and 11 machines, including the final synchronization machine. For example, an instance with 3 machines belonging to the first set has 2 machines in each semi-line, and the final synchronization machine, whereas an instance with 3 and 5 machines belonging to second the set has 2 machines in  
405 one semi-line, 4 machines in the other semi-line, and the final synchronization machine. The set with the same number of machines contains 120 instances: 10 instances for each combination number of jobs - number of machines. The set with different number of machines contains 110 instances: 50 instances with 10 jobs, 50 instances with 20 jobs, and 10 instances with 50 jobs.

410 The computational experiments were **run** on a Intel Core i3, 3.1GHz with 4GB of RAM. The results of the MIP model were obtained with CPLEX 12.6.1, and the proposed algorithms were implemented in C++. We were able to obtain optimal solutions for 186 instances.

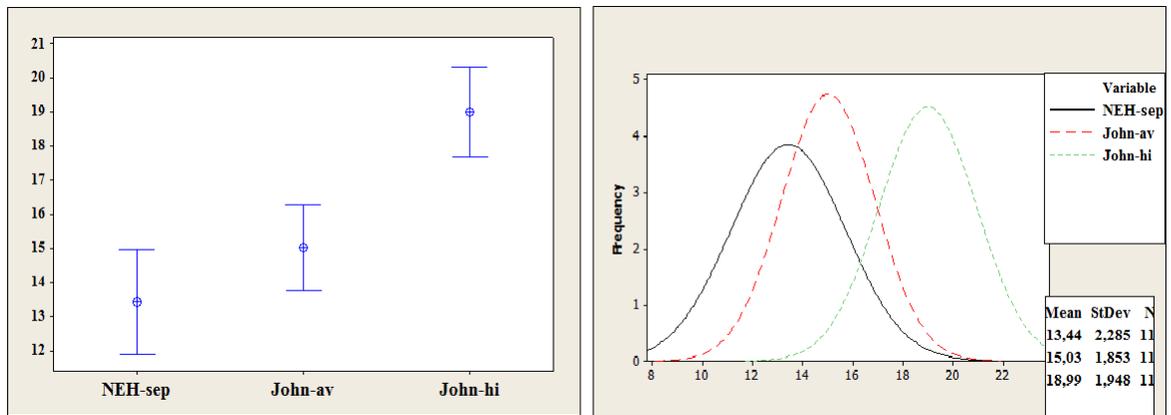
In the first two comparative settings, i.e., comparisons of the heuristics proposed in Section 3 and tuning parameters of the metaheuristics proposed in  
415 Section 4, the quality of the solution obtained by the proposed algorithms is measured by the percentage gap  $= \frac{ub-opt}{opt} * 100$ , where *ub* is the solution obtained by the proposed algorithm and *opt* is the optimal or the best solution obtained in these settings (in the case of the remaining open instances).

420 Figure 7 shows, for each heuristic proposed in Section 3, the average percentage gap, the range interval of the average gaps, and the fit to a normal distribution. Figure 7a shows results for the instances with the same number of machines in each semi-line, and Figure 7b results for the instances with different number of machines.

425 Considering the results shown in Figure 7a, we can see that the idea of reducing the problem to a classical permutation flowshop is not effective, since  $NEH_{av}$  and  $NEH_{hi}$  present the higher gaps. The performance of the adaptations



(a) Same number of machines.



(b) Different number of machines.

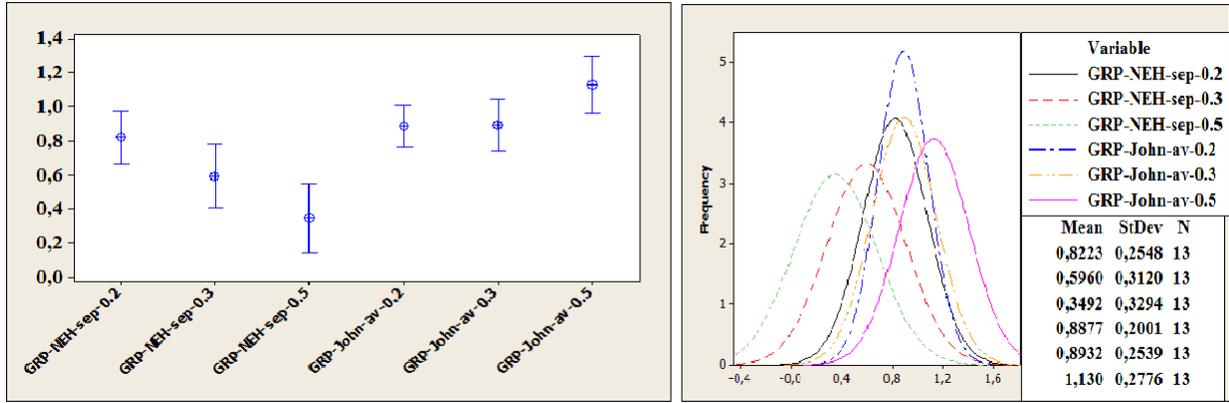
Figure 7: Comparison of the proposed heuristics.

of Johnson’s algorithm vary considerably, since  $Joh_{av}$  and  $Joh_{hi}$  present the higher standard deviations. The heuristic  $NEH_{sep}$  obtained the best average results on the both set of instances. We remark that for the instances for which the optimal solution were obtained, the optimal values are higher than the optimal values considering each semi-line along with the synchronization machine as a classical permutation flowshop, i.e., there is not a semi-line that dominates the other one. The average gaps observed for  $NEH_{sep}$ ,  $Joh_{av}$ , and  $Joh_{hi}$  are higher for the set with different number of machines. As  $NEH_{sep}$  and  $Joh_{av}$  were the heuristics to present the best results, they were used in our GRASP and SA algorithms, c.f., Section 4.

