New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics

Article *in* Expert Systems with Applications · July 2018

DOI: 10.1016/j.eswa.2018.07.055

CITATIONS
4

READS
115

3 authors:

Victor Fernandez-Viagas
Universidad de Sevilla
34 PUBLICATIONS **401** CITATIONS
SEE PROFILE

Jose M. Molina-Pariente
Universidad de Sevilla
14 PUBLICATIONS **112** CITATIONS
SEE PROFILE

Jose M. Framinan
Universidad de Sevilla
188 PUBLICATIONS **3,131** CITATIONS
SEE PROFILE

New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *

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

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

Camino de los Descubrimientos s/n, 41092 Seville, Spain, {vfernandezviagas,jmolina1,framinan}@us.es

February 4, 2019

Abstract

This paper addresses the hybrid flow shop scheduling problem to minimise makespan, a well-known scheduling problem for which many constructive heuristics have been proposed in the literature. Nevertheless, the state of the art is not clear due to partial or non homogeneous comparisons. In this paper, we review these heuristics and perform a comprehensive computational evaluation to determine which are the most efficient ones. A total of 20 heuristics are implemented and compared in this study. In addition, we propose four new heuristics for the problem. Firstly, two memory-based constructive heuristics are proposed, where a sequence is constructed by inserting jobs one by one in a partial sequence. The most promising insertions tested are kept in a list. However, in contrast to the Tabu search, these insertions are repeated in future iterations instead of forbidding them. Secondly, we propose two constructive heuristics based on Johnson's algorithm for the permutation flowshop scheduling problem. The computational results carried out on an extensive testbed show that the new proposals outperform the existing heuristics.

Keywords: Scheduling, hybrid Flowshop, Heuristics, Makespan, Computational evaluation, HFS, memory-based constructive heuristics

*Preprint submitted to Computers & Industrial Engineering. DOI:10.1016/j.eswa.2018.07.055

†Corresponding author. Email: vfernandezviagas@us.es

1 Introduction

The flowshop scheduling problem is one of the most active research areas within Operations Research (see e.g. Framinan et al., 2004; Ruiz and Maroto, 2005; Fernandez-Viagas et al., 2017 for reviews on the topic). In the flowshop layout, n jobs have to be processed on m stages, each one composed of a single machine, following each job the same route of stages. The problem then consists in obtaining the best sequence of jobs in each machine according to a certain objective (typically the minimisation of makespan, see e.g. Fernandez-Viagas and Framinan, 2014, or the total completion times, see e.g. Fernandez-Viagas and Framinan, 2017a). However, in many manufacturing scenarios several machines in parallel are used to perform an operation as it serves to increase the capacity and/or throughput; to balance the use of the stages; and to decrease the influence of the bottleneck machine (Naderi et al., 2010). This flowshop problem with parallel machines in each stage is usually denoted as the Hybrid Flowshop Scheduling (HFS) problem or flexible flowshop scheduling problem. In this paper we address the HFS with the objective of makespan minimisation which is known to aim at minimising production run and maximising machine utilisation. The problem can be denoted as $HFm||C_{max}$ or $FFm||C_{max}$ following Graham et al. (1979) and, alternatively, by $FHm,((PM^k)_{k=1}^m)||C_{max}$ following Ruiz and Vázquez-Rodríguez (2010).

Since the problem under consideration is known to be NP-hard by Gupta, 1988 (for the problem even when there are two stages: one with two machines and the other one with a single machine), and by Rinnooy Kan, 1976 (for the problem with a single stage with more than two machines), many approximated algorithms have been developed in the literature (see in this regard the reviews by Ruiz and Vázquez-Rodríguez, 2010; Ribas et al., 2010 and e.g. Dios et al., 2018; Chung et al., 2017; Zhong and Shi, 2018; Ying and Lin, 2018). In these reviews, most contributions focus on the HFS with identical parallel machines and the maximum completion time or makespan (denoted as C_{max}) as objective, which is also the problem under consideration here. Despite the different heuristics proposed for the problem (see Section 2), we are not aware of any computational evaluation comparing all of them under the same conditions, and only partial comparisons have been performed in the existing literature, using a small subset of heuristics

and/or different sets of instances for each comparison.

Among the heuristics proposed for the problem, the NEH (originally proposed by Nawaz et al., 1983 for flowshop scheduling and adapted for our problem by Brah and Loo, 1999) seems to be, up to now, one of the best heuristics for the problem, due to its extensive use as initial solution of metaheuristics or as a reference procedure for other constructive heuristics. Despite the excellent performance of the NEH for a range of scheduling problems, recent research has shown different strategies to enhance it: On the one hand, the use of the original objective function of the problem to select the best partial sequence in a heuristic must not necessary imply the best decision in an iteration of the algorithm (see e.g. Dong et al., 2008; Fernandez-Viagas and Framinan, 2015b, where tie-breaking mechanisms based on idle times are included in the evaluation of partial sequences to improve the solutions in related problems); On the other hand, Fernandez-Viagas and Framinan (2017b) found that, under certain conditions, some stages could be ignored in the traditional flowshop, being approximately equivalent to a single-machine scheduling problem. Note that both reasonings could be also applied to the problem under consideration.

To tackle these challenges, our contribution to the problem is twofold: Firstly, an exhaustive computational evaluation of the heuristics available for the problem is performed. Secondly, we propose four new efficient (memory-based and Johnson-based) constructive heuristics that take into account the aforementioned ideas and that our experiments show that they outperform the existing ones. The first two heuristics construct a solution step by step in a greedy manner, but also taking into consideration the most promising partial solutions obtained in the previous iteration. The last two heuristics reduce the problem to different two-machine flowshop scheduling problems and use the Johnson's algorithm (Johnson, 1954) to solve them exactly. The remainder of the paper is organised as follows: In Section 2, the problem under consideration is formally described and its background is discussed. The constructive heuristics proposed are described in Section 3. The computational evaluation of both existing and new heuristics is presented in Section 4. Finally, the conclusions are discussed in Section 5.

2 Problem description and background

The problem under study can be defined as follows. There is a set $\mathcal{N}$ of n jobs that have to be processed on a set $\mathcal{M}$ of m stages. Each stage i ($\forall i \in \{1, \ldots, m\}$) is composed of m_i identical machines. Each job has to be processed on only one machine in each stage, all jobs following the same order of stages. The processing time of job j in stage i is denoted by p_{ij}. The problem then consists in determining, for each stage, both the machines where each job is to be processed and the order of jobs to process for each machine in order to minimise the maximum completion time or makespan (C_{max}). In addition, the following hypotheses are also adopted: each machine processes at most one job at the same time, and each job is available at initial time; setup times are considered as sequence-independent and non-anticipatory, and they can hence be included in the processing times of the jobs; finally, unlimited inventory is considered between stages.

Note that many approximated algorithms have been proposed to solve the problem in the existing literature, as already mentioned in Section 1 (see Ruiz and Vázquez-Rodríguez, 2010; Ribas et al., 2010 for a more detailed review and explanation of all these approaches). Approximate algorithms can be classified in heuristics and metaheuristics (Framinan et al., 2005 and Ruiz and Maroto, 2005). While heuristics (constructive and improvement) typically obtain a fast solution using a fixed number of iterations, metaheuristics are typically forced to stop after a fixed CPU time or number of iterations. In this section, we focus in studies proposing constructive or improvement heuristics for the $HFm||C_{max}$ problem. In addition, metaheuristics typically require initial solutions obtained using constructive/improvement heuristics. Therefore, we also review the existing metaheuristics for the problem in order to identify additional constructive/improvement heuristics.

