New hard benchmark for the 2-stage multi-machine assembly scheduling problem: Design and computational evaluation

February 5, 2021

Abstract

The assembly scheduling problem is a common layout with many applications in real manufacturing scenarios. Despite the high number of studies dealing with this problem, no benchmark has been proposed up-to-now in the literature generating neither hard nor balanced instances. In this paper we present two extensive sets of instances for two variants of the 2-stage assembly scheduling problem. The first set is composed of 240 instances for the variant with one assembly machine in the second stage, while in the second set 960 instances are proposed for the variant with several assembly machines. An exhaustive experimental procedure, generating several preliminary testbeds with different processing times and number of jobs and machines, is carried out in order to identify the most representative instances of the problem under study. A total of 120,000 instances are generated and, among them, 1,200 are selected ensuring that the new benchmarks satisfy the desired characteristics of any benchmark: adequacy, empirical hardness, exhaustiveness, and amenity for statistical analysis. Finally, two computational evaluations are performed comparing and evaluating the existing heuristics in the literature, thus establishing the set of efficient heuristics for this assembly problem.

Keywords: scheduling, assembly, benchmark, testbed, instances generator, total completion time

1 Introduction

Assembly scheduling problems have many applications in the industry (Sheikh et al., 2018), since many products are made up of different components that need to be manufactured in the earlier stages and then assembled in a later stage. Some examples of applications are: personal computer manufacturing (Potts et al., 1995), fire engine assembly plants (Lee et al., 1993), circuit board production (Cheng and Wang, 1999), food and fertilizer production (Hwang and Lin, 2012), car assembly industry (Fattahi et al., 2013), motor assembly industry (Liao et al., 2015), or plastic industry (Allahverdi and Aydilek, 2015). Other examples can also be found in services/IT, including distributed database systems (Allahverdi and Al-Anzi, 2006; Al-Anzi and Allahverdi, 2006b, 2007), or multi-page invoice printing systems (Zhang et al., 2010).

Among the different assembly scheduling problems, in this paper we focus on the so-called 2-stage assembly scheduling problem. This problem consists of m_1 ($m_1 > 1$, fixed) dedicated parallel machines (DPm_1[1]) in the first stage (to manufacture the components) and m_2 ($m_2 \geq 1$, fixed) assembly machines in the second stage. The objective considered is the minimization of the total completion time. The variant with one assembly machine (labelled as SA or Single Assembly in the following) is denoted as $DPm_1 \rightarrow 1||\sum C_j$ by Framinan et al. (2019), and is equivalent to the regular two-machine flow-shop scheduling problem if there is only one machine in the first stage. Since the $F2||\sum C_j$ problem is strongly NP-hard (Garey et al., 1976), our problem is also NP-hard. The variant with several identical parallel machines in the last stage is denoted as $DPm_1 \rightarrow Pm_2||\sum C_j$, and is referred as MA (from Multi machine Assembly) in the following. Obviously, this problem is also strongly NP-hard (Garey et al., 1976).

Despite the considerable number of papers published so far proposing constructive and improvement heuristics to solve both variants (for more details see the reviews by Hwang and Lin, 2018; Sheikh et al., 2018; Framinan et al., 2019), the state of the art regarding solution methods for the problem is unclear. One of the causes might be the lack of a standard and representative testbed since:

- Every time that a new approximate method for the problem is proposed, it is tested in

[1]An alternative notation is PDm_1, used in Leung et al. (2005) and Roemer (2006).

different different sets of instances with different parameters and processing times. Therefore, it might occur that a method obtains a good performance in a set of instances and a bad performance in another one. This fact may lead to an unclear knowledge about the state-of-the-art algorithms for this problem (see Section 4, where some results are different from those obtained in Talens et al., 2020 where the same heuristics are compared using a different testbed).

- Since the scheduling problems are very sensitive to the input data of the instance (Fernandez-Viagas and Framinan, 2015b), in some cases the methodologies adopted to generate the processing times of the instances do not guarantee that the researchers are solving their specific problems. More specifically, we will show that there is a strong relationship among the variants under study and the customer order and the traditional parallel machine problem, among others (see Section 3.1.1 for more details). In other words, the good performance of some approximate methods might have been established by solving, in fact, instances of a different (albeit related) scheduling problem. As a consequence, to fulfil this requirement, the procedure to design a new benchmark has to follow a methodology which ensures the adequacy of the instances to the scheduling problem under study such as e.g. in Fernandez-Viagas and Framinan (2020).
- The procedures adopted to generate the instances do not ensure that the resulting instances represent the hardest ones. This is an important aspect (see Vallada et al., 2015; Fernandez-Viagas and Framinan, 2020) since, for a given scheduling problem, using a solution procedure that outperforms others in the hardest instances ensures an excellent performance of this procedure when applied to easier instances, whereas the opposite does not have to be true.

This paper is aimed to tackle these issues. More specifically, the contribution is twofold: First, a computational analysis is performed in order to determine the relationship among our variants and the related scheduling problems. Then, using this information, we propose two comprehensive benchmarks for the 2-stage multi-machine scheduling problem with total completion time criterion (one for each variant considered). Secondly, a computational evaluation is performed in order to compare and evaluate the existing heuristics in the literature in both considered variants.

The paper is organised as follows: in Section 2 we define the 2-stage assembly scheduling problem, and review the different sets of instances proposed in the literature; in Section 3 the procedure to generate the proposed instances is detailed; the computational evaluation of heuristics is carried out in Section 4; and, finally, Section 5 presents the conclusions of the paper.

2 Background

The problem studied in this paper can be stated as follows: there are n jobs to be scheduled in a layout composed of two stages. Each job has $m_1 + 1$ operations. In the first stage, there are m_1 dedicated parallel machines, where the first m_1 operations are conducted, one in each machine i, with a processing time given by p_{ij}. In the assembly stage there are m_2 identical parallel machines, which execute the last of the $m_1 + 1$ operations, being $m_2 \geq 1$. Only after the first m_1 operations are completed, the assembly operation may start in a machine of the second stage with a processing time denoted as at_j. The decision problem consists on scheduling the jobs on each machine so the sum of the completion times of the jobs is minimised.

Regarding the relation with similar scheduling problems, the *SA* variant is highly connected to two scheduling problems. On the one hand, it can be considered that the Customer Order scheduling problem, denoted *CO*, is tantamount to the one under consideration if the processing times of the jobs in the assembly stage are zero. Even if this is not the usual case, it could be interesting to analyse the similarity between both problems if the processing times in the first stage are much higher than in the assembly stage, as in this case the solution methods for the *CO* scheduling problem could potentially be applied to our problem. On the other hand, if the processing times in the second stage are much higher than those in the first stage, it can be assumed that *SA* is similar to the Single Machine scheduling problem, denoted *SM*. Note that the two-machine flowshop scheduling problem is a particular case of this problem if $m_1 = 1$. As with *SA*, depending on the influence of the processing times on each stage, the *MA* variant is connected to the *CO* scheduling problem, or to the Parallel Machine scheduling problem, denoted as *PM* in the following.

Next, we present a literature review for our problem, which is divided into two parts: one related to the existing solution procedures; and another one related to the sets of instances used in the literature to test the different proposed methods.

2.1 Heuristics for the considered problems

In this section, we review the literature and analyse the existing solution procedures for the *SA* and *MA* variants under consideration. In addition, we incorporate a review of the *CO*, *SM* and *PM* problems due to their relation with the variants under study, as discussed previously.

Concerning the *SA* variant, Tozkapan et al. (2003) prove that there exists an optimal solution where the jobs are processed in the same sequence for all the machines, and propose two heuristics to find an upper bound for their branch and bound algorithm. Al-Anzi and Allahverdi (2006a) derive a number of theoretical properties and propose three simple constructive heuristics based on the idea of sorting the jobs according to the Shortest Processing Time (SPT) rule, and two additional constructive heuristics. Framinan and Perez-Gonzalez (2017b) develop a constructive heuristic which outperforms the previous ones. Finally, Lee (2018) propose six lower bounds, which are tested in a branch and bound algorithm, and four greedy-type constructive heuristics. The branch and bound algorithm is compared against the proposal by Tozkapan et al. (2003), while the heuristics are not contrasted with the previous literature.