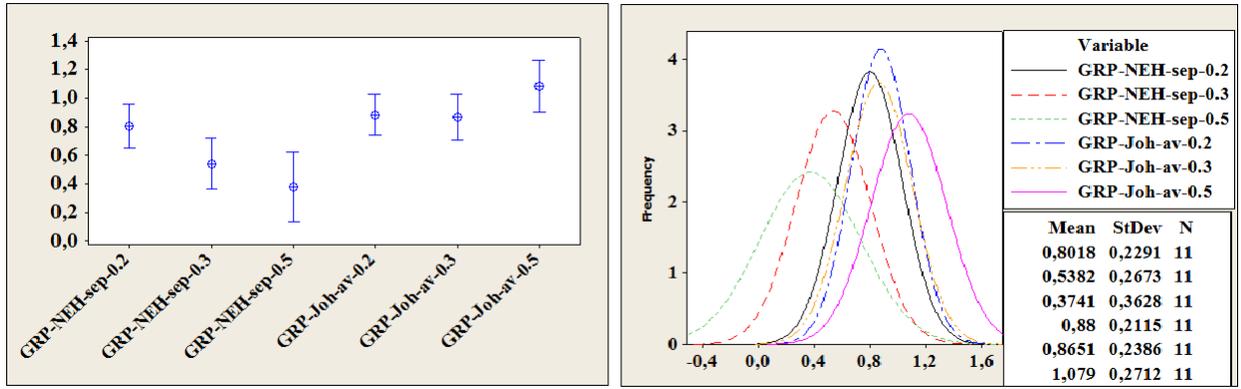
Figure 8 shows results for GRASP with different values of the parameter  $\alpha$  which controls the cardinality of the RCL. The number of GRASP iterations without improvement, i.e., the stopping criteria, was set to 100. Increasing the value of  $\alpha$  leads to a RCL with a larger cardinality. We consider for  $\alpha$  the values 20%, 30%, and 50%. We were able to drastically reduce solution costs embedding  $NEH_{sep}$  and  $Joh_{av}$  into GRASP.  $GRASP_{NEH_{sep}}$  with  $\alpha = 50\%$  presents average gaps of less than 0.5% to the optimal or to the best solution obtained in this setting.

Figure 9 shows results for SA with different values of the parameter  $\alpha$ , the cooling factor, which controls the reduction of the temperature. The initial and the final temperatures were set to 6000 and  $10^{-4}$ , respectively. The number of SA iterations with a **constant** temperature without improvement was set to 90. We consider for  $\alpha$  the values 0.20, 0.50, and 0.95. Although important improvements can be observed, SA was not able to improve average gaps in the same manner as GRASP. The best results with SA presents average gaps higher than 2%.

We now report in a final comparative setting detailed results to assess the effectiveness of GRASP. For such purpose we report optimality gaps computed with respect to optimal or best lower bounds obtained with the MIP model proposed in Section 2. Since  $GRASP_{NEH_{sep}}$  with  $\alpha = 50\%$  obtained the best results in the previous experiments, we investigate greater values of  $\alpha = 75\%$ ,

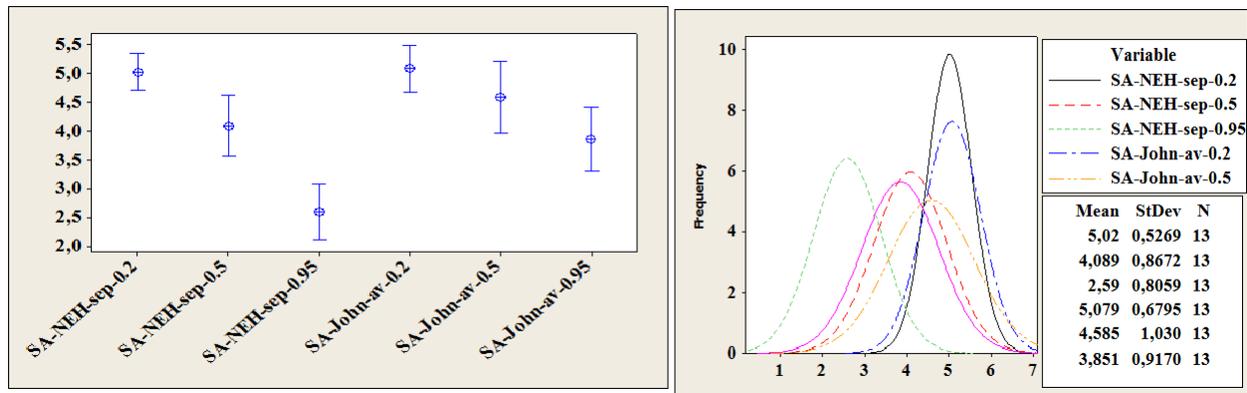


(a) Same number of machines.