Lee and Vairaktarakis (1994) solve the two-stage HFS problem to minimise makespan by using a simple heuristic that assigns jobs to the first stage with the First Available Machine rule (FAM), i.e. each job in a sequence is assigned to the first machine which becomes available. In the second stage, a mirror image of the FAM rule, named Last Busy Machine rule (LBM), is developed to assign the jobs. Koulamas and Kyparisis (2000) propose three linear time heuristics to solve the two- and three-stage case. More specifically, the H_L heuristic solves the two-stage

case and two heuristics (denoted as H_0 and H_S) solve the three-stage case. Several heuristics are also proposed by Soewandi and Elmaghraby (2001) to solve the three-stage problem.

For the m-stage case, Santos et al. (1996) adapt four heuristics by Campbell et al. (1970), Palmer (1965), Gupta (1971), and Dannenbring (1977), originally developed for the permutation flowshop to minimise makespan. In their experiments, the proposals by Campbell et al. (1970) (denoted as CDS1) and Dannenbring (1977) (denoted as DNN) outperform the heuristics by Palmer (1965), and Gupta (1971). Brah and Loo (1999) compare the CDS1 heuristic against four other heuristics originally proposed for the permutation flowshop problem by Nawaz et al. (1983), Hundal and Rajgopal (1988), Park et al. (1984), and Ho (1995). All heuristics were adapted to the hybrid flowshop to minimise makespan and other objectives. The most promising heuristic regarding the makespan are the heuristic by Nawaz et al., 1983 (denoted as NEH), the CDS1 heuristic, and the adaptation of CDS2 (originally proposed by Park et al., 1984). Acero-Dominguez and Paternina-Arboleda (2004); Paternina-Arboleda et al. (2008) propose a heuristic, denoted as BH, based in the bottleneck concept according to the theory of constraints (Goldratt and Cox (1992)). They compare their proposal against the traditional shifting bottleneck heuristic (proposed by Adams et al., 1988 to solve a job shop layout) and against the hybrid shifting bottleneck-local search (proposed by Pinedo and Chao, 1999).

Regarding metaheuristics, Alaykýran et al. (2007) and Engin and Döyen (2004) propose an Artificial Immune System and an Ant Colony Optimisation, respectively. With respect to the generation of initial solutions for these algorithms, the former uses a random population based on the idle times between the jobs, while in the latter the method to obtain the initial population is not described. Their algorithms outperform a B&B by Néron et al. (2001) (using a maximum CPU time) on the set of instances proposed by Carlier and Néron (2000) (note that, although this benchmark only considers a maximum number of jobs equal to 15, it is the most used benchmark so far for the problem under consideration). Negenman (2001) adapts several local search algorithms from the flow shop and job shop literature, and compares them against a variable-depth search and a Simulated Annealing. No indication of the initial solution is detailed. Niu et al. (2009) propose a quantum algorithm using an initial population randomly generated (see e.g. Norman and Bean, 1999; Kurz and Askin, 2004). Liao et al. (2012) propose a hybrid Particle

Swarm Optimization (PSO) using the BH heuristic to obtain the initial solution. Their results have been compared using the benchmark from Carlier and Néron (2000) against some existing metaheuristics (Carlier and Néron, 2000; Néron et al., 2001; Engin and Döyen, 2004; Niu et al., 2009; Alaykýran et al., 2007). A simple iterated greedy algorithm and two different constructive heuristics, denoted as WT1(x) and WT2(x), are proposed by Kizilay et al. (2015). Regarding WT1(x) and WT2(x), they found that both outperform NEH. However, the comparison is carried out using different computational CPU times. An Estimation of Distribution Algorithm using a random population is proposed by Wang et al. (2013). This algorithm outperforms the proposals by Liao et al. (2012), and Engin and Döyen (2004). Pan et al. (2014) develop a Discrete Artificial Bee Colony Algorithm (DABC). In addition, they propose several dispatching rules and NEH-based heuristics to solve the problem. These heuristics reduce the number of positions where each job has to be inserted. Among them, the best results are found by the dispatching rules SPTB (jobs ordered according to non-decreasing processing times till bottleneck stage) and bLPTB (jobs ordered in a backward manner according to non-decreasing processing times till bottleneck stage), and by the NEH-based heuristics $\text{NEH}_{\text{LPT}}(\lambda)$, and $\text{bNEH}_{\text{SPT}}(\lambda)$.

Regarding contributions in related (more constrained) problems, Barman (1997) proposes several dispatching rules for the three-stage problem considering release dates, and minimising total flow time, total tardiness or number of tardy jobs. Each rule uses a different method to construct the sequence in each stage. The best result is found for the total flow time by chosing the jobs in the queue between stages according to the SPT rule, denoted as FIFO(SPT). Jayamohan and Rajendran (2000) extend the experimentation performed by Barman (1997) to other objectives (maximum flowtime, variance of flowtime, maximum tardiness and variance of tardiness) and considering new dispatching rules. The PT+WINQ+AT rule (Holthaus and Rajendran, 1997), denoted as FIFO(PT+WINQ+AT), obtains the best result for the maximum flowtime minimisation.

To summarise, different heuristics have been proposed to solve the problem under study during the last twenty years. Nevertheless, the state-of-the art regarding heuristic methods is nowadays unclear due to the following issues:

1. Despite the good results obtained by Santos et al. (1996), and Brah and Loo (1999), to

the best of our knowledge, their heuristics have not been used and/or compared against new proposals, so their actual performance remains unclear.

2. Several heuristics have been proposed in the literature during the last years (see e.g. Acero-Dominguez and Paternina-Arboleda, 2004; Paternina-Arboleda et al., 2008; Pan et al., 2014) to solve the problem either to obtain fast solutions or to be used as seed sequences of more complex algorithms. However, to the best of our knowledge, a direct comparison between them does not exist.

3. In most of the existing heuristics, jobs are sequenced between stages according to simple rules, such as the FIFO rule (jobs ordered according to non decreasing completion times in the previous stage). However, the results by Barman (1997), Holthaus and Rajendran (1997) and Jayamohan and Rajendran (2000) suggest that other rules could provide good solutions for the problem. Despite their potential, such rules have not been incorporated so far in the existing algorithms.

4. Recent advances in constructive greedy heuristics have shown the potential of using tailored indicators to improve the myopic nature of these heuristics (Fernandez-Viagas et al., 2017).

5. Finally, it can be seen that additional heuristics could be developed by further exploring the division of the original problem in subsets. Particularly, the use of heuristics for the 2-machine flowshop (i.e. Johnson) has not been sufficiently analysed: despite being a straightforward adaptation from the flowshop layout, the CDS2 heuristic provides very good results, and it could be further refined by considering the number of machines on each stage.

As a result, both a computational evaluation of heuristics and new efficient approaches are pertinent. In Section 3 we propose several heuristics based in the aforementioned considerations, while in Section 4 the computational evaluation is carried out.

3 Proposed constructive heuristics

In this section, two memory-based constructive heuristics (see Subsection 3.1) and two Johnson-based constructive heuristics (see Subsection 3.2) are proposed to find fast solutions for the problem. All these heuristics use a unique sequence to represent a solution for the problem. This sequence indicates the order in which the jobs are processed in the first stage. The FIFO rule is used to extend the solution to other stages, i.e. the sequence of jobs in stage i (with $i > 1$) is obtained by ordering the jobs according to non-decreasing completion times in stage $i - 1$. Furthermore, within each stage, jobs are processed in the machines according to the FAM rule, i.e. the first job of the sequence in a stage is assigned to the first machine that becomes available and so forth.