Regarding the *MA* variant, there are some references addressing the assembly scheduling problem with several machines in the second stage. Sung and Kim (2008) and Al-Anzi and Allahverdi (2012) consider two identical parallel machines in the second stage. Sung and Kim (2008) develop an heuristic applying a processing time-based pairwise exchange mechanism, while in Al-Anzi and Allahverdi (2012), a mathematical model and three metaheuristics are proposed. Recently, Talens et al. (2020) consider the problem with several identical parallel machines in the second stage and design two constructive heuristics, which outperfoms the previous ones.

Regarding the *CO* problem (or equivalently the Concurrent Open Shop scheduling problem, see Wagneur and Sriskandarajah, 1993; Roemer, 2006), which is NP-hard according to Roemer and Ahmadi (1997) for more than one machine, Sung and Yoon (1998) propose two constructive heuristics based on the SPT rule. The first one schedules the order (equivalent to a job in our context) with the smallest total processing time across all m machines and the second one selects the order with the smallest maximum processing time across the m machines. Ahmadi et al. (2005) characterize the optimal schedule, derive tight lower bounds, and propose several heuristic solutions. Leung et al. (2005) present some optimality properties and propose a constructive heuristic that selects as the next order to be sequenced the one that would be completed the earliest, that is, the order with the Earliest Completion Time (ECT). Based on this idea and including some look-ahead concepts, Framinan and Perez-Gonzalez (2017a) propose a constructive heuristic and two specific local search mechanisms for the problem.

With respect to the other related problems, *SM* can be optimally solved in polynomial time by the Shortest Processing Time (SPT) rule, as established in Smith (1956), and *PM* can be solved in polynomial time (Conway et al., 1967) by the SPT plus ECT rule, as established in Pinedo (2008). As we will see in Section 4, most of the methods analysed in this section are re-implemented in this study and a comparison among them is carried out to determine the most efficient ones.

2.2 Sets of instances in the related literature

In this section, we analyse the sets of instances used in the literature to test the solution methods applied to the considered problem. We also incorporate in the analysis related scheduling problems with different constraints, which can be easily adapted to the problem under study. In Table 2, we summarize the characteristics of the different sets of instances. The table is organised as follows: the first column indicates the problem for which the set of instances is designed and the second column indicates the paper in which the set of instances is tested. The number of instances is shown in the third column and, the number of jobs, n, is considered in the fourth column. The number of machines in the first stage, m_1, and in the second stage, m_2, are shown in the fifth and sixth column, respectively. Finally, the last two columns show the different distributions adopted to generate the processing times in the first stage, p_{ij}, and in the second stage, at_j.

Some observations about the characteristics of the different sets of instances can be made:

- Regarding the number of jobs, n, some papers (see Sung and Kim, 2008 and Lee, 2018) consider a number of jobs smaller than 15, while others (see Leung et al., 2005; Al-Anzi and Allahverdi, 2006a; Lin et al., 2008) consider a higher number of jobs, being 120, 200 and 500 the maxima, respectively. For the most commonly used testbeds (Al-Anzi and Allahverdi, 2006a; Allahverdi and Al-Anzi, 2012), the maximum number of jobs considered are 120 and 70, respectively. There are other cases using higher values of the number of jobs (e.g. Leung et al., 2005; Blocher and Chhajed, 2008; Shi et al., 2018), but these sets of instances have been scarcerly used.

- Concerning the number of machines in both stages, there are some papers which consider only one level of m_1, such as Sung and Yoon (1998), Wu et al. (2018) and Sung and Kim (2008) with m_1=2 or Lee (2018) with m_1=5. However, the rest of the works consider different levels of m_1. Regarding the SA variant, the most used testbeds are those by Al-Anzi and Allahverdi (2006a) and Allahverdi and Al-Anzi (2012), where $m_1 \in \{2, 4, 6, 8\}$. With respect to the number of machines in the second stage, Allahverdi and Al-Anzi (2009) and Tozkapan et al. (2003) consider one assembly machine. There are also papers which consider different levels of m_2 (see Nejati et al., 2016; Mozdgir et al., 2013).

- Regarding the processing times, some papers (see Leung et al., 2005 or Al-Anzi and Allahverdi, 2006a) follow a uniform distribution $U[1, 100]$ in both stages. Other works (see Sung and Yoon, 1998; Tozkapan et al., 2003; Sung and Kim, 2008) generate separately the processing times in different classes (see Table 1) in order to incorporate what the respective authors consider as dominance between the two stages [2]. In Sung and Yoon (1998), the first class represents the balance between the stages. In the second class, the workload in the first stage is slightly higher than that in the second stage. Finally, the third class represents the extremely unbalanced case. Tozkapan et al. (2003) represents the non-dominance case between the stages in the first class; in the second class the second stage dominates the first one, while in the third case, the opposite occurs. Sung and Kim (2008) represents the balance between the stages in the first class and, in the second class,

[2]In the cited works it can be checked that no study has carried out to ensure that the generated instances are representative of the different classes.

	Class 1		Class 2		Class 3	
Paper	Stage 1	Stage 2	Stage 1	Stage 2	Stage 1	Stage 2
Sung and Yoon (1998)	U[1,30]	U[1,30]	U[5,30]	U[1,25]	U[10,30]	U[1,20]
Tozkapan et al. (2003)	U[1,100]	U[1,100]	U[1,80]	U[20,100]	U[20,100]	U[1,80]
Sung and Kim (2008)	U[1,20]	U[1,20]	U[1,15]	U[5,20]		

Table 1: References with the generation of the processing times in different classes.

the case in which the workload on the second stage is higher than that in the first stage.

- There are some papers proposing two sets of instances with different sizes to test both exact and approximate methods, e.g. two sets with 900 instances each one are proposed in Sung and Yoon (1998) or one set with 360 instances and another one with 2,430 instances are designed in Mozdgir et al. (2013).

After this analysis, some conclusions can be obtained:

- Compared to other testbeds designed for different scheduling problems (see Taillard, 1993; Vallada et al., 2015; Fernandez-Viagas and Framinan, 2020), the levels of the number of jobs considered in most testbeds are very small.
- Some sets of instances are not well-suited for a statistical analysis as they do not use the same number of levels of the parameters (number of machines in both stages) for each number of jobs (see Mozdgir et al., 2013).
- So far no study has been carried out to ensure that any of the so-generated instances are representative of the problem under study.
- Since all proposals to solve the problem, approximate and exact algorithms, have been tested using very different sets of instances, the conclusions obtained may be different depending on the set used, i.e. the values of the quality of the solutions and computational effort may become very different depending on the chosen set.
- To the best of our knowledge, no analysis of the hardness of the instances has been carried out, so the instances generated might be relatively easy to solve.

Table 2: Sets of instances generated to solve related problems.