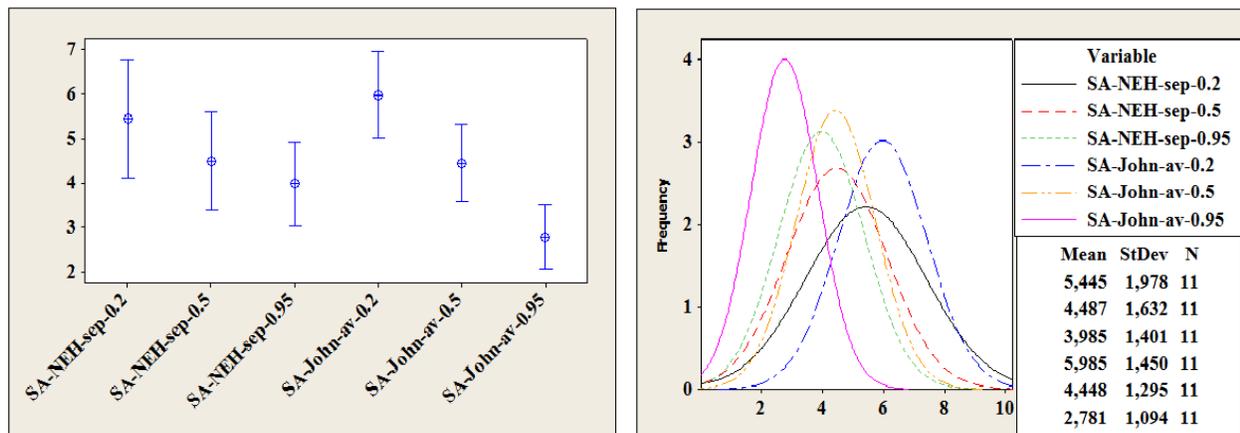


(b) Different number of machines.

Figure 8: GRASP performances regarding different values of the Restricted Candidate List (RCL) parameter.



(a) Same number of machines.



(b) Different number of machines.

Figure 9: SA performances regarding different the values of the cooling factor parameter.

and  $\alpha = 100\%$ .<sup>1</sup> It turns out that best results were obtained with  $\alpha = 100\%$ ,  
460 which corresponds to a random multistart algorithm, i.e., each iteration consists  
of generating a completely random permutation and applying the local search  
procedure.

Tables 2 and 3 present average results for GRASP\_NEH<sub>sep</sub> with  $\alpha = 50\%$ ,  
 $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances with the same and different number of  
465 machines in each semi-line, respectively. Each line of the tables corresponds  
to average results for 10 instances of a given combination of number of jobs -  
number of machines. The first column indicates the number of jobs and the  
number of machines. For example, in Table 2,  $E10 \times 03$  indicates instances with  
10 jobs, 2 machines in each semi-line, and the final synchronization machine,  
470 whereas in Table 3,  $D10 \times 03 \times 05$  indicates instances with 10 jobs, 2 machines  
in one semi-line, 4 machines in the other semi-line, and the final synchronization  
machine. For the MIP model it is shown, in the second and third columns, the  
number of instances for each combination solved to optimality and the average  
computational time in seconds, respectively. Then, for each value of  $\alpha$ , it is  
475 shown the number of instances for each combination the optimal solution was  
found, the average optimality gap in percentage, the average computational  
time in seconds, and the average total number of iterations.

Note that not only the number of jobs, but also the number of machines play  
an important role in how difficult is to solve the problem to optimality with the  
480 MIP model. For example, instances with 50 jobs and 3 machines in each semi-  
line were solved 10 times faster in average than instances with 20 jobs and 5  
machines in each semi-line. Instances with a large number of jobs and machines  
were out of reach for the MIP model. Results show that GRASP\_NEH<sub>sep</sub> is  
an effective and robust algorithm to tackle the problem. The overall average  
485 optimality gaps are smaller than 1% for all three values of  $\alpha$ , and only for  
4 combinations with 11 machines out of 23 combinations of number of jobs

---

<sup>1</sup>We thank the anonymous reviewer for his/her suggestion to investigate greater values of  $\alpha$ .

- number of machines average optimality gaps are in the range between 1% and 5%. In particular GRASP\_NEH<sub>sep</sub> with  $\alpha = 100\%$ , the random multistart version, obtained the optimal solution for 186 out of 230 instances and for only  
490 2 combinations the average optimality gap exceeded 1% ( $E50 \times 11$  with 2.43% and  $D20 \times 03 \times 11$  with 1.14%). Moreover, these results were obtained with low computational times, even for instances out of reach for the MIP model.

Tables 4 to 9 present detailed results for the combinations which optimal values were not obtained for most instances, namely:  $E20 \times 07$ ,  $E20 \times 11$ ,  
495  $E50 \times 07$ ,  $E50 \times 11$ ,  $D20 \times 03 \times 11$ , and  $D20 \times 05 \times 11$ , in this order. In these tables, the first column identifies the instance, and the second column presents the best lower bound obtained when running the MIP model. Then, for each value of  $\alpha$ , it is shown for each instance the upper bound obtained, the optimality gap in percentage, the computational time in seconds, and the  
500 total number of iterations. The detailed results on the harder instances confirm that GRASP\_NEH<sub>sep</sub> is an effective and robust algorithm to tackle the problem, specially GRASP\_NEH<sub>sep</sub> with  $\alpha = 100\%$  that obtained optimality gaps below 5% for all instances but  $E50 \times 11 - 6$ .