3.1 Memory-based constructive heuristics

Two memory-based constructive heuristics are proposed in this section: a Fast Memory-based Constructive Heuristic (denoted as FMCH), and a Memory-based Constructive Heuristic (denoted as MCH). Both heuristics construct a complete sequence by inserting, one by one, a job in the best position of a partial sequence, following a mechanism similar to that in the NEH heuristic. However, these heuristics use an objective function that combines the minimisation of makespan and idle time to evaluate the partial sequences. As explained in Section 1, the evaluation of these partial sequences substantially influences the efficiency of this type of heuristics. Although the final objective is minimising the makespan, using solely the minimisation of this criterion to select the best partial sequence may result in a wrong choice in the first iterations of the algorithm when the partial sequences are composed of very few jobs. In addition, a memory mechanism is incorporated in our proposals. This mechanism works exactly in the opposite way than the tabu list in the tabu search. Thus, the idea behind this mechanism is to keep the most promising moves to test them in future iterations.

Regarding FMCH, in the related permutation flowshop scheduling problem ($Fm|prmu|C_{max}$), Kalczynski and Kamburowski (2007, 2008, 2009) obtained very good results by inserting each job in the position with the lowest makespan, and in case of ties, in

either the first or the last position (among those with the lowest makespan). These positions are chosen based on ideas taken from Johnson's algorithm (Johnson, 1954). In addition, Fernandez-Viagas and Framinan (2014) also found promising results by selecting the partial sequence with the lowest idle time among the sequences with the lowest makespan. However, in the aforementioned heuristics, the relative position of a job already inserted in a previous iteration remains unchanged. The idea behind our proposal is not to discard other promising insertions that have the same value of the objective function in future iterations. More specifically, let $\Gamma := (\gamma_1, \ldots, \gamma_n)$ be a sequence where the jobs are sorted according to the non-increasing sum of their processing times. In iteration k, γ_k job is tested in each position of a partial sequence $\Pi^{(k-1)} = (\pi_1^{(k-1)}, \ldots, \pi_{k-1}^{(k-1)})$. Let $\Pi^{(k)}$ be the sequence after the insertion of γ_k in the position with lowest C_{max} among the k tested positions. In case of ties (i.e. several positions with the same value of the best makespan), the position with the lowest value of the indicator $OF^{'} = C_{max} + I/L$ (instead of directly C_{max}) is chosen, where I is the total idle time after the insertion of γ_k and L is a big number. In addition, the last position l with lowest makespan (in case of ties) is kept to be tested in the next iteration. Let $r^{(k)}$ denote its previous job in the sequence, i.e. $r^{(k)} = \pi_{l-1}^{(k-1)}$. Finally, once the final sequence Π^n is generated, n different solutions are generated by fixing Π^n in the first stage and randomly varying the sequence in the other stages, i.e. a random job in the queue of a stage is selected when a job should be placed in any machine of that stage. The pseudocode of the proposed FMCH algorithm is shown in Figure 1.

Procedure *FMCH*

$\Gamma := \{\gamma_1, ..., \gamma_i, ..., \gamma_n\}$: Jobs ordered by non-increasing sum of the processing times;

$T^{(1)} := 1$;

$\Pi^{(1)} := \{\gamma_1\}$;

for $k = 2$ **to** n **do**

Insert job γ_k in any possible position j of $\Pi^{(k-1)}$, with $j \in \{1, \ldots, k\}$.

$T^{(k)}$:= number of sequences with the lowest value of C_{max};

$\Pi^{(k)}$:= sequence obtained by inserting γ_k in the position of $\Pi^{(k-1)}$ with lowest $OF^{'}$. Let F^b denote such value of the objective function;

if $T^{(k)} > 1$ **then**

$r^{(k)}$:= job before γ_k, after testing γ_k in the last position with the lowest value of C_{max};

end

if $T^{(k-1)} > 1$ **then**

$\Phi^{(k)}$:= permutation obtained by removing job γ_{k-1} from $\Pi^{(k)}$ and re-inserting it after job $r^{(k-1)}$. Let F^m be the value of $OF^{'}$;

if $F^m < F^b$ **then**

$F^b = F^m$;

$\Pi^{(k)} = \Phi^{(k)}$;

end

end

end

for $k = 1$ **to** n **do**

Ω:= solution obtained by considering $\Pi^{(n)}$ as the sequence in the first stage, and by generating a random sequence for the other stages as follows: Once a machine becomes free, the next job to be processed is randomly chosen among the jobs in the queue. Let F^s denote the objective function obtained;

if $F^s < F^b$ **then**

$F^b = F^s$;

end

end

end

Figure 1: Pseudocode of FMCH

Our second proposal extends the idea of using several promising insertions to be tested in

the following iterations. As compared to the previous one, this heuristic keeps several moves in a list, denoted as memory list, instead of only one. Let $S^{(k)}$ then be the size of such list in iteration k. Note that we develop a dynamic list size $S^{(k)}$ whose size increases linearly with each iteration, i.e. $S^{(k)} = k \cdot x$ where x is a constant of proportionality, as the number of tested positions in each iteration also linearly increases in the algorithm proposed (e.g. for $x = 1$ there are two jobs in the memory list in the first iteration, $k = 2$, and n in the last iteration). Let $r_s^{(k)}$ denote the job before γ_k, after testing γ_k in the sth position of Π^k with lowest OF', in iteration k. Hence, in each iteration k, the $S^{(k-1)}$ best insertions in the last iteration are again tested. More specifically, γ_{k-1} is again tested after the $S^{(k-1)}$ best jobs (i.e. $r_s^{(k-1)}$ with $s \in \{1, \ldots, (k-1) \cdot x\}$) found in the last iteration. The pseudocode of the proposed MCH algorithm is shown in Figure 2.

Procedure *MCH(x)*

$\Gamma := \{\gamma_1, ..., \gamma_i, ..., \gamma_n\}$: Jobs ordered by non-increasing sum of the processing times;

$\Pi^{(1)} := \{\gamma_1\}$;

for $k = 2$ **to** n **do**

$S^{(k)} := k \cdot x$;

Insert job γ_k in any possible position j of $\Pi^{(k-1)}$, with $j \in \{1, \ldots, k\}$;

$\Pi^{(k)}$:= sequence obtained by inserting γ_k in the position of $\Pi^{(k-1)}$ with lowest OF'. Let F^b denote such value of the objective function;

$r_s^{(k)}$ (with $s \in \{1, \ldots, S^{(k)}\}$):= job before γ_k, after testing γ_k in the sth position with lowest OF';

if $k > 2$ **then**

for $s = 1$ **to** $S^{(k-1)}$ **do**

$\Phi^{(k)}$:= permutation obtained by removing job γ_{k-1} from $\Pi^{(k)}$ and re-insert it after job $r_s^{(k-1)}$. Let F^m be the value of OF';

if $F^m < F^b$ **then**

$F^b = F^m$;

$\Pi^{(k)} = \Phi^{(k)}$;

end

end

end

end

for $k = 1$ **to** n **do**

Ω:= solution obtained by considering $\Pi^{(n)}$ as the sequence in the first stage, and by generating a random sequence for the other stages as follows: Once a machine becomes free, the next job to be processed is randomly chosen among the jobs in the queue. Let F^s denote the objective function obtained;

if $F^s < F^b$ **then**

$F^b = F^s$;

end

end

end

Figure 2: Pseudocode of MCH

3.2 Johnson-based Constructive Heuristics

As stated in Section 2, a subset of the stages in flowshop-type scheduling problems can have a big influence on the final objective function. This reasoning might explain the good results found in the literature by Santos et al. (1996), and Brah and Loo (1999) using the CDS1 and CDS2 heuristics (adapted from the permutation flowshop scheduling problem). However, these adaptations consist of simply changing the evaluation of the final sequence, and the fact that there are parallel machines in the stages is not explicitly considered. In this section, we propose two new Johnson-based heuristics, denoted as JbH1 and JbH2. The heuristics reduce the HFS in $m-1$ and $\frac{(m+1)m}{2}$ two-machine flowshop problems, respectively, which are optimally solved. This reduction is performed taking into account the workload of the stages as explained below.