Problem	Authors	Number of instances	n	m_1	m_2	p	at
		CO					
$DP2 \to 0\|\|\sum w_jC_j$	Sung and Yoon (1998)	900 900	$\{5,6,7,8,9,10\}$ $\{13,15,20,30,40,50\}$	2	0	$U[1,30]$ $U[5,30]$ $U[10,30]$	$U[1,30]$ $U[1,25]$ $U[1,20]$
$DP_m \to 0\|\|\sum C_j$	Leung et al. (2005)[a]	480	$\{20,50,100,200\}$	$\{2,5,10,20\}$	0	$U[1,100]$	$U[1,100]$
		SA					
$DP3 \to 1\|\text{learning-effect}\|\sum C_j$	Wu et al. (2018)	7200 7200	$\{8,10,12,14\}$ $\{30,40,50,60\}$	2	1	$U[1,100]$ $U[1,100]$ $U[1,100]$	$U[1,100]$ $U[1,50]$ $U[1,25]$
$DP_m \to 1\|\|\sum C_j$	Al-Anzi and Allahverdi (2006a)[b]	720	$\{20,40,60,80,100,120\}$	$\{2,4,6,8\}$	1	$U[0,100]$	$U[1,100]$
$DP_m \to 1\|\|\sum C_j$	Blocher and Chhajed (2008)	6000 4800	$\{6,9,12,15,18\}$ $\{20,50,100,200\}$	$\{2,4,6\}$	1	[c] [c]	[c] [c]
$DP_m \to 1\|\|\sum C_j$	Allahverdi and Al-Anzi (2012)[d]	600	$\{30,40,50,60,70\}$	$\{2,4,6,8\}$	1	$U[1,100]$	$U[1,100]$
$DP_m \to 1\|\|\sum C_j$	Lee (2018)	120	$\{6,8,10,12\}$	5	1	$U[1,100]$ $U[1,80]$ $U[20,100]$	$U[1,100]$ $U[20,100]$ $U[1,80]$
$DP_m \to 1\|ST_{si}\|\sum C_j$	Allahverdi and Al-Anzi (2009)	2700	$\{20,30,40,50,60,70\}$	$\{3,6,9\}$	1	$U[1,100]$	$U[1,100]$
$DP_m \to 1\|\|\sum w_jC_j$	Tozkapan et al. (2003)	120	$\{10,15\}$	$\{5,10\}$	1	$U[1,100]$ $U[1,80]$ $U[20,100]$	$U[1,100]$ $U[20,100]$ $U[1,80]$
$F2\|s_{ij}\|\sum C_j/n$	Allahverdi (2000)	210	$\{10,15,20,25,30,35\}$	1	1	$U[1,100]$	$U[1,100]$
$F2\|\|Lex(C_{max},\sum C_j)$	T'kindt et al. (2002)	300 200	$\{50,80,110,140,170,200\}$ $\{10,15,20,25\}$	1	1	$U[1,100]$	$U[1,100]$
$F2\|prmu\|\sum C_j$	Lin et al. (2008)	300	$\{50,100,200,300,400,500\}$	1	1	$U[1,100]$	$U[1,100]$
		MA					
$DP_m \to 2\|\|\sum C_j$	Sung and Kim (2008)	600	$\{5,7,9,11,13\}$	2	2	$U[1,20]$ $U[1,15]$	$U[1,20]$ $U[5,20]$
$DP_m \to P_m\|\|BS_{ij}, \text{work - shift - lot}\|\|\sum w_jC_j$	Nejati et al. (2016)	100	$\{10,15,20,100\}$	$\{2,3\}$	$\{1,2,3\}$	$U[1,25]$	$U[1,25]$
$DP_m \to R_m\|\|F_l(C_{max},\sum C_j)$	Mozdgir et al. (2013)	360 2430	$\{6,7,8\}$ $\{15,30,45\}$	$\{3,5,7\}$ $\{3,5,7\}$	$\{2\},\{2,3\}$ $\{2,3,4\}$	$U[1,100]$	$U[1,100]$
		PM					
$P_m \to 0\|\|\sum w_jC_j$	Leung et al. (2008)	1600	$\{20,50,100,200\}$	$\{2,5,10,20\}$	0	$U[0,100]$	$U[0,100]$
$P_m \to 0\|\|\text{p-batch, incompt}\|\|\sum w_jC_j$	Shi et al. (2018)	1920	$\{20,40,60,80,100,200,300,400\}$	$\{1,2,5,10\}$	0	$U[30,70]$	$U[30,70]$

[a]For example, this set of istances is also used in Framinan and Perez-Gonzalez (2017a).
[b]For example, this set of instances is also used in Framinan and Perez-Gonzalez (2017b).
[c]The distributions are explained in the text.
[d]For example, this set of instances is also used in Al-Anzi and Allahverdi (2012, 2013); Allahverdi and Al-Anzi (2012).

After carrying out this analysis, it is clear that an extensive testbed for each variant of the problem is relevant, verifying the characteristics that the previous sets of instances do not fulfil. Therefore, the issues identified previously in Section 1 have to be handled.

3 New benchmarks generation

In the previous sections, the testbeds from the literature have been analysed and some disadvantages have been identified. To overcome these issues, we propose two new large benchmarks, one for the *SA* variant, and other for the *MA* variant, which are detailed in this section. According to the works by Hall and Posner (2001), Vallada et al. (2015) and Fernandez-Viagas and Framinan (2020), the following characteristics of a testbed for scheduling problems are desirable:

- Adequacy: The way in which the processing times in both stages are generated is an important aspect to comply with the adequacy of the instances. This characteristic is highly connected to the concept of balance between the stages and, depending on how the processing times are generated, three scenarios can be distinguished:

  1. *1st stage* unbalance if the processing times in the first stage are higher than those in the second stage. It should be studied if the generated instances for *SA* and *MA* in that way can be efficiently solved by methods designed for the *CO* scheduling problem, since the assembly times might not greatly influence the total completion time.

  2. *2nd stage* unbalance if the processing times in the second stage are higher than those in the first stage. In this case, it should be studied if the generated instances for *SA* and *MA* in that way can be efficiently solved by methods designed for the *SM* and *PM* scheduling problems, respectively, since the processing times of the dedicated machines might not greatly influence the total completion time.

  3. *Balance* if the workload in both stages are similar. In this case we can assume that the instances generated are representative of the two variants under study, *SA* and *MA*.

- Exhaustiveness: The benchmarks should include a large number of instances and these instances should cover different sizes of the parameters of the problem.

- Amenability for statistical analysis: In order to perform suitable statistical tests, all the levels of all the parameters must be combined and the levels should be equidistant.

- Hardness: The proposed instances have to be hard to be solved by approximate algorithms, i.e. if two methods can find the optimal solution for most of the instances, the benchmark will not be interesting since it does not have discriminant power and it could not be used to compare the methods. This characteristic, together with the exhaustiveness, lead us to obtain a more discriminant benchmark.

The methodology adopted to select the most suitable instances is explained in Section 3.1. The experiments needed to apply the methodology are included in Section 3.2.

3.1 Methodology

In this section, the procedure generated to design the new benchmarks and satisfy the previous characteristics is described. Figure 1 shows the outline of the procedure. Firstly, the adequacy characteristic is determined by using exact and approximate methods selected from the problem under consideration and from the related scheduling problems. Sets of small and medium size preliminary testbeds are solved by these methods, and the solutions are evaluated for SA and MA. Both sets of instances depend on a parameter α (see Section 3.1.1). Once the most suitable value for this parameter is established, a large size preliminary testbed is generated to select the hardest ones to be included in the benchmark of each variant, SA and MA (see Section 3.1.2). This procedure is carried out evaluating the distance between a near optimal solution (solution obtained by applying an iterated greedy algorithm) and a lower bound of the problem. The description of the preliminary testbeds and the selection procedure to select the suitable instances for the final benchmarks, denoted $\mathcal{B}_1$ and $\mathcal{B}_2$ in Figure 1, are explained in Section 3.1.3. Finally, after the methodology has been determined, the experimental analysis is carried out in Section 3.2.

The use of a generic representation of the solutions (permutation based) is a key aspect in this methodology. For each one of the considered variants, the representation of a solution to provide a schedule is given by: For *SA*, the same sequence of jobs on all of the machines, including the assembly machine (Al-Anzi and Allahverdi, 2006a), and, for *MA*, a sequence on all the dedicated machines in the first stage and applying the ECT rule to assign the jobs to the assembly machines (Al-Anzi and Allahverdi, 2012; Talens et al., 2020). Regarding the related problems, for *CO*, the solution is given by a permutation of n components (Framinan and Perez-Gonzalez, 2017a); for *SM*, the schedule is given by a sequence (Smith, 1956) and, for *PM*, the solution is given by a sequence and assigning the jobs to the machines using a dispatching rule (Conway et al., 1967) as seen in Section 2. As it can be observed, the use of a sequence provides a generic solution for all the considered scheduling problems.