## 6. Conclusion

505 This work focused on the development of optimization methods to solve a variant of the permutation flowshop scheduling problem with parallel semi-lines and a final synchronization operation. This study emerged from a practical situation in a welding process of an industry that manufactures electro-electronic system products, and may be a common problem in other industrial sectors as  
510 well. Our purpose was to design efficient methods to solve the problem, which, to the best of our knowledge, has not yet been treated.

In a first approach, a mixed-integer linear programming model, inspired from the classical permutation flowshop, was introduced. This model required adap-  
515 tations regarding to the treatment of the semi-lines separately and a particular

attention in the finalization of the semi-lines since this is the point of junction of the two halves of the jobs that are produced in the two semi-lines. The model was efficient for small instances, but for larger instances, it was not able to find optimal solutions in moderate computational times. We also proposed  
520 constructive heuristics and metaheuristics in an attempt to find optimal or near-optimal solutions with reasonable computational times. In particular, an adaptation of the NEH heuristic embedded in a GRASP algorithm lead to an effective and robust algorithm to tackle the problem, obtaining average optimality gaps of less than 1% in low computational times.

525

Future extension of our work could focus on further exploration of problem-specific characteristics and developing more effective methods and local search procedures for this problem. Moreover, due to the effectiveness of the meta-heuristic methods, it could be interesting to analyze their performances by using  
530 other objective criteria such as the total flowtime or total tardiness.

Instances	MIP model			GRASP_NEH <sub>sep</sub> $\alpha = 50\%$			GRASP_NEH <sub>sep</sub> $\alpha = 75\%$			GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)					
	opt	t(s)	it	opt	g(%)	t(s)	it	opt	g(%)	t(s)	it	opt	g(%)	t(s)	it
	$E_{10 \times 03}$	10	1	105	6	0.09	27	105	10	0.00	35	110	10	0.00	30
$E_{10 \times 05}$	10	1	118	6	0.11	34	118	10	0.00	38	116	10	0.00	30	111
$E_{10 \times 07}$	10	2	117	6	0.12	48	117	10	0.00	50	114	10	0.00	48	110
$E_{10 \times 11}$	10	11	112	3	0.27	46	112	8	0.02	52	112	10	0.00	49	112
$E_{20 \times 03}$	10	1	110	5	0.09	44	110	10	0.00	48	112	10	0.00	49	110
$E_{20 \times 05}$	10	405	112	5	0.12	44	112	7	0.05	45	122	10	0.00	49	119
$E_{20 \times 07}$	4	2289	132	4	0.14	43	132	5	0.23	51	139	4	0.14	48	136
$E_{20 \times 11}$	0	7542	143	0	2.29	43	143	0	1.29	52	146	0	0.93	47	139
$E_{50 \times 03}$	10	6	106	5	0.11	46	106	8	0.01	51	140	10	0.00	49	148
$E_{50 \times 05}$	9	4916	152	4	0.18	47	152	7	0.05	51	145	9	0.02	47	149
$E_{50 \times 07}$	4	13695	149	0	0.58	57	149	3	0.13	59	160	5	0.19	60	141
$E_{50 \times 11}$	0	12746	166	0	3.39	57	166	0	4.08	61	174	0	2.43	60	165
av gap(%)					0.62				0.49				0.31		

Table 2: Average results for GRASP\_NEH<sub>sep</sub> with  $\alpha = 50\%$ ,  $\alpha = 75\%$  and  $\alpha = 100\%$  - same number of machines.

Instances	MIP model			GRASP_NEH <sub>sep</sub> $\alpha = 50\%$			GRASP_NEH <sub>sep</sub> $\alpha = 75\%$			GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)						
	opt	t(s)	it	opt	g(%)	t(s)	it	opt	g(%)	t(s)	it	opt	g(%)	t(s)	it	
	$D_{10 \times 03 \times 05}$	10	1	108	6	0.07	27	108	8	0.03	27	112	10	0.00	26	118
$D_{10 \times 03 \times 07}$	10	1	120	5	0.54	33	120	7	0.05	33	119	10	0.00	33	119	
$D_{10 \times 03 \times 11}$	10	1	121	6	0.36	48	121	7	0.06	45	122	10	0.00	46	121	
$D_{10 \times 05 \times 07}$	10	1	128	6	0.14	45	128	9	0.01	45	122	10	0.00	48	127	
$D_{10 \times 05 \times 11}$	10	2	126	5	0.06	46	126	7	0.03	45	125	10	0.00	52	122	
$D_{20 \times 03 \times 05}$	10	4	121	5	0.23	48	121	8	0.08	49	126	10	0.00	53	129	
$D_{20 \times 03 \times 07}$	10	53	127	4	0.32	44	127	7	0.05	49	130	10	0.00	54	132	
$D_{20 \times 03 \times 11}$	5	14250	140	0	1.88	50	140	1	0.62	47	138	4	1.14	52	138	
$D_{20 \times 05 \times 07}$	10	200	135	3	0.14	49	135	7	0.04	48	137	10	0.00	52	137	
$D_{20 \times 05 \times 11}$	4	18208	145	0	1.24	51	145	0	0.62	48	146	4	0.64	52	166	
$D_{50 \times 03 \times 05}$	10	154	156	4	0.08	58	156	9	0.01	50	149	10	0.00	53	164	
av gap(%)					0.46											0.16

Table 3: Average results for GRASP\_NEH<sub>sep</sub> with  $\alpha = 50\%$ ,  $\alpha = 75\%$  and  $\alpha = 100\%$  - different number of machines.