Regarding JbH1, the heuristic constructs $m-1$ two-machine flowshops which are optimally solved by Johnson's algorithm. The processing times of the jobs in the first and second artificial machines of the k flowshop ($k \in \{1,\ldots,m-1\}$) are formed by considering the first and last kth machines in the shop, respectively (let $m^{'}$ and $m^{''}$ be the number of machines which form the first and second artificial machines, respectively, i.e. $m^{'} = m^{''} = k$). To form the first artificial machine, the algorithm gives a higher value to the processing times on the first machines in the shop, and the opposite in the second artificial machine. In addition, the resulting processing times are weighted by the number of machines in each stage. More specifically, the processing times of each job in the kth reduced problem ($k \in \{1,\ldots,m\}$) are $p^{'}_{1j}$ in the first machine:

$$p^{'}_{1j} = \sum_{i=1}^{k}(k+1-i+m\cdot a)/(m_i+b)p_{ij}$$

and $p^{'}_{2j}$ in the second machine:

$$p^{'}_{2j} = \sum_{i=m-k}^{m}(i-m+k+m\cdot a)/(m_i+b)p_{ij}$$

a and b are parameters of the algorithms to set more precisely the weight of the processing times of the reduced problems. A detailed pseudocode of the algorithm is shown in Figure 3.

Procedure *JbH1*

for $k = 1$ **to** $m - 1$ **do**

for $j = 1$ **to** n **do**

$p'_{1j} = \sum_{i=1}^{k} \frac{k+1-i+m\cdot a}{m_i+b} \cdot p_{ij}$;

$p'_{2j} = \sum_{i=m-k}^{m} \frac{i-m+k+m\cdot a}{m_i+b} \cdot p_{ij}$;

end

$\Pi^k :=$ permutation obtained by applying Johnson's algorithm to the reduced kth two-machine flowshop problem using p'_{ij} (with $i \in \{1, 2\}$) as the processing times. Let OF^k denote the value of the objective function of such sequence;

if $k = 1$ **then**

$OF^b := OF^k$;

else if $OF^k < OF^b$ **then**

$OF^b := OF^k$;

end

end

end

Figure 3: Pseudocode of JbH1

In the heuristic JbH2, the number of machines which compose the first and second artificial machines changes as respect to JbH1. Thereby, an index k_1 is introduced to indicate the number of machines ($m' = k_1$) grouped to form the first artificial machine (i.e. machines $i = 1 \dots k_1$ are used). In a similar manner, an index k_2 for the second artificial machine (in this case, machines $i = k_2 \dots m$ are used to construct the second machine of the reduced problem). Let $m'' = m - k_2 + 1$ be such number of machines. Note that a different number of machines may be used to form both artificial machines of the reduced two-machine problem, and therefore we must normalize the processing times in both artificial machines, which are constructed as follows:

$$p'_{1j} = \sum_{i=1}^{k_1} \frac{(\max\{m', m''\} - i + 1) \cdot m''/m' + m \cdot c}{m_i + d} \cdot p_{ij}$$

$$p'_{2j} = \sum_{i=k_2}^{m} \frac{\max\{m', m''\} - m + i + m \cdot c}{m_i + d} \cdot p_{ij}$$

To normalize the processing times in a machine i, they are multiplied at most by $\max\{m^{'}; m^{''}\}$ (to have the same maximum reference in both artificial machines). In addition, the processing times which form the first reduced machine are normalised by $m^{''}/m^{'}$ if $m^{''} \neq m^{'}$ (this weight is introduced to balance the case where the number of machines considered in the second artificial machine is either lower or greater than in the first one). c and d are again parameters of the algorithm which are introduced to balance the expressions.

Procedure *JbH2*

- **for** $k_1 = 1$ **to** m **do**
  - **for** $k_2 = k_1$ **to** m **do**
    - $m^{'} = k_1$;
    - $m^{''} = m - k_2 + 1$;
    - max=$\max\{m^{'}; m^{''}\}$
    - **for** $j = 1$ **to** n **do**
      - $p^{'}_{1j} = \sum_{i=1}^{k_1} \frac{(\max\{m^{'};m^{''}\}-i+1)\cdot m^{''}/m^{'}+m\cdot a}{m_i+b} \cdot p_{ij}$;
      - $p^{'}_{2j} = \sum_{i=k_2}^{m} \frac{\max\{m^{'};m^{''}\}-m+i+m\cdot a}{m_i+b} \cdot p_{ij}$;
    - **end**
    - Π := permutation obtained by Johnson's algorithm in the reduced two-machine flowshop problem using $p^{'}_{ij}$ (with $i \in \{1, 2\}$) as the processing times. Let OF denote the value of the objective function of such sequence;
    - **if** $k_1 = 1 \& k_2 = 1$ **then**
      - $OF^b := OF$;
    - **else if** $OF < OF^b$ **then**
      - $OF^b := OF$;
    - **end**
  - **end**
- **end**

end

Figure 4: Pseudocode of JbH2

4 Computational evaluation

In this section, a computational evaluation is carried out to compare the different heuristics. Subsection 4.1 explains the sets of instances generated for the comparisons. The indicators to assess the algorithms are detailed in Subsection 4.2. In Subsection 4.3, an experimental parameter tuning is performed to find the best values of the parameters for JbH1 and JbH2. The heuristics implemented are listed in Subsection 4.4. Finally, the results of the computational evaluation are presented in Subsections 4.5.

4.1 Testbeds

In the problem under study, the parameters n, m, m_i, and p_{ij} must be completely defined for each instance. Several approaches and instances have been employed in the literature to test approximate algorithms. However, most of them use small instances (see e.g. Carlier and Néron, 2000; Liao et al., 2012) and/or very different values for the parameters. In this paper, a common set of big instances (denoted as β_1) is generated as follows:

- Number of jobs, n: $n \in \{20, 50, 100, 150\}$ (see e.g. Naderi et al., 2009; Pan et al., 2014 for similar ranges).
- Number of stages, m: $m \in \{5, 10, 20\}$ (taken from Taillard, 1993).
- Number of machines per stage, m_i: three different procedures have been applied to generate the set of instances using parameter $s \in \{0, 1, 2\}$. $s = 0$ generates instances with 2 machines in a unique stage (randomly generated) and 3 machines in the rest (see e.g. Carlier and Néron, 2000; Kouvelis and Vairaktarakis, 1998 for similar approaches); $s = 1$ generates instances with the same number of machines, $m_i = 3\ \forall i$, in each stage (see e.g. Naderi et al., 2009); finally, $s = 2$ generates machines using a uniform distribution [1,3]. Note that the use of uniform distribution is the most common approach to generate the machines, see e.g. Liao et al., 2012; Pan et al., 2014; Dios et al., 2018).
- Processing times, p_{ij}: all instances are generated using a uniform distribution [1,99] (see e.g. Naderi et al., 2009; Pan et al., 2014).