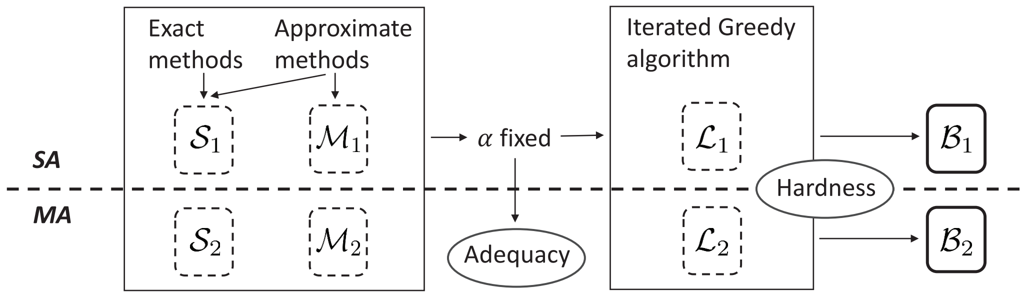


Figure 1: Diagram of the methodology designed to propose the new benchmarks.

3.1.1 Adequacy

In order to guarantee the adequacy of the generated benchmarks for each variant, *SA* and *MA*, the balance between the stages is analysed to determine the relationship among the problems. As usual in the problem under consideration (see Table 2) and in the scheduling literature (see Taillard, 1993; Vallada et al., 2015; Fernandez-Viagas and Framinan, 2020), the processing times are randomly generated from a uniform distribution. Therefore, we consider $p_{ij} \sim U[1, 100]$ on all m_1 machines in the first stage. Regarding the second stage, in order to control the aforementioned different scenarios of balance/unbalance, the processing times are generated using $at_j \sim U[1, \alpha \cdot m_2 \cdot 100]$, where α is a parameter representing the existing relationship between the workload of the two stages. Based on preliminary results in which a wider range of α values are evaluated, the instances consider $\alpha \in \{1, 2, 3\}$ for *SA* and *MA*. The objective is to determine the value of α that makes the instances suitable to be included in the benchmarks, avoiding those that can be efficiently solved by methods designed for *CO*, *SM* and *PM*.

To accomplish with the adequacy, small instances are first solved using exact methods (explained below), developed for *CO*, *SM*, *PM*, *SA* and *MA*. Secondly, some approximate methods are re-implemented for the same problems, and their solutions and conclusions are validated in the same small instances. Finally, the approximate algorithms are also tested in medium size instances to cover a wider extension of the problem under consideration. In the case that a (exact or approximate) method is specific for *CO*, *SM* or *PM*, the instances need to be adapted by making $p_{ij} = 0$ or $at_j = 0$, depending on each case. Let S_T denote the solution obtained from an algorithm specific for problem $\mathcal{T}$, with $\mathcal{T} \in \{SA, MA, CO, PM, SM\}$. Then, this solution is evaluated for other related problem by computing $\sum C_j^{\mathcal{T}'}(S_{\mathcal{T}})$, i.e. the total completion time of the solution obtained by a specific method of the problem $\mathcal{T}$, $S_{\mathcal{T}}$, evaluated for the problem $\mathcal{T}'$. Three different cases can be defined depending on the value of α:

- Instances for which specific methods designed for *CO* yield a good performance indicate that there is unbalance and the workload in the first stage is higher than in the second stage.
- Instances for which specific methods designed for SM/PM yield a good performance indicate that there is unbalance and the workload in the second stage is higher than in the first stage.
- Instances for which specific methods designed for SA/MA yield a good performance and specific methods designed for *CO* and SM/PM yield a bad performance indicate that are suitable and eligible for the benchmark.

Regarding the exact methods to solve the small size instances, the following have been employed:

- For *SA* and *MA*: the MILP model by Navaei et al. (2013).

- For *CO*: an adaptation of the previous model, removing the constraints related to the second stage.
- For *SM* and *PM*: SPT and SPT+ECT rules, respectively.

Regarding the approximate (heuristics) methods to solve the small and medium size instances:

- For *SA*: the best-known constructive heuristic for the $DPm_1 \rightarrow 1||\sum C_j$, FAP, by Framinan and Perez-Gonzalez (2017b). .
- For *MA*: the best-known constructive heuristic for the $DPm_1 \rightarrow Pm_2||\sum C_j$, CH_{MMA}, by Talens et al. (2020).
- For *CO*: the best-known constructive heuristic for solving the *CO* problem, $NEW-ECT$, by Framinan and Perez-Gonzalez (2017a).
- For *SM* and *PM*: SPT and SPT+ECT rules, respectively, since both problems are polynomially solvable.

The evaluation procedure of the adequacy is illustrated by an example, with four jobs to be scheduled in a 2-stage assembly system. The first stage consists of three dedicated parallel machines and the second stage has two identical parallel machines. The processing times of the jobs are shown in Table 3. Different methods are applied to the instance: First, the exact method for *MA* is applied, providing the solution $S_{MA} = (2, 1, 3, 4)$ with objective function $\sum C_j^{MA}(S_{MA})$=136 (see Figure 2a). Then, processing times in the second stage are considered equal to zero, and the exact method for CO is applied, providing the solution $S_{CO} = (1, 2, 3, 4)$. The sequence obtained is evaluated for *MA* using the data of the original instance (see Figure 2b), yielding $\sum C_j^{MA}(S_{CO})$=138. Finally, the SPT+ECT rule is applied to the instance assuming that the processing times in the dedicated machines are equal to zero (SM/PM case), obtaining $S_{PM} = (1, 4, 2, 3)$. As in the previous case, the sequence is evaluated for *MA* using the data of the original instance (see Figure 2c), with the value of the objective function $\sum C_j^{MA}(S_{PM})$=142.

j	p_{1j}	p_{2j}	p_{3j}	at_j
1	2	9	7	3
2	10	5	5	18
3	5	4	8	25
4	4	5	9	17

Table 3: Processing times of the jobs in both stages.

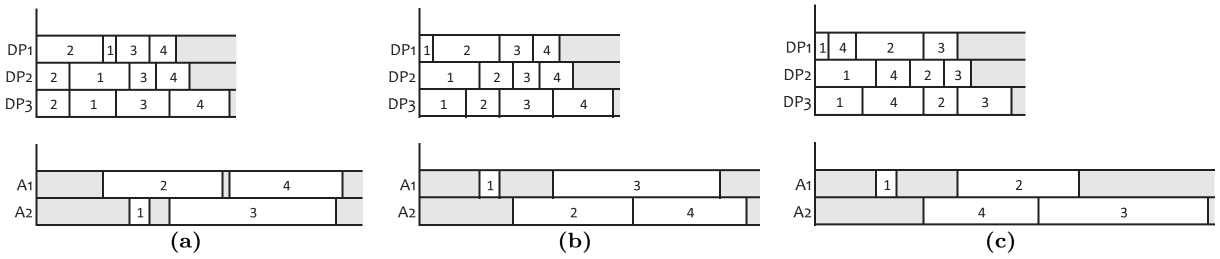


Figure 2: Gantt charts of the different optimal sequences evaluated for the *MA* variant. Figure 2a Gantt chart of the optimal sequence of the *MA* variant, S_{MA}. Figure 2b Gantt chart of the optimal sequence of the *CO* problem, S_{CO}, when it is evaluated for *MA*. Figure 2c Gantt chart of the optimal sequence of the *PM* problem, S_{PM}, when it is evaluated for *MA*.

3.1.2 Hardness

The empirical hardness of the instances has been measured by the difference between the performance of a metaheuristic and a lower bound, as in similar studies (see Taillard, 1993 and Vallada et al., 2015). The metaheuristic selected is the Iterated Greedy, denoted as IG, developed by Ruiz and Stützle (2007), which is among the most effective metaheuristics in the scheduling literature (see some IG based algorithms in Hatami et al., 2015; Lin, 2018; Pan et al., 2019) and it is also used to determine the empirical hardness in Vallada et al. (2015) and Fernandez-Viagas and Framinan (2020). Following the literature (see Hatami et al., 2015; Vallada et al., 2015), the stopping criterion is set to $n \cdot m/2 \cdot 90/1000$ seconds, where m is equal to $m_1 + 1$, since the number of machines in the second stage does not have influence since the ECT rule is applied. The parameters of the IG are set as in the original paper (Ruiz and Stützle, 2007).