Instances	GRASP_NEH <sub>sep</sub> $\alpha = 50\%$				GRASP_NEH <sub>sep</sub> $\alpha = 75\%$				GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)				
	LB	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it
	$E_{20 \times 07 - 1}$	642	643	0.16	43	115	645	0.47	52	136	643	0.16	53
$E_{20 \times 07 - 2}$	946	946	0.00	44	160	946	0.00	53	150	946	0.00	50	126
$E_{20 \times 07 - 3}$	1284	1284	0.00	42	131	1284	0.00	53	122	1284	0.00	43	134
$E_{20 \times 07 - 4}$	725	725	0.00	42	129	725	0.00	43	146	725	0.00	45	145
$E_{20 \times 07 - 5}$	1063	1066	0.28	48	138	1070	0.66	46	125	1070	0.66	49	139
$E_{20 \times 07 - 6}$	1357	1362	0.37	42	133	1369	0.88	48	155	1359	0.15	47	145
$E_{20 \times 07 - 7}$	1357	1362	0.37	45	132	1360	0.22	55	165	1360	0.22	49	125
$E_{20 \times 07 - 8}$	1361	1363	0.15	42	136	1362	0.07	62	128	1362	0.07	51	149
$E_{20 \times 07 - 9}$	1342	1343	0.07	44	138	1342	0.00	54	124	1344	0.15	49	150
$E_{20 \times 07 - 10}$	690	690	0.00	43	113	690	0.00	48	136	690	0.00	49	125

Table 4: Detailed results for GRASP with  $\alpha = 50\%$ ,  $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances  $E_{20 \times 07}$ .

Instances	GRASP_NEH <sub>sep</sub> $\alpha = 50\%$						GRASP_NEH <sub>sep</sub> $\alpha = 75\%$						GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)							
	LB	UB	g(%)	t(s)	it	it	UB	g(%)	t(s)	it	it	UB	g(%)	t(s)	it	it	UB	g(%)	t(s)	it
	$E_{20} \times 11 - 1$	805	809	0.50	44	150	44	812	0.87	50	143	50	808	0.37	48	141		808	0.37	48
$E_{20} \times 11 - 2$	1601	1608	0.44	45	135	45	1606	0.31	54	147	54	1604	0.19	43	133		1604	0.19	43	133
$E_{20} \times 11 - 3$	1375	1376	0.07	42	115	42	1382	0.44	50	118	50	1380	0.36	46	141		1380	0.36	46	141
$E_{20} \times 11 - 4$	1229	1296	5.45	44	157	44	1234	0.41	49	155	49	1234	0.41	46	156		1234	0.41	46	156
$E_{20} \times 11 - 5$	926	1018	9.93	46	112	46	929	0.32	53	160	53	930	0.43	46	149		930	0.43	46	149
$E_{20} \times 11 - 6$	1687	1706	1.13	40	155	40	1700	0.77	50	124	50	1700	0.77	44	139		1700	0.77	44	139
$E_{20} \times 11 - 7$	1593	1600	0.44	40	117	40	1600	0.44	54	165	54	1600	0.44	44	145		1600	0.44	44	145
$E_{20} \times 11 - 8$	1581	1608	1.71	43	169	43	1600	1.20	54	174	54	1606	1.58	56	113		1606	1.58	56	113
$E_{20} \times 11 - 9$	1599	1638	2.44	44	132	44	1639	2.50	55	137	55	1635	2.25	51	126		1635	2.25	51	126
$E_{20} \times 11 - 10$	1600	1613	0.81	43	184	43	1690	5.62	49	133	49	1640	2.50	47	148		1640	2.50	47	148

Table 5: Detailed results for GRASP with  $\alpha = 50\%$ ,  $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances  $E_{20} \times 11$ .