- Number of replications. 10 instances have been generated for each combination of n, m, and s, which results in a total of 360 instances.

In addition, a different testbed of 360 instances, denoted by β_2, is generated following the same procedure as above and is used to obtain the best values for the parameters of the algorithms JbH1 and JbH2, in order to avoid an overcalibration of these algorithms.

4.2 Performance indicators

In this paper, a total of 20 heuristics are compared under the same conditions. Since each algorithm obtains a value of the objective function typically requiring a different CPU time in each instance, in order to have a fair comparison, the Average Relative Percentage Deviation (ARPD), and Average Computational CPU Times (ACT) have been computed:

$$ARPD_h = \sum_{i=1}^{I} \frac{RPD_{ih}}{I}, \forall h = 1, \ldots, H$$

$$ACT_h = \sum_{i=1}^{I} \frac{T_{ih}}{I}, \forall h = 1, \ldots, H$$

where H is the number of algorithms under comparison ($h \in \{1, \ldots, H\}$), and I is the number of instances ($i \in \{1, \ldots, I\}$). RPD_{ih} is defined by:

$$RPD_{ih} = \frac{OF_{ih} - Best_i}{Best_i} \cdot 100, \forall h = 1, \ldots, H$$

where OF_{ih} is the C_{max} obtained by algorithm h in instance i, and $Best_i$ is the best value found in instance i, i.e. $\min_{\forall h} OF_{ih}$.

In addition, according to Fernandez-Viagas and Framinan (2015a); Fernandez-Viagas et al. (2017), the computational effort of heuristics should also be evaluated by means of $ARPT_h$ to avoid an over-representation of the largest instances of the indicators. $ARPT_h$ is defined as:

$$ARPT_h = 1 + \sum_{i=1}^{I} \frac{RPT_{ih}}{I}, \forall h = 1, \ldots, H$$

where RPT is the relative percentage computation time, defined by:

$$RPT_{ih} = \frac{T_{ih} - ACT_i}{ACT_i}, \forall h = 1, \ldots, H$$

with

$$ACT_i = \sum_{h=1}^{H} \frac{T_{ih}}{I}, \forall i = 1, \ldots, I$$

and where T_{ih} is the CPU time (in seconds) of algorithm h for instance i.

Finally, for the calibration of the Johnson-based heuristics, a slightly different measure of the quality of the solution is used, denoted as $ARPD^{'}$ (see e.g. Fernandez-Viagas et al., 2016):

$$ARPD^{'} = \sum_{i=1}^{I} \frac{(OF_i - Base_i/Base_i}{I}$$

where $Base_i$ is the solution obtained in instance i by a reference algorithm (NEH).

4.3 Experimental parameter tuning

Two full factorial designs of experiments are performed to find the best values of the parameters a and b for JbH1, and c and d for JbH2. The following range of values for the parameters have been chosen in the experimentation:

$$a = \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$$

$$b = \{0.50, 0.75, 1.00, 1.25, 1.50, 1.75\}$$

$$c = \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$$

$$d = \{0.50, 0.75, 1.00, 1.25, 1.50, 1.75\}$$

Non parametric Kruskal-Wallis tests are carried out using the $ARPD^{'}$ indicator, and the corresponding p-values obtained are 0.975, 0.846, 0.892, and 0.989 for the a, b, c, and d parameters, respectively. No statistical significant difference between the values of the parameters has been found, which show the robustness of the proposals against them. $ARPD^{'}$ values obtained for

the different levels are shown in Table 1. Results show that although there is no statistical different between the level, the average values between some levels are quite different. The best combination of the parameters is found for $a = 0.1$, $b = 1.25$, $c = 0.1$, and $b = 1.50$.

Parameter	**JbH1**	**Parameter**	**JbH2**
a=0	-0.08	c=0	-0.71
a=0.05	-0.10	c=0.05	-0.78
a=0.1	-0.12	c=0.1	-0.81
a=0.15	-0.12	c=0.15	-0.79
a=0.2	-0.12	c=0.2	-0.76
a=0.25	-0.10	c=0.25	-0.72
a=0.3	-0.04	c=0.3	-0.69
b=0.5	-0.06	d=0.5	-0.73
b=0.75	-0.10	d=0.75	-0.73
b=1	-0.10	d=1	-0.74
b=1.25	-0.11	d=1.25	-0.76
b=1.5	-0.11	d=1.5	-0.76
b=1.75	-0.10	d=1.75	-0.76

Table 1: Experimental parameter tuning

4.4 Implemented heuristics

Following the conclusions obtained in Section 2, the new heuristics proposed (FMCH, MCH, JbH1, and JbH2) are compared against the most promising heuristics for the problem under consideration (results are shown in Subsection 4.5), which are enumerated next:

- CDS1 adapted for the problem by Santos et al. (1996).
- DNN adapted for the problem by Santos et al. (1996).
- CDS2 adapted for the problem by Brah and Loo (1999).
- NEH adapted for the problem by Brah and Loo (1999).
- BH proposed by Acero-Dominguez and Paternina-Arboleda (2004); Paternina-Arboleda et al. (2008).
- SPTB proposed by Pan et al. (2014).

- bLPTB proposed by Pan et al. (2014).
- $\text{NEH}_{\text{LPT}}(\lambda)$ proposed by Pan et al. (2014).
- bNEH proposed by Pan et al. (2014).
- $\text{bNEH}_{\text{SPT}}(\lambda)$ proposed by Pan et al. (2014).
- WT1(x) $\forall x \in \{1, 5, n/m, n\}$ proposed by Kizilay et al. (2015).
- WT2(x) $\forall x \in \{1, 5, n/m, n\}$ proposed by Kizilay et al. (2015).

We also incorporate in the comparison other heuristics by changing the traditional rule (FIFO rule) to sequence the jobs between the stages in the NEH heuristic:

- NEH_{SPT}: NEH heuristic using the SPT rule when there are more than one job in the queue, as proposed by Barman (1997), i.e. when a machine becomes free, it take the next job according to the SPT rule.
- NEH_{LPT}: NEH heuristic using the LPT rule to determine the next job to be taken in the queue.
- $\text{NEH}_{\text{PT+WINQ+AT}}$: NEH heuristic using the PT+WINQ+AT, proposed by Holthaus and Rajendran (1997).
- NEH_{rand}: NEH heuristic taking the job to be sequenced randomly from the queue.

All these heuristics have been fully coded in C# using Visual Studio in an Intel Core i7-3770 with 3.4 GHz, 16 GB RAM, and with Microsoft Windows 8.1 64 bit, using the same common functions and libraries.

4.5 Comparison of heuristics

The 20 heuristics presented in Subsection 4.4 have been run on Benchmark β_1. The computational results with respect to the quality of the solutions (ARPD) are shown in Table 2, grouped by parameters n, m, and s, and the detailed results of ARPD in terms of $n \times m$ in Table 3. The

average results in terms of ARPD, ACT, and ARPT are shown in Table 4 and graphically in Figure 5. In view of the results, a number of conclusions can be noted:

1. The JbH1 and JbH2 heuristics show the best solution quality among the fast heuristics in the literature (as e.g. CDS2, BH, SPTB, and $\text{NEH}_{\text{LPT}}(\lambda)$). To support this conclusion, the following hypotheses have to be checked, hypotheses H_1: JbH1=CDS2, H_2: JbH2=CDS2 (see Table 5).

2. The JbH2 heuristic proposed ($ARPD_{JbH2} = 2.19$ and $ARPT_{JbH2} = 0.04$) improves the quality of the solutions obtained by the NEH heuristic ($ARPD_{NEH} = 3.02$ and $ARPT_{NEH} = 0.12$) using less computational effort (hypothesis H_3: JbH2=NEH).