The lower bound used in this study is proposed by Blocher and Chhajed (2008), where the authors addressed the problem $DPm_1 \rightarrow 1||\sum C_j$. This lower bound is computed following Equation (1), where $w_j = \sum_{i=1}^{m} p_{ij}/m_1$ and $w_{[k]}$ is the average processing time of job in position k with the jobs ordered according to $w_{[1]} \leq w_{[2]} \leq \cdots \leq w_{[n]}$. Similarly, $p_{i[k]}$ is the processing time of job in position k in machine i and the jobs are ordered according to $p_{i[1]} \leq p_{i[2]} \leq \cdots \leq p_{i[n]}$, with $1 \leq i \leq m_1$. The sum in Equation (1) is an estimate of the completion time of each job j in the system: On the one hand, the completion time of each job j in the dedicated machines is computed as the maximum between the sum of the (largest integer value given by the) average processing times of the jobs scheduled prior to job j given in the order defined by $w_{[k]}$; and the maximum among all the dedicated machines of the processing times of the jobs scheduled previously to job j given in the order defined by $p_{i[k]}$. On the other hand, the processing time in the second stage is added. Note that, since our lower bound proposed for the $DPm_1 \rightarrow 1||\sum C_j$ does not consider job waiting time between the first and the second stage, it is clear that it is also a LB for the $DPm_1 \rightarrow Pm_2||\sum C_j$ problem. Besides, since the total completion time cannot increase with the number of machines in the second stage, then this LB is tighter for the MA variant.

$$LB = \sum_{j=1}^{n} \left(max \left\{ \sum_{k=1}^{j} \left\lceil w_{[k]} \right\rceil, \max_{i=1,\ldots,m_1} \left\{ \sum_{k=1}^{j} p_{i[k]} \right\} \right\} + at_j \right) \tag{1}$$

The difference between the performance of the metaheuristic and the lower bound is determined using the Relative Percentage Deviation computed as follows:

$$RPD_{M_{\mathcal{T}}}(\mathcal{T}') = \frac{\sum C_j^{\mathcal{T}'}(h) - MIN}{MIN} \tag{2}$$

where MIN is the solution obtained by the lower bound and $\sum C_j^{\mathcal{T}'}(h)$ is the evaluation for $\mathcal{T}' \in \{SA, MA\}$ of the solution provided by metaheuristic h, in this case IG. Following the idea by Vallada et al. (2015), the higher the RPD_{IG}, the harder the instance is, i.e. the best known solution is further from the theoretical lower bound. On the contrary, a low value of RPD_{IG} means that the instance is easy, since the method is able to provide an objective function value close to the lower bound. The instances with the highest RPD values will be selected to be part of the new benchmark.

3.1.3 Preliminary testbeds and selection procedure

In this section, the procedure to select the final sets of instances, denoted as $\mathcal{B}_1$ and $\mathcal{B}_2$, is explained. The selection procedure is carried out using different preliminary testbeds: $\mathcal{S}_1$, $\mathcal{M}_1$ and $\mathcal{L}_1$ for SA including small, medium and large size instances, respectively; and $\mathcal{S}_2$, $\mathcal{M}_2$ and $\mathcal{L}_2$ for MA, equivalently. Regarding the parameters for the preliminary instances, the levels considered are (always verifying the equidistance and thus ensuring the amenability for statistical analysis):

- Number of jobs: $n \in \{8, 10, 12\}$ for $\mathcal{S}_1$ and $\mathcal{S}_2$, $n \in \{30, 40, 50, 60, 70\}$ for $\mathcal{M}_1$ and $\mathcal{M}_2$, and $n \in \{50, 100, 150, 200, 250, 300\}$ for $\mathcal{L}_1$ and $\mathcal{L}_2$. Although the maximum number of jobs considered in the literature is equal to 200 (see Table 2), it seems appropriate to consider a higher number of jobs following the literature of other benchmarks (see Vallada et al., 2015; Fernandez-Viagas and Framinan, 2020).

- Number of machines in the first stage: $m_1 \in \{2, 4\}$ in $\mathcal{S}_1$ and $\mathcal{S}_2$, and $m_1 \in \{2, 4, 6, 8\}$ (based on the literature, see Table 2) in $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{L}_1$, $\mathcal{L}_2$.

- Number of machines in the second stage: $m_2 = 1$ in $\mathcal{S}_1$, $\mathcal{M}_1$ and $\mathcal{L}_1$, $m_2 \in \{2, 3\}$ in $\mathcal{S}_2$, $m_2 \in \{2, 4, 6, 8\}$ in $\mathcal{M}_2$ and $\mathcal{L}_2$. In the literature (see Table 2) the maximum value of m_2 is 4, so here it has been extended.

- Number of replicates: 30 is established for $\mathcal{S}_1$ and $\mathcal{S}_2$, 10 for $\mathcal{M}_1$ and $\mathcal{M}_2$, and finally 1,000 for $\mathcal{L}_1$ and $\mathcal{L}_2$.

- Values of parameter α: $\alpha \in \{1, 2, 3\}$ for $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{M}_1$ and $\mathcal{M}_2$ (see Section 3.1.1). For $\mathcal{L}_1$ and $\mathcal{L}_2$, α is equal to 2, according to the results obtained in Section 3.2.

Considering these parameters, $\mathcal{S}_1$ and $\mathcal{S}_2$ have 540 and 1,080 instances, respectively; $\mathcal{M}_1$ and $\mathcal{M}_2$ 600 and 2,400 instances, respectively; and $\mathcal{L}_1$ and $\mathcal{L}_2$ 24,000 and 96,000 instances, respectively.

As far as the selection procedure is concerned, it consists of two steps:

Step 1. Analysis of the small and medium size preliminary testbeds: In order to obtain the adequacy of the instances, the most suitable values of α should be identified for SA and MA.

- For the *SA* variant, $\mathcal{S}_1$ is solved by the exact methods. Additionally, $\mathcal{S}_1$ and $\mathcal{M}_1$ are solved with the approximate methods. All the methods have been previously described in Section 3.1.1. Next, for each preliminary testbed, an Analysis of Variance (ANOVA) is carried out with the following factors: n, m_1, m_2, α and T, where T is the problem considered. The factor T has the levels SA, CO and SM. When exact methods are used, the dependent variable employed is the total completion time obtained after evaluating the optimal sequence of *CO* and *SM* in the *SA* variant for $\mathcal{S}_1$. For the approximate methods, the dependent variable is the total completion time obtained after evaluating the best sequence provided for the related problems in *SA*, in this case for $\mathcal{S}_1$ and $\mathcal{M}_1$. Next, a post-hoc Tukey's Honest Significant Difference test is applied to determine the statistical significant differences between the levels of each factor. In this study, the differences between the levels of the factor T are tested, contrasting the hypotheses $H_1 : SA = CO$ and $H_2 : SA = SM$, for each value of α.

- For the *MA* variant, the procedure is the same using $\mathcal{S}_2$ and $\mathcal{M}_2$. In this case the levels of T are MA, CO and PM, and the hypotheses $H_3 : MA = CO$ and $H_4 : MA = PM$.

The value of α for which all the hypotheses are rejected is chosen to generate the balanced instances of the corresponding variant, *SA* and *MA*.

Step 2. Selection of the most suitable instances: The hardness is determined by selecting the most suitable instances of the preliminary large size testbeds, $\mathcal{L}_1$ and $\mathcal{L}_2$, for *SA* and *MA*, respectively. These testbeds have been generated for the value of α provided in the previous step. The high number of replicates ensures that the hardest instances are included in the benchmark. In this case, as explained in Section 3.1.2, the instances are solved by the Iterated Greedy, and the lower bound for each one is computed. For each instance size, the *Average RPD* obtained is sorted in decreasing order and the 10 first instances are selected to be part of the new benchmark. The instances selected for each variant form the benchmark $\mathcal{B}_1$ for *SA*, with 240 large size instances (Section 3.2.1), and $\mathcal{B}_2$ for *MA*, with 960 large size instances (Section 3.2.2).