Instances	GRASP_NEH <sub>sep</sub> $\alpha = 50\%$						GRASP_NEH <sub>sep</sub> $\alpha = 75\%$						GRASP_NEH <sub>sep</sub> $\alpha = 100\%$																	
	LB			UB			g(%)			t(s)			it			UB			g(%)			t(s)			it					
	(Random Multistart)																													
$E50 \times 07 - 1$	2081	2128	2081	2.26	57	166	2083	0.10	58	172	2092	0.53	59	129	2083	0.10	58	172	2092	0.53	59	129	2083	0.10	58	172	2092	0.53	59	129
$E50 \times 07 - 2$	1455	1460	1455	0.34	55	166	1456	0.07	57	170	1455	0.00	58	144	1456	0.07	57	170	1455	0.00	58	144	1456	0.07	57	170	1455	0.00	58	144
$E50 \times 07 - 3$	945	946	945	0.11	58	124	945	0.00	59	139	945	0.00	61	165	945	0.00	59	139	945	0.00	61	165	945	0.00	59	139	945	0.00	61	165
$E50 \times 07 - 4$	1603	1605	1603	0.12	58	127	1604	0.06	59	164	1603	0.00	57	149	1604	0.06	59	164	1603	0.00	57	149	1604	0.06	59	164	1603	0.00	57	149
$E50 \times 07 - 5$	2064	2066	2064	0.10	57	133	2064	0.00	59	139	2064	0.00	62	122	2064	0.00	59	139	2064	0.00	62	122	2064	0.00	59	139	2064	0.00	62	122
$E50 \times 07 - 6$	2185	2186	2185	0.05	57	150	2186	0.05	57	183	2186	0.05	64	137	2186	0.05	57	183	2186	0.05	64	137	2186	0.05	57	183	2186	0.05	64	137
$E50 \times 07 - 7$	1475	1476	1475	0.07	58	141	1475	0.00	60	175	1475	0.00	61	143	1475	0.00	60	175	1475	0.00	61	143	1475	0.00	60	175	1475	0.00	61	143
$E50 \times 07 - 8$	2224	2249	2224	1.12	57	133	2238	0.63	59	173	2238	0.63	60	154	2249	1.12	57	173	2238	0.63	60	154	2249	1.12	57	173	2238	0.63	60	154
$E50 \times 07 - 9$	1575	1600	1575	1.59	58	180	1580	0.32	60	159	1582	0.44	61	127	1600	1.59	58	159	1582	0.44	61	127	1600	1.59	58	159	1582	0.44	61	127
$E50 \times 07 - 10$	1643	1644	1643	0.06	57	170	1645	0.12	60	125	1648	0.30	60	138	1644	0.06	57	125	1645	0.12	60	138	1644	0.06	57	125	1645	0.12	60	138

Table 6: Detailed results for GRASP with  $\alpha = 50\%$ ,  $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances E50x07.

Instances	GRASP_NEH <sub>sep</sub> $\alpha = 50\%$						GRASP_NEH <sub>sep</sub> $\alpha = 75\%$						GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)								
	LB	UB	g(%)	t(s)	it	it	UB	g(%)	t(s)	it	it	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it		
	$E50 \times 11 - 1$	2437	2548	4.55	54	183	2582	5.95	64	150	2518	3.32	49	172	2664	2.62	54	164	2606	0.62	58
$E50 \times 11 - 2$	2596	2760	6.32	59	189	2624	1.08	62	181	2736	1.15	61	176	2700	3.33	60	185	3358	5.00	63	171
$E50 \times 11 - 3$	2590	2662	2.78	59	153	2778	7.26	60	153	2556	0.08	60	183	2514	4.23	60	187	2514	4.23	63	164
$E50 \times 11 - 4$	2705	2736	1.15	59	162	2736	1.15	61	162	3022	1.27	65	175	3022	1.27	62	167	3022	1.27	62	167
$E50 \times 11 - 5$	2613	2802	7.23	56	184	2800	7.16	61	176	3208	4.60	59	189	3150	2.71	63	128	3150	2.71	63	128
$E50 \times 11 - 6$	3198	3225	0.84	57	156	3456	8.07	63	183	3208	4.60	59	189	3150	2.71	63	128	3208	4.60	59	189
$E50 \times 11 - 7$	2554	2557	0.12	57	152	2556	0.08	60	183	2556	0.08	60	183	2556	0.08	66	162	2556	0.08	66	162
$E50 \times 11 - 8$	2412	2549	5.68	56	155	2514	4.23	60	187	2514	4.23	63	164	2514	4.23	63	164	2514	4.23	63	164
$E50 \times 11 - 9$	2984	3060	2.55	58	164	3022	1.27	65	175	3022	1.27	62	167	3022	1.27	62	167	3022	1.27	62	167
$E50 \times 11 - 10$	3067	3150	2.71	58	167	3208	4.60	59	189	3150	2.71	63	128	3150	2.71	63	128	3150	2.71	63	128

Table 7: Detailed results for GRASP with  $\alpha = 50\%$ ,  $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances  $E50 \times 11$ .

Instances	GRASP_NEH <sub>sep</sub> $\alpha = 50\%$				GRASP_NEH <sub>sep</sub> $\alpha = 75\%$				GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)				
	LB	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it
	$D_{20} \times 03 \times 11 - 1$	726	768	5.78	46	144	734	1.10	46	131	738	1.65	56
$D_{20} \times 03 \times 11 - 2$	1122	1125	0.27	46	156	1124	0.18	45	139	1124	0.18	57	119
$D_{20} \times 03 \times 11 - 3$	1516	1522	0.40	48	145	1518	0.13	48	118	1516	0.00	51	153
$D_{20} \times 03 \times 11 - 4$	836	844	0.96	56	150	837	0.12	44	122	836	0.00	51	148
$D_{20} \times 03 \times 11 - 5$	1171	1182	0.94	51	118	1190	1.62	48	155	1224	4.53	51	137
$D_{20} \times 03 \times 11 - 6$	775	785	1.29	51	126	776	0.13	48	134	775	0.00	49	139
$D_{20} \times 03 \times 11 - 7$	804	840	4.48	50	131	814	1.24	45	140	816	1.49	49	136
$D_{20} \times 03 \times 11 - 8$	813	826	1.60	55	152	813	0.00	46	137	813	0.00	52	128
$D_{20} \times 03 \times 11 - 9$	798	800	0.25	53	154	800	0.25	49	146	808	1.25	51	156
$D_{20} \times 03 \times 11 - 10$	1177	1210	2.80	48	123	1194	1.44	47	151	1204	2.29	57	133

Table 8: Detailed results for GRASP with  $\alpha = 50\%$ ,  $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances  $D_{20} \times 03 \times 11$ .