3. In addition, the FMCH heuristic proposed ($ARPD_{FMCH} = 2.78$ and $ARPT_{FMCH} = 0.12$) also outperforms the NEH heuristic by using a similar computational effort (hypothesis H_4: FMCH=NEH).

4. Among all implemented heuristics, the best results in terms of quality of the solutions have been obtained by the proposed MCH heuristic, $ARPD_{MCH(12)} = 0.67$ (hypothesis H_5: MCH(12)=WT1(n)).

5. To summarise, the efficient heuristics for the problem are: JbH1, JbH2, MCH(1), MCH(2), MCH(4), MCH(6), MCH(8), MCH(10), and MCH(12) (the following hypotheses should be added to the previous ones: hypothesis H_6: MCH(4)=WT1(5); H_7: MCH(4)=WT2(5); H_8: JbH2=WT1(1)).

To justify the hypotheses, a Holm's procedure (Holm, 1979) is performed (see results in Table 5). In this test, the hypotheses are sorted in non-descending order of p-values, denoted by β_i ($i \in \}1, \ldots, k\}$, with k the number of hypotheses). p-values are obtained following a non-parametric Wilcoxon signed-rank test (see e.g. Fernandez-Viagas et al., 2018 for similar studies) assuming a 0.95 confidence level, i.e. $\alpha = 0.05$. Then, each hypothesis i is rejected if $p > \alpha/(k - \beta_i + 1)$. As it can be seen from Table 5, each hypothesis is rejected with the exception of hypotheses H_5 (i.e. no statistical evidence has been found between $ARPD_{MCH(12)}$ and $ARPD_{WT1(n)}$) and H_1 ($ARPD_{JbH1} = ARPD_{CDS2}$). In the former, although no differences

has been found in the ARPD, ARPT and ACT of the heuristics proposed is 1.95, and 5.61 as compared to 9.19 and 57.99 of the WT1(n) heuristic. In the latter, H_1 has been rejected according to the non-parametric Wilcoxon signed-rank test but not using Holm's procedure.

Heuristic	$n=20$	$n=50$	$n=100$	$n=150$	$m=5$	$m=10$	$m=20$	$s=0$	$s=1$	$s=2$	All
NEH	4.58	3.68	2.20	1.61	1.71	3.25	4.09	3.18	4.27	1.60	3.02
NEH_{SPT}	6.08	4.80	3.10	2.53	3.04	4.42	4.93	4.77	4.29	3.33	4.13
NEH_{LPT}	8.74	9.25	9.16	8.47	4.93	9.51	12.28	8.30	7.97	10.44	8.90
$\text{NEH}_{\text{PT+WINQ+AT}}$	4.92	3.76	2.07	1.59	1.83	3.27	4.16	3.27	4.08	1.92	3.09
NEH_{rand}	6.31	5.68	3.86	3.15	2.84	4.95	6.46	4.52	6.32	3.42	4.75
Palmer	10.35	8.17	5.07	3.78	5.53	7.08	7.91	7.03	7.53	5.96	6.84
BH	17.72	15.17	11.21	9.62	9.00	12.39	18.90	10.07	19.27	10.96	13.43
$\text{NEH}_{\text{LPT}}(\lambda)$	7.00	5.54	3.35	2.42	2.61	4.70	6.41	4.66	6.55	2.52	4.58
bNEH	6.31	3.89	2.36	1.95	3.05	3.62	4.22	3.41	4.97	2.51	3.63
$\text{bNEH}_{\text{SPT}}(\lambda)$	8.58	6.46	4.72	3.87	5.29	5.98	6.45	5.55	7.59	4.58	5.91
SPTB	15.75	12.80	8.88	8.07	10.40	11.42	12.31	9.60	14.97	9.56	11.38
bLPTB	25.68	22.77	18.88	16.21	20.00	21.92	20.74	19.28	25.06	18.32	20.88
FMCH	3.96	3.31	2.27	1.57	1.44	2.79	4.11	2.93	3.97	1.44	2.78
MCH(1)	3.37	2.63	1.34	1.03	1.23	2.13	2.91	2.30	2.82	1.16	2.09
MCH(2)	2.45	1.80	1.13	0.87	0.94	1.50	2.25	1.74	1.82	1.11	1.56
MCH(4)	1.05	1.11	0.90	0.54	0.54	0.91	1.25	1.07	0.99	0.64	0.90
MCH(6)	1.02	0.81	0.61	0.53	0.43	0.88	0.92	0.81	0.87	0.56	0.75
MCH(8)	1.05	0.75	0.60	0.51	0.53	0.76	0.90	0.83	0.89	0.47	0.73
MCH(10)	1.03	0.73	0.59	0.45	0.44	0.83	0.83	0.81	0.76	0.53	0.70
MCH(12)	1.07	0.77	0.47	0.37	0.45	0.72	0.85	0.81	0.69	0.51	0.67
WT1(1)	4.50	3.61	2.12	1.81	1.60	2.92	4.51	3.14	4.19	1.70	3.01
WT1(5)	2.31	2.06	1.18	1.10	0.74	1.61	2.64	1.66	2.42	0.91	1.66
WT1(n/m)	3.48	2.33	1.07	0.85	0.64	1.70	3.46	1.97	2.66	1.17	1.93
WT1(n)	1.23	1.07	0.54	0.39	0.30	0.82	1.29	0.74	1.34	0.34	0.81
WT2(1)	4.85	4.12	2.40	1.74	1.65	3.63	4.55	3.43	4.36	2.04	3.28
WT2(5)	2.42	2.38	1.46	1.06	0.81	1.84	2.84	1.83	2.79	0.87	1.83
WT2(n/m)	3.69	2.64	1.30	0.89	0.72	2.06	3.61	2.24	2.90	1.25	2.13
WT2(n)	1.24	1.06	0.58	0.39	0.28	0.77	1.40	0.72	1.37	0.36	0.82
CDS1	5.36	4.10	2.79	2.56	2.76	3.99	4.36	2.87	5.10	3.14	3.70
CDS2	4.58	3.45	2.34	2.04	2.04	3.42	3.84	2.45	3.76	3.09	3.10
DNN	7.51	5.28	3.68	3.10	2.64	5.15	6.88	4.25	5.60	4.83	4.89
JbH1	4.43	3.29	2.11	1.86	2.07	3.18	3.52	2.64	3.25	2.89	2.92
JbH2	2.93	2.67	1.71	1.45	1.82	2.29	2.45	1.77	2.75	2.04	2.19