As $\mathcal{B}_1$ and $\mathcal{B}_2$ are selected from the testbeds $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, it is ensured that the instances are amenable for statistical analysis since all the levels of the parameters are combined and they are equidistant. In addition, as the benchmarks consist of a high number of instances, their exhaustiveness is also fulfilled. The instances files of benchmarks $\mathcal{B}_1$ and $\mathcal{B}_2$ are published as additional material in the following link: `http://grupo.us.es/oindustrial/en/research/results/`.

3.2 Experimental results

In this section, the results obtained applying the procedure of Section 3.1 are presented. Section 3.2.1 presents the results for *SA*, and Section 3.2.2 for *MA*. In each case, the ANOVA shows the statistical significant influence of all the factors in the response variable. For each value of α, we are interested in the differences among the levels of the factor T (problem). The results from the Tukey's HSD test are presented in a simplified way. All detailed results are available in Anova & Tukey Test Experimental Results files in http://grupo.us.es/oindustrial/en/research/results/.

3.2.1 Results for the *SA* variant

This section shows the results provided for the *SA* variant. Regarding the adequacy, the preliminary testbed $\mathcal{S}_1$ has been solved by exact methods, and both, $\mathcal{S}_1$ and $\mathcal{M}_1$ by approximate methods. Table 4 shows the conclusions provided by the Tukey's HSD test for each value of α. On the one hand, the hypothesis $H_1 : SA = CO$ contrasts the equality of the mean total completion time provided by the optimal solution of *SA*, and by the optimal solution of *CO* evaluated for *SA* when the exact methods are applied to $\mathcal{S}_1$. When the approximate methods are applied to $\mathcal{S}_1$ and $\mathcal{M}_1$, the equality of the mean total completion time of the best solution provided for *SA* and the best solution provided for *CO* is contrasted. On the other hand, the hypothesis $H_2 : SA = SM$ is similar comparing *SA* to *SM*. Therefore, in Table 4 it is shown if the hypothesis is rejected (R), or if there is not significant evidence to reject it (-), and the significance provided by the test (Sig.).

From the results for H_1, it can be observed that there are not statistical differences between *SA* and *CO* for $\alpha = 1$ when exact methods are applied to $\mathcal{S}_1$. Although for approximate methods the hypothesis is rejected, instances generated with $\alpha = 1$ are not suitable for *SA* since the exact method of CO provides good results. Additionally, for $\alpha = 2$ and $\alpha = 3$, CO and SA are not similar regardless the method applied to small and medium size instances. From the results for H_2, $\alpha = 1$ and $\alpha = 2$ indicate that SA and SM are not similar for all the cases. For $\alpha = 3$, there are not statistical differences between SA and SM when approximate methods are applied to $\mathcal{S}_1$. Therefore, instances generated with $\alpha = 3$ are not suitable for SA. As conclusions, the instances generated with $\alpha = 1$ and $\alpha = 3$ are not suitable (unbalanced) for the SA variant, and $\alpha = 2$ is consistently balanced because of the rejection of the hypotheses in all the cases.

		$H_1 : SA = CO$					
Method	Instances	α=1	Sig.	α=2	Sig.	α=3	Sig.
Exact	$\mathcal{S}_1$	-	0.993	R	1.000	R	1.000
Approximate	$\mathcal{S}_1$	R	1.000	R	1.000	R	1.000
Approximate	$\mathcal{M}_1$	R	1.000	R	1.000	R	1.000
		$H_2 : SA = SM$					
Method	Instances	α=1	Sig.	α=2	Sig.	α=3	Sig.
Exact	$\mathcal{S}_1$	R	1.000	R	1.000	R	1.000
Approximate	$\mathcal{S}_1$	R	1.000	R	1.000	-	0.150
Approximate	$\mathcal{M}_1$	R	1.000	R	1.000	R	1.000

Table 4: Conclusions from HSD Tukey tests for SA.

Regarding the hardness, testbed $\mathcal{L}_1$ has been solved by IG. Then, the $ARPD_{IG}^{SA}$ with respect to the lower bound has been computed. In Table 5, the values of $ARPD$ of the 10 hardest instances of each combination are shown in the rows labelled as “Selected instances”, while in the rows labelled as “Total instances” the values of $ARPD$ of the total instances of each combinations is computed.

3.2.2 Results for the MA variant

In this section, the results for the MA variant are shown. Similar to the previous section, the adequacy is analysed. The preliminary testbed $\mathcal{S}_2$ is solved by exact and approximate methods, and $\mathcal{M}_2$ by approximate methods. Table 6 has the same structure that Table 4. In this case, the hypotheses are $H_3 : MA = CO$ and $H_4 : MA = PM$.

Hypothesis H_3 shows that there are not statistical differences between MA and CO for $\alpha = 1$ when approximate methods are applied to $\mathcal{S}_1$. Although the hypothesis is rejected in the rest

	m_1	n 50	100	150	200	250	300	Average
Selected instances	2	114.54	109.90	112.70	108.23	102.77	101.73	108.31
	4	100.52	96.18	89.93	86.69	85.85	84.17	90.56
	6	95.30	90.40	83.71	79.39	77.56	77.65	84.00
	8	89.86	82.65	78.14	76.82	74.83	74.43	79.46
	Average	100.05	94.78	91.12	87.78	85.25	84.49	90.58
Total instances	2	62.71	69.24	72.26	73.39	73.38	74.07	70.84
	4	54.96	59.55	60.61	61.32	61.83	62.49	60.13
	6	50.80	54.87	56.25	56.84	57.28	57.62	55.61
	8	50.53	53.59	54.27	55.02	54.91	55.34	53.94
	Average	54.75	59.31	60.85	61.64	61.85	62.38	60.13

Table 5: Values of *ARPD* for the 10 hardest instances and for the total instances of each combination of parameters in testbed $\mathcal{L}_1$.

of the cases, instances generated with $\alpha = 1$ are not suitable for *MA*. Additionally, for $\alpha = 2$ and $\alpha = 3$, *CO* and *MA* are not similar regardless the method applied to small and medium size instances. For H_4, *MA* and *PM* are not similar for all the cases when $\alpha = 1$ and $\alpha = 2$. However, there are not statistical differences when $\alpha = 3$ and approximate methods are applied to $\mathcal{S}_2$. Therefore, instances generated with $\alpha = 3$ are not suitable for *MA*. In the same way that the previous case, instances generated with $\alpha = 1$ and $\alpha = 3$ are unbalanced for the *MA* variant, being $\alpha = 2$ the suitable value again.

$H_3 : MA = CO$							
Method	Instances	α=1	Sig.	α=2	Sig.	α=3	Sig.
Exact	$\mathcal{S}_2$	R	1.000	R	1.000	R	1.000
Approximate	$\mathcal{S}_2$	-	0.184	R	1.000	R	1.000
Approximate	$\mathcal{M}_2$	R	1.000	R	1.000	R	1.000
$H_4 : MA = PM$							
Method	Instances	α=1	Sig.	α=2	Sig.	α=3	Sig.
Exact	$\mathcal{S}_2$	R	1.000	R	1.000	R	1.000
Approximate	$\mathcal{S}_2$	R	1.000	R	1.000	-	0.317
Approximate	$\mathcal{M}_2$	R	1.000	R	1.000	R	1.000

Table 6: Conclusions from HSD Tukey tests for *MA*

Regarding the hardness of the instances, testbed $\mathcal{L}_2$ has been solved by *IG* and the same procedure as in the previous section is followed. Table 7 has a structure similar to Table 5, including a column for the number of jobs in the second stage m_2. As in the *SA* variant, the values of $ARPD$ of the hardest instances are higher than the average of all instances.

Finally, after following the procedure detailed throughout Section 3, the hardest instances of $\mathcal{L}_1$ and $\mathcal{L}_2$ form the proposed two new benchmarks, whose parameters can be summarised as follows:

- $\mathcal{B}_1$: $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$, $m_2 \in \{1\}$. 240 instances in total.
- $\mathcal{B}_2$: $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$, $m_2 \in \{2, 4, 6, 8\}$. 960 instances instances.