Instances	GRASP_NEH <sub>sep</sub> $\alpha = 50\%$				GRASP_NEH <sub>sep</sub> $\alpha = 75\%$				GRASP_NEH <sub>sep</sub> $\alpha = 100\%$ (Random Multistart)				
	LB	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it	UB	g(%)	t(s)	it
	$D_{20 \times 05 \times 11 - 1}$	1071	1083	1.12	54	126	1080	0.84	47	150	1078	0.65	53
$D_{20 \times 05 \times 11 - 2}$	1246	1259	1.04	53	160	1250	0.32	50	143	1255	0.72	51	162
$D_{20 \times 05 \times 11 - 3}$	1226	1255	2.36	48	137	1250	1.96	54	172	1244	1.47	54	153
$D_{20 \times 05 \times 11 - 4}$	1293	1296	0.23	53	134	1294	0.08	47	136	1293	0.00	51	168
$D_{20 \times 05 \times 11 - 5}$	1214	1222	0.66	52	142	1216	0.16	47	144	1214	0.00	52	175
$D_{20 \times 05 \times 11 - 6}$	1304	1330	1.99	54	135	1308	0.31	47	134	1308	0.31	51	166
$D_{20 \times 05 \times 11 - 7}$	1388	1392	0.29	48	153	1389	0.07	47	129	1388	0.00	56	174
$D_{20 \times 05 \times 11 - 8}$	1639	1674	2.13	52	149	1644	0.30	46	137	1662	1.40	53	164
$D_{20 \times 05 \times 11 - 9}$	1215	1240	2.06	47	155	1240	2.06	49	156	1238	1.89	50	159
$D_{20 \times 05 \times 11 - 10}$	1562	1568	0.38	54	162	1563	0.06	47	163	1562	0.00	52	164

Table 9: Detailed results for GRASP with  $\alpha = 50\%$ ,  $\alpha = 75\%$ , and  $\alpha = 100\%$  on instances  $D_{20 \times 05 \times 11}$ .

**Acknowledgments:** This research was partially supported by Brazilian funding agencies CAPES and CNPq.

## References

- 535 [1] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly* 1 (1954) 61–68.
- [2] J. N. D. Gupta, E. F. Stafford, Flowshop scheduling research after five decades, *European Journal of Operational Research* 169 (2006) 699–711.
- [3] M. R. Garey, D. S. Johnson, R. Sethi, The complexity of flowshop and  
540 jobshop scheduling, *Mathematics of Operations Research* 1 (1976) 117–129.
- [4] F. T. Tseng, E. F. Stafford Jr., J. N. D. Gupta, An empirical analysis of integer programming formulations for the permutation flowshop, *Omega* 32 (2004) 285–293.
- [5] J. V. Frasc, S. O. Krumke, S. Westphal, MIP formulations for flowshop scheduling with limited buffers, *Theory and Practice of Algorithms in*  
545 *(Computer) Systems* 6595 (2011) 127–138.
- [6] B. Naderi, M. Aminnayeri, M. Piri, M. H. H. Yazdi, Multi-objective no-wait flowshop scheduling problems: models and algorithms, *International Journal of Production Research* 50 (10) (2012) 2592–2608. doi:10.1080/00207543.2010.543937.  
550
- [7] D. P. Ronconi, E. G. Birgin, Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness, in: R. Z. Ríos-Mercado, Y. A. Ríos-Solís (Eds.), *Just-in-Time Systems*, Vol. 60, Springer, 2012, pp. 91–105.
- 555 [8] F. Hnaien, F. Yalaoui, A. Mhadhbi, Makespan minimization on a two-machine flowshop with an availability constraint on the first machine, *International Journal of Production Economics* 164 (2015) 95–104.

- [9] G. Mainieri, D. Ronconi, New heuristics for total tardiness minimization in a flexible flowshop, *Optimization Letters* 7 (2013) 665–684.
- 560 [10] M. Nawaz, E. E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega* 11 (1983) 91–95.
- [11] S. F. Rad, R. Ruiz, N. Boroojerdian, New high performing heuristics for minimizing makespan in permutation flowshops, *Omega* 37 (2009) 331–345.
- [12] M. Widmer, A. Hertz, A new heuristic method for the flow shop sequencing  
565 problem, *European Journal of Operational Research* 41 (1989) 186–193.
- [13] A. Allahverdi, H. Aydilek, Heuristics for the two-machine flowshop scheduling problem to minimise makespan with bounded processing times, *International Journal of Production Research* 48 (2010) 6367–6385.
- [14] Q.-K. Pan, L. Wang, L. Gao, W. Li, An effective hybrid discrete differential  
570 evolution algorithm for the flow shop scheduling with intermediate buffers, *Information Sciences* 181 (2011) 668–685.
- [15] H. Allaoui, A. Artiba, Scheduling two-stage hybrid flow shop with availability constraints, *Computers & Operations Research* 33 (2006) 1399–1419.
- [16] V. Fernandez-Viagas, J. Framinan, Neh-based heuristics for the permutation  
575 flowshop scheduling problem to minimise total tardiness, *Computers & Operations Research* 60 (2015) 27–36.
- [17] C. Low, J. Yeh, K. Huang, A robust simulated annealing heuristic for flow shop scheduling problems, *The International Journal of Advanced Manufacturing Technology* 23 (2004) 762–767.
- 580 [18] A. C. Nearchou, Flow-shop sequencing using hybrid simulated annealing, *Journal of Intelligent manufacturing* 15 (2004) 317–328.
- [19] H. Mirsanei, M. Zandieh, M. J. Moayed, M. Khabbazi, A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-