Table 2: Detailed values of ARPD grouped by n, m, and s

Heuristic	20x5	20x10	20x20	50x5	50x10	50x20	100x5	100x10	100x20	150x5	150x10	150x20	All
NEH	3.77	5.25	4.71	1.76	3.88	5.39	0.92	2.09	3.60	0.38	1.76	2.69	3.02
NEH_{SPT}	6.30	6.39	5.55	3.08	5.13	6.19	1.50	3.58	4.22	1.27	2.57	3.76	4.13
NEH_{LPT}	7.94	9.22	9.06	4.60	10.10	13.05	4.20	9.88	13.40	2.96	8.83	13.62	8.90
$\text{NEH}_{\text{PT+WINQ+AT}}$	3.96	5.30	5.51	1.94	4.03	5.30	0.91	2.22	3.09	0.51	1.52	2.75	3.09
NEH_{rand}	5.73	6.99	6.20	2.93	5.79	8.32	1.68	3.76	6.15	1.01	3.26	5.16	4.75
Palmer	10.35	10.60	10.11	6.36	8.46	9.68	3.13	5.13	6.95	2.31	4.11	4.91	6.84
BH	17.71	16.26	19.20	8.30	13.65	23.56	5.43	10.51	17.70	4.58	9.16	15.13	13.43
$\text{NEH}_{\text{LPT}}(\lambda)$	6.41	7.44	7.15	2.46	5.44	8.71	1.14	3.49	5.43	0.44	2.44	4.36	4.58
bNEH	7.37	5.85	5.71	2.68	4.01	4.98	1.26	2.70	3.12	0.90	1.90	3.06	3.63
$\text{bNEH}_{\text{SPT}}(\lambda)$	10.27	8.62	6.85	5.62	6.43	7.34	3.10	4.71	6.33	2.17	4.16	5.27	5.91
SPTB	18.67	14.82	13.76	10.93	12.38	15.09	6.64	9.47	10.54	5.36	9.01	9.85	11.38
bLPTB	30.14	26.67	20.22	21.13	24.01	23.17	16.04	19.25	21.33	12.69	17.73	18.22	20.88
FMCH	3.06	4.31	4.52	1.53	3.14	5.25	0.72	2.25	3.85	0.43	1.47	2.81	2.78
MCH(1)	2.77	3.46	3.90	1.26	2.80	3.83	0.60	1.19	2.22	0.29	1.09	1.70	2.09
MCH(2)	1.99	2.68	2.67	0.97	1.54	2.90	0.44	0.93	2.01	0.34	0.85	1.40	1.56
MCH(4)	0.98	1.06	1.11	0.52	1.17	1.63	0.42	0.80	1.48	0.23	0.61	0.79	0.90
MCH(6)	0.87	1.11	1.08	0.41	1.13	0.89	0.24	0.74	0.86	0.19	0.54	0.87	0.75
MCH(8)	0.95	1.27	0.94	0.50	0.76	0.99	0.46	0.40	0.94	0.21	0.61	0.72	0.73
MCH(10)	0.94	1.21	0.95	0.30	0.95	0.94	0.37	0.61	0.80	0.16	0.54	0.65	0.70
MCH(12)	0.99	1.20	1.01	0.35	0.89	1.08	0.31	0.39	0.72	0.15	0.39	0.58	0.67
WT1(1)	3.57	4.73	5.19	1.67	3.63	5.52	0.79	1.78	3.79	0.39	1.52	3.53	3.01
WT1(5)	1.66	2.69	2.59	0.81	1.88	3.50	0.26	1.00	2.28	0.21	0.86	2.21	1.66
WT1(n/m)	1.74	3.52	5.19	0.61	1.88	4.50	0.15	0.79	2.28	0.08	0.60	1.86	1.93
WT1(n)	0.82	1.40	1.47	0.29	1.00	1.91	0.07	0.50	1.03	0.03	0.38	0.76	0.81
WT2(1)	4.01	5.46	5.08	1.56	4.97	5.84	0.65	2.28	4.25	0.40	1.80	3.03	3.28
WT2(5)	1.94	2.58	2.73	0.72	2.47	3.96	0.38	1.27	2.73	0.19	1.04	1.95	1.83
WT2(n/m)	2.04	3.96	5.08	0.62	2.47	4.83	0.14	1.02	2.73	0.10	0.80	1.79	2.13
WT2(n)	0.78	1.41	1.52	0.26	0.91	2.01	0.06	0.43	1.26	0.03	0.33	0.81	0.82
CDS1	5.51	5.73	4.84	2.54	4.60	5.17	1.69	2.66	4.01	1.31	2.99	3.40	3.70
CDS2	4.36	5.03	4.33	1.99	3.82	4.56	1.00	2.51	3.50	0.80	2.32	2.99	3.10
DNN	5.69	8.07	8.75	2.38	5.55	7.91	1.43	3.80	5.80	1.07	3.16	5.07	4.89
JbH1	4.40	4.74	4.15	2.13	3.59	4.16	0.96	2.34	3.03	0.80	2.05	2.73	2.92
JbH2	3.67	2.76	2.34	1.96	2.85	3.20	0.92	1.82	2.38	0.74	1.75	1.86	2.19

Table 3: Detailed values of ARPD grouped by nxm

Heuristic	ARPD	ACT	ARPT	Heuristic	ARPD	ACT	ARPT
MCH(12)	**0.67**	**5.61**	**2.09**	NEH	3.02	0.29	0.13
MCH(10)	**0.70**	**4.44**	**1.79**	$\text{NEH}_{\text{PT+WINQ+AT}}$	3.09	0.72	0.22
MCH(8)	**0.73**	**3.46**	**1.46**	CDS2	3.10	0.00	0.00
MCH(6)	**0.75**	**2.51**	**1.07**	WT2(1)	3.28	0.42	0.16
WT1(n)	0.81	57.99	9.77	bNEH	3.63	0.32	0.11
WT2(n)	0.82	58.00	9.78	CDS1	3.70	0.00	0.00
MCH(4)	**0.90**	**1.68**	**0.71**	NEH_{SPT}	4.13	0.75	0.22
MCH(2)	**1.56**	**0.94**	**0.39**	$\text{NEH}_{\text{LPT}}(\lambda)$	4.58	0.25	0.08
WT1(5)	1.66	2.13	0.91	NEH_{rand}	4.75	0.51	0.18
WT2(5)	1.83	2.13	0.87	DNN	4.89	0.00	0.00
WT1(n/m)	1.93	4.90	1.07	$\text{bNEH}_{\text{SPT}}(\lambda)$	5.91	0.28	0.09
MCH(1)	**2.09**	**0.61**	**0.24**	Palmer	6.84	0.00	0.00
WT2(n/m)	2.13	4.89	1.01	NEH_{LPT}	8.90	0.90	0.25
JbH2	**2.19**	**0.01**	**0.04**	SPTB	11.38	0.00	0.00
FMCH	2.78	0.30	0.13	BH	13.43	0.00	0.00
JbH1	**2.92**	**0.00**	**0.00**	bLPTB	20.88	0.00	0.00
WT1(1)	3.01	0.43	0.18				

Table 4: Summary of computational results. In bold type the efficient heuristics for the problem are shown