In summary, these benchmarks fulfil the required characteristics of a benchmark. The adequacy is achieved by generating instances with both stages balanced and, thus, representative of the *SA* and *MA* variants. The benchmarks are hard since the hardest instances have been selected after solving a huge number of instances with the Iterated Greedy algorithm and evaluating the solutions with respect to a lower bound. The exhaustiveness is ensured by the large number of instances, 240 and 960 respectively, in which different sizes of the parameters have been considered. Finally, the benchmarks are amenable for statistical analysis since the levels of the parameters are equidistant and all the levels have been combined to generate the instances.

4 Computational evaluation of heuristics

In this section, we analyse the efficiency, in the proposed benchmarks, of the heuristics reviewed in Section 2.1 and enumerated in Section 4.1. Then, in Section 4.2, a comparison among the heuristics is performed and the results are analysed. All methods have been coded in C# using Visual Studio and carried out in an Intel Core i7-3770 PC with 3.4 GHz and 16 GB RAM, using the same common functions and libraries. Finally, in Section 4.3, upper bounds of the problem are computed in order to compare the algorithms.

4.1 Heuristics

The existing heuristics developed to solve *SA*, *MA* and the related problems are adapted, if required, and implemented to solve each one of the considered variants.

	m_1	m_2	n 50	100	150	200	250	300	Average
Selected instances	2	2	98.15	100.64	105.70	102.85	95.26	96.52	99.85
		4	75.17	86.65	95.12	94.68	81.25	79.61	85.41
		6	59.47	78.06	81.78	85.70	74.26	72.44	75.29
		8	47.96	68.56	74.64	79.80	72.48	69.86	68.88
	4	2	87.31	88.56	84.72	82.49	92.04	90.95	87.68
		4	68.23	77.02	76.89	76.58	76.36	76.29	75.23
		6	54.82	66.07	72.22	74.58	69.91	70.98	68.10
		8	44.87	58.66	66.36	69.88	67.77	67.69	62.54
	6	2	83.30	83.60	79.05	75.64	86.39	84.87	82.14
		4	65.80	73.26	71.95	70.53	70.60	73.61	70.96
		6	53.10	61.50	65.45	67.22	66.59	67.72	63.60
		8	43.54	54.89	60.30	63.17	63.39	64.17	58.25
	8	2	78.64	76.51	73.84	73.37	81.58	83.18	77.85
		4	62.46	67.31	67.38	68.25	67.19	68.63	66.87
		6	50.67	60.04	62.45	63.25	62.27	64.55	60.54
		8	41.88	53.78	57.59	59.49	61.38	60.29	55.73
	Average		61.15	70.30	72.65	73.64	72.90	72.99	70.60
Total instances	2	2	53.76	63.33	67.67	69.58	70.54	71.64	66.09
		4	41.30	54.75	61.10	64.29	59.25	59.85	56.76
		6	32.39	48.80	54.86	59.43	55.05	55.10	50.94
		8	25.75	42.96	50.18	55.46	52.46	53.38	46.70
	4	2	47.61	54.70	56.89	58.21	65.93	67.51	58.48
		4	37.21	47.72	51.66	54.01	55.61	56.68	50.48
		6	29.66	41.66	47.61	51.05	51.66	52.37	45.67
		8	23.91	36.96	43.79	47.85	49.62	50.87	42.17
	6	2	44.20	50.55	52.92	54.06	62.10	63.95	54.63
		4	34.81	44.24	48.19	50.25	52.35	54.00	47.31
		6	27.94	39.26	44.26	47.18	49.14	50.37	43.02
		8	22.65	34.92	40.82	44.35	47.31	48.21	39.71
	8	2	44.08	49.49	51.15	52.39	58.29	61.13	52.76
		4	34.89	43.42	46.65	48.76	50.10	51.55	45.90
		6	28.12	38.06	43.29	45.84	46.67	48.22	41.70
		8	22.92	33.93	39.99	43.14	45.00	46.29	38.55
	Average		34.45	45.30	50.06	52.87	54.44	55.70	48.80

Table 7: Values of $ARPD$ for the 10 hardest instances and for the total instances of each combination of parameters in testbed $\mathcal{L}_2$.

- All the following methods have been developed for the SA variant. The adaptation to solve the MA variant can be consulted in Talens et al. (2020).
  - $TCK1$ and $TCK2$ (Tozkapan et al., 2003).
  - $A1$ and $A2$ (Al-Anzi and Allahverdi, 2006a).
  - $S1$, $S2$ and $S3$ (Al-Anzi and Allahverdi, 2006a).
  - FAP (Framinan and Perez-Gonzalez, 2017b).
  - $G1$, $G2$, $G3$ and $G4$ (Lee, 2018).
- The following methods have been developed to solve the CO problem. The adaptations to solve the MA variant are explained in Talens et al. (2020).
  - $STPT$ and $SMPT$ (Sung and Yoon, 1998).
  - ECT (Ahmadi et al., 2005; Leung et al., 2005).
  - $SHIFT_k$ and $SHIFT_{k_{OPT}}$ (Framinan and Perez-Gonzalez, 2017a).
- The next heuristics have been developed to solve the MA variant. Their adaptation to solve the SA variant is done by directly considering one assembly machine when the jobs are scheduled in the second stage.
  - SAK (Sung and Kim, 2008).
  - CH_{MMA}, $BSCH_V$ (x=2), $BSCH_V$ (x=n/10), $BSCH_V$ (x=5), $BSCH_V$ (x=10), $BSCH_V$ (x=15), $BSCH_{MMA}$ (x=n) and $BSCH_{MMA}$ (x=$n + n/2$) (Talens et al., 2020).

4.2 Evaluation in the new benchmarks, $\mathcal{B}_1$ and $\mathcal{B}_2$

The objective of this section is to analyse the actual state-of-the-art heuristics for the two variants considered in this paper, SA and MA. Moreover, this experimentation can help us to make a solid comparison of the existing approximate methods and to identify the most efficient ones to solve the SA and MA variants.

To evaluate the efficiency of the different heuristics, the $ARPD$ is computed following the Equation (2), where $\sum C_j^{T'}(h)$ is the value of the objective function found by each heuristic h and MIN is the minimum known solution for each instance. Moreover, the following indicator, RPT'_M (similarly as in Fernandez-Viagas and Framinan, 2015a), is computed in order to evaluate heuristics with different number of steps in their procedure:

$$RPT'_M = \frac{T_M - \overline{T}}{\overline{T}} \tag{3}$$

where T_M is the time (in seconds) required by heuristic method M to obtain a solution, and $\overline{T}$ is the average time needed by all the methods. Table 8 shows the average results, and it is organised as follows: column 1 and column 2 indicates the problem and the heuristics for each variant. Results for the benchmark of SA, $\mathcal{B}_1$, are presented in columns 3 to 10. More specifically, columns 3 to 8 show the values of $ARPD$ for the different number of jobs, column 9 indicates the Total $ARPD$ and column 10 the Total $ARPT'$. Results for the benchmark of MA, $\mathcal{B}_2$, are presented in the columns 11 to 18, and the columns are organised in the same way as for $\mathcal{B}_1$. The Pareto set for each variant is indicated in bold in Table 8.