- dependent setup times, *Journal of Intelligent Manufacturing* 22 (2011) 965–  
585 978.
- [20] B. Santosa, A. Rofiq, The development of simulated annealing algorithm for hybrid flow shop scheduling problem to minimize makespan and total tardiness, in: *Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management*, 2014, pp. 1348–1355.
- 590 [21] G. Prabhakaran, B. S. H. Khan, L. Rakesh, Implementation of GRASP in flow shop scheduling, *The International Journal of Advanced Manufacturing Technology* 30 (2006) 1126–1131.
- [22] B. Shahul Hamid Khan, G. Prabhakaran, P. Asokan, A grasp algorithm for m-machine flowshop scheduling problem with bicriteria of makespan and  
595 maximum tardiness, *International Journal of Computer Mathematics* 84 (2007) 1731–1741.
- [23] K. Chakravarthy, C. Rajendran, A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization, *Production Planning & Control* 10 (1999) 707–714.
- 600 [24] P. Sivasankaran, T. Sornakumar, R. Panneerselvam, Design and comparison of simulated annealing algorithm and grasp to minimize makespan in single machine scheduling with unrelated parallel machines, *Intelligent Information Management* 2 (2010) 406–416.
- [25] The distributed permutation flowshop scheduling problem, *Computers & Operations Research* 37 (4) (2010) 754 – 768. doi:<https://doi.org/10.1016/j.cor.2009.06.019>.  
605
- [26] A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion, *Computers & Industrial Engineering* 111 (2017) 239 – 250. doi:<https://doi.org/10.1016/j.cie.2017.07.020>.  
610

- [27] Iterated greedy methods for the distributed permutation flowshop scheduling problem, *Omega* 83 (2019) 213 – 222. doi:<https://doi.org/10.1016/j.omega.2018.03.004>.
- [28] S. Hatami, R. Ruiz, C. Andrés Romano, Two simple constructive algorithms for the distributed assembly permutation flowshop scheduling problem, in: C. Hernández, A. López-Paredes, J. M. Pérez-Ríos (Eds.), *Managing Complexity*, Springer International Publishing, Cham, 2014, pp. 139–145.
- [29] X. Li, X. Zhang, M. Yin, J. Wang, A genetic algorithm for the distributed assembly permutation flowshop scheduling problem, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 3096–3101. doi:[10.1109/CEC.2015.7257275](https://doi.org/10.1109/CEC.2015.7257275).
- [30] Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times, *International Journal of Production Economics* 169 (2015) 76 – 88. doi:<https://doi.org/10.1016/j.ijpe.2015.07.027>.
- [31] An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem, *Computers & Industrial Engineering* 97 (2016) 128 – 136. doi:<https://doi.org/10.1016/j.cie.2016.05.005>.
- [32] Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion, *Swarm and Evolutionary Computation* 44 (2019) 64 – 73. doi:<https://doi.org/10.1016/j.swevo.2018.12.001>.
- [33] Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures, *European Journal of Operational Research* 273 (2) (2019) 401 – 417. doi:<https://doi.org/10.1016/j.ejor.2018.04.033>.

- [34] I. F. Guimarães, Y. Ouazene, M. C. de Souza, F. Yalaoui, Semi-parallel  
640 flow shop with a final synchronization operation scheduling problem, IFAC-  
PapersOnLine 49 (2016) 1032–1037.
- [35] H. M. Wagner, An integer linear-programming model for machine schedul-  
ing, Naval Research Logistics Quarterly 6 (1959) 131–140.
- [36] E. F. Stafford, On the development of a mixed-integer linear programming  
645 model for the flowshop sequencing problem, Journal of the Operational  
Research Society 39 (1988) 1163–1174.
- [37] F. T. Tseng, E. F. Stafford, Two milp models for the  $n \times m$  sdst flow-  
shop sequencing problem, International Journal of Production Research 39  
(2001) 1777 – 809.
- 650 [38] J. C. Arroyo, A. A. de Souza Pereira, A grasp heuristic for the multi-  
objective permutation flowshop scheduling problem, The International  
Journal of Advanced Manufacturing Technology 55 (2011) 741–753.
- [39] J. Hurkała, A. Hurkała, Effective design of the simulated annealing algo-  
rithm for the flowshop problem with minimum makespan criterion, Journal  
655 of Telecommunications and Information Technology 2 (2012) 92–98.
- [40] P. Jarosław, S. Czesław, Ż. Dominik, Optimizing bicriteria flow shop  
scheduling problem by simulated annealing algorithm, Procedia Computer  
Science 18 (2013) 936–945.
- [41] M. G. C. Resende, C. C. Ribeiro, Greedy randomized adaptive search pro-  
660 cedures: Advances, hybridizations, and applications, Handbook of Meta-  
heuristics 146 (2010) 283–319.
- [42] E. Taillard, Benchmarks for basic scheduling problems, European Journal  
of Operational Research 64 (1993) 278–285.