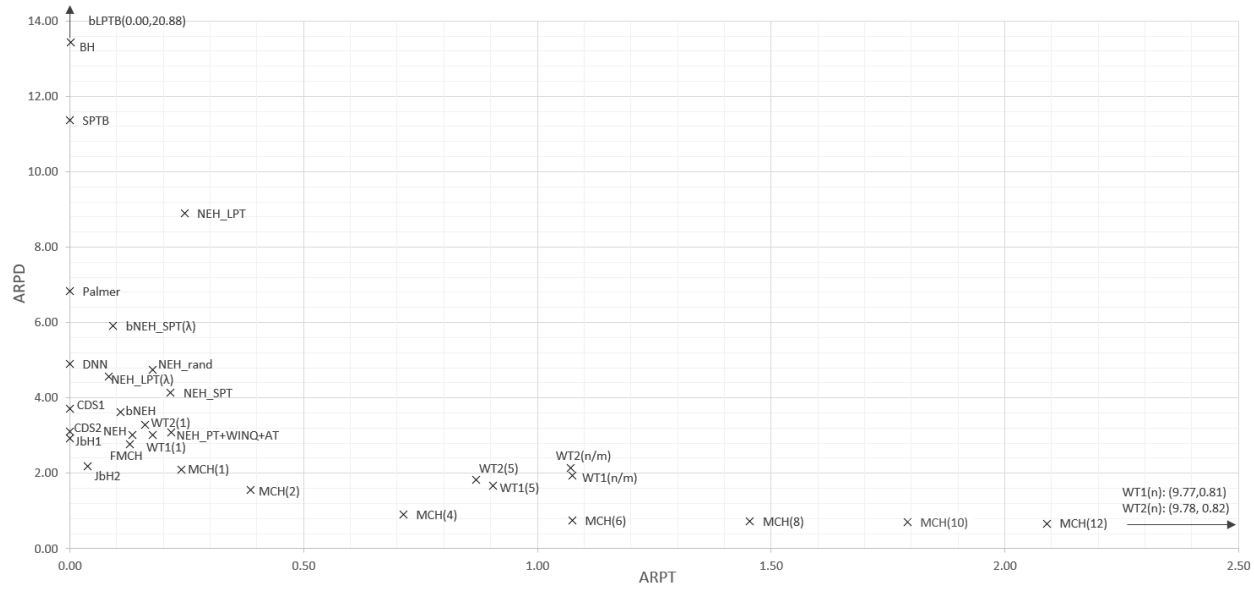

Figure 5: ARPD versus ARPT

Hypothesis	p-value	β_i	Wilcoxon	$\alpha/(8-\beta_i+1)$	Holm's procedure
H_2: JbH2=CDS2	0.000	1	R	0.006	R
H_3: JbH2=NEH	0.000	2	R	0.007	R
H_6: MCH(4)=WT1(5)	0.000	3	R	0.008	R
H_7: MCH(4)=WT2(5)	0.000	4	R	0.010	R
H_8: JbH2=WT1(1)	0.000	5	R	0.013	R
H_4: FMCH=NEH	0.018	6	R	0.017	R
H_1: JbH1=CDS2	0.030	7		0.025	
H_5: MCH(12)=WT1(n)	0.069	8		0.050	

Table 5: Holm's Procedure

5 Conclusions

In this paper, we have proposed two memory-based heuristics (FMCH and MCH) and two Johnson-based constructive heuristics (JbH1 and JbH2) to efficiently solve the hybrid flowshop with makespan minimisation. The first two heuristics use promising non-selected moves from past iterations to be repeated in future steps, while the other two construct several reduced

two-machine flowshop problems which are optimally solved using Johnson's algorithm. An extensive computational evaluation has been carried out comparing the proposals with the existing heuristics for the problem.

Regarding the computational evaluation of heuristics, the best results in terms of efficiency have been found by the JbH1, JbH2, and MCH heuristics. On the one hand, excellent fast solutions are found by the Johnson-based constructive heuristics, i.e. CDS2, JbH1, and JbH2. Note that, despite of the good performance of CDS2, the use of this heuristic, as seed sequence or comparison heuristic, is very scarce in the literature on the $HFm||C_{max}$ problem since its proposal by Brah and Loo (1999). On the other hand, the best results in terms of quality of the solutions are found by the MCH heuristic. In addition, our proposals (JbH1, JbH2, and MCH) outperform each other heuristic in the literature, representing the new state-of-the-art heuristics for the problem.

Regarding future research lines, further analysis could be conducted by comparing the efficiency of the new state-of-the-art heuristics when these are embedded in both population and non-population metaheuristics. In addition, although special effort has been carried out for generating a complete set of instances, we consider that further research should be addressed by building an extensive and exhaustive set of instances for the hybrid flowshop, including bottleneck considerations in a stage (see Fernandez-Viagas and Framinan, 2017b), and thus generating instances that are not solved easily.

Acknowledgements

The authors wish to thank the referees for their comments on the earlier versions of the manuscript. This research has been funded by the Spanish Ministry of Science and Innovation, under the project "PROMISE" with reference DPI2016-80750-P.

References

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

Adams, J., Balas, E., and Zawack, D. (1988). Shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401.

Alaykýran, K., Engin, O., and Döyen, A. (2007). Using ant colony optimization to solve hybrid flow shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 35(5-6):541–550.

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

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

Campbell, H., Dudek, R., and M.L., S. (1970). Heuristic algorithm for the n job, m machine sequencing problem. *Management Science*, 16(10):630–637.

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

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

Dannenbring, D. G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, 23(11):1174–1182.

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

Dong, X., Huang, H., and Chen, P. (2008). An improved NEH-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, 35(12):3962–3968.

Engin, O. and Döyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Computer Systems*, 20(6 SPEC. ISS.):1083–1095.

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

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

Fernandez-Viagas, V. and Framinan, J. (2017a). A beam-search-based constructive heuristic for the pfsp to minimise total flowtime. *Computers and Operations Research*, 81:167–177.

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

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

Fernandez-Viagas, V., Leisten, R., and Framinan, J. (2016). A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. *Expert Systems with Applications*, 61:290–301.

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

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

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

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

Goldratt, E. and Cox, J. (1992). *The Goal*. North River Press.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5:287–326.

Gupta, J. (1971). A functional heuristic algorithm for the flowshop scheduling problem. *Operational Research Quarterly*, 22(1):39–47.

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

Ho, J. (1995). Flowshop sequencing with mean flowtime objective. *European Journal of Operational Research*, 81(3):571–578.

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

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

Hundal, T. and Rajgopal, J. (1988). An extension of palmer's heuristic for the flow shop scheduling problem. *International Journal of Production Research*, 26(6):1119–1124.

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

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

Kalczynski, P. J. and Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. *OMEGA, The International Journal of Management Science*, 35(1):53–60.

Kalczynski, P. J. and Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers & Operations Research*, 35(9):3001–3008.

Kalczynski, P. J. and Kamburowski, J. (2009). An empirical analysis of the optimality rate of flow shop heuristics. *European Journal of Operational Research*, 198(1):93 – 101.

Kizilay, D., Tasgetiren, M., Pan, Q.-K., and Wang, L. (2015). An iterated greedy algorithm for the hybrid flowshop problem with makespan criterion. *IEEE Symposium Series on Computational Intelligence - CIPLS 2014, Proceedings*, pages 16–23.

Koulamas, C. and Kyparisis, G. (2000). Asymptotically optimal linear time algorithms for two-

stage and three-stage flexible flow shops. *Naval Research Logistics*, 47(3):259–268.

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

Kurz, M. and Askin, R. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, 159(1):66–82.

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

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

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

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

Nawaz, M., Enscore, J. E. E., and Ham, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem. *OMEGA, The International Journal of Management Science*, 11(1):91–95.

Negenman, E. (2001). Local search algorithms for the multiprocessor flow shop scheduling problem. *European Journal of Operational Research*, 128(1):147–158.

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

Norman, B. and Bean, J. (1999). A genetic algorithm methodology for complex scheduling problems. *Naval Research Logistics*, 46(2):199–211.

Néron, E., Baptiste, P., and Gupta, J. (2001). Solving hybrid flow shop problem using energetic reasoning and global operations. *Omega*, 29(6):501–511.

Palmer, D. (1965). Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near optimum. *Operational Research Quarterly*, 16(1):101–107. cited By 346.

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

Park, Y., Pegden, C., and Enscore, E. (1984). A survey and evaluation of static flowshop scheduling heuristics. *International Journal of Production Research*, 22(1):127–141.

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

Pinedo, M. and Chao, X. (1999). *Operations Scheduling with Applications in Manufacturing and Services*. McGraw-Hill, New York.

Ribas, I., Leisten, R., and Framinan, J. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Com-*

puters and Operations Research, 37(8):1439–1454.

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

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

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

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

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

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

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

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

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