Figure 3 shows the confidence intervals for the $ARPD$ of the heuristics designed to solve the SA variant, providing good results solving benchmarks $\mathcal{B}_1$ and $\mathcal{B}_2$. In the case of Figure 4, it shows the confidence intervals for the values of $ARPD$ of the best heuristics, designed to solve the MA variant, solving both benchmarks. The figures show the values of $ARPD$ per levels of n in $\mathcal{B}_1$ and per levels of n and m_2 in $\mathcal{B}_2$. From the analysis of Table 8 and Figures 3 and 4, some comments can be made:

- The heuristics designed to solve the SA variant performs similarly in both benchmarks, $\mathcal{B}_1$ and $\mathcal{B}_2$. Regarding the results shown in Table 8, there is slight variation in the $ARPD$ of the heuristics, with an average absolute difference between the $ARPD$ values equal to 1.83. Some reasons for this behaviour may be the way in which these heuristics have been adapted to solve the MA variant (see Talens et al., 2020 for the details) and the fact that parallel machines at the second stage flatten the significance of assembly operations. The best heuristics of this group are FAP, $TCK2$ and $G1$ with an $ARPD$ equal to 2.96, 5.72

and 7.20, respectively, for the SA variant, and 1.51, 5.40 and 8.92, respectively, for the MA variant. The performance of these heuristics is graphically shown in Figure 3.

- The heuristics designed for CO are not suitable to solve SA and MA, as they are not as good as some of the heuristics of the other two groups. However, it becomes clear that, for both variants, the best results are achieved by the heuristic $SHIFT_{k_{OPT}}$ with an $ARPD$ equal to 6.14, for SA and for MA.

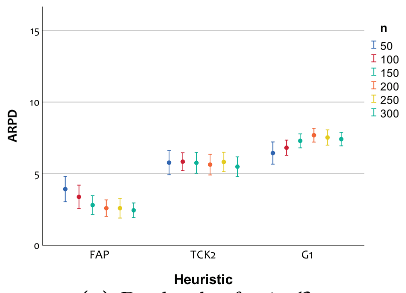
- Regarding the methods proposed to solve the MA variant, on the one hand, the heuristic CH_{MMA} yields a good performance in both benchmarks, with an $ARPD$ lower than 1 in both cases (see Table 8). Taking into account the results shown in Figure 4, regarding the number of jobs, CH_{MMA} shows a considerable difference between $n = 50$ and $n = 100$ in benchmark $\mathcal{B}_2$, while it performs similarly for n $\geq$150 in both benchmarks, $\mathcal{B}_1$ and $\mathcal{B}_2$. CH_{MMA} performs also similarly for m_2, being its $ARPD$ around 1 for all the levels. On the other hand, the best performance of the beam search-based constructive heuristic is achieved by $BSCH_V$ (x=2) and $BSCH_{MMA}$ (x=n) for $\mathcal{B}_1$, and $BSCH_V$ (x=2) and $BSCH_V$ (x=n) for $\mathcal{B}_1$, providing the lowest values of $ARPD$ and consuming less computational time. Regarding the number of jobs, these methods yield good results in both benchmarks, with an $ARPD$ lower than 0.5 for all the levels. Moreover, the worst $ARPD$ of the three heuristics is obtained for $n = 50$ and, as n increases, their performance become similar. Note that, with respect to m_2, the values of $ARPD$ of $BSCH_V$ (x=2), $BSCH_V$ (x=n) and $BSCH_{MMA}$ (x=n) increases, but in all the cases, it is lower than 1. Therefore, it can be pointed out that these versions are the best heuristics for solving the variants SA and MA.

- Comparing the results obtained from this evaluation and that carried out in Talens et al. (2020), several changes in the relative performance of some heuristics can be observed, possibly due to the lack of adequacy in the testbed used in the referred work. In this evaluation, heuristics $G1$ and $G4$ are more efficient than $G2$ and $G3$, while in Talens et al. (2020) $G2$ and $G3$ yield better results. Heuristic $A2$ performs better than $A1$ in benchmarks $\mathcal{B}_1$ and $\mathcal{B}_2$, while in Talens et al. (2020) the opposite happens. Regarding $S1$,

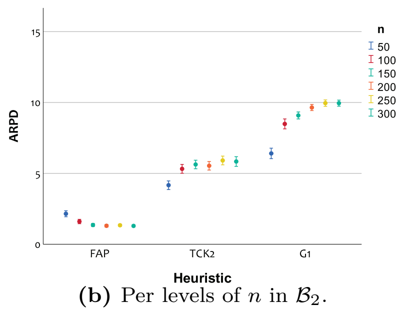
$S2$ and $S3$, in benchmarks $\mathcal{B}_1$ and $\mathcal{B}_2$, $S3$ is more efficient than $S1$ and $S2$, being similar the performance of the two latter. On the contrary, in Talens et al. (2020), $S2$ yields a better result than $S1$ and $S3$, and $S3$ is also more efficient than $S1$.

Finally, to summarize the main aspects of these results, it is worth noting that, regarding the number of jobs, the performance of the beam search-based constructive heuristics improves as n increases. On the contrary, all the other heuristics worsen their performance when n is equal to 250 and 300 than with lower number of jobs. Note that this behaviour had not been previously detected, as existing testbeds did not include instances with such large number of jobs. Taking into account this information, the methods recommended to solve the considered variants are:

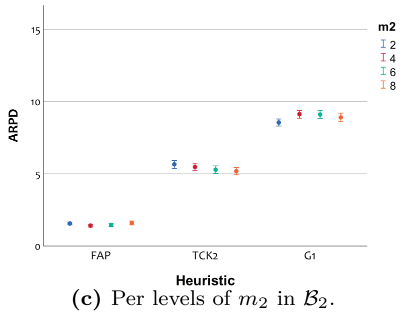
- For SA: the dispatching rules $STPT/SMPT$ and $S3$, and the heuristics $TCK2$, CH_{MMA} and $BSCH_V$ (x=2) and $BSCH_{MMA}$ (x=n).
- For MA: the dispatching rules $S1$ and $STPT/SMPT$, and the heuristics CH_{MMA} and $BSCH_V$ (x=2) and $BSCH_V$ (x=n).



(a) Per levels of n in $\mathcal{B}_1$.

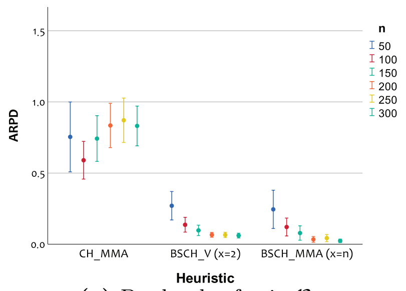


(b) Per levels of n in $\mathcal{B}_2$.

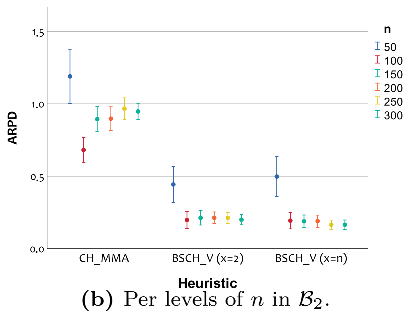


(c) Per levels of m_2 in $\mathcal{B}_2$.

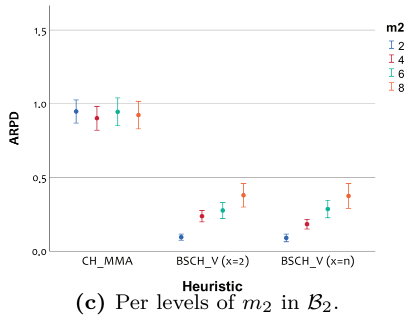
Figure 3: 95% Confidence Intervals for $ARPD$ of the best SA heuristics.



(a) Per levels of n in $\mathcal{B}_1$.



(b) Per levels of n in $\mathcal{B}_2$.



(c) Per levels of m_2 in $\mathcal{B}_2$.

Figure 4: 95% Confidence Intervals for $ARPD$ of the best MA heuristics.

To establish the statistical significance of the results, a Holm's procedure (Holm, 1979) is performed where each hypothesis is evaluated using a non-parametric Mann-Whitney test assuming a 95% confidence level (i.e. α=0.05). In Holm's test, the hypotheses are formed by a heuristic in the Pareto frontier and the closer dominated heuristic (in terms of $ARPD$) which provides higher $ARPT'$. The hypotheses are sorted in non-descending order of the p-values obtained in the Mann-Whitney test, evaluating the RPD of each heuristic. Each hypothesis H_i is rejected if $p \leq /(k - i + 1)$ where k is the total number of hypotheses. The objective of this procedure is to establish if there are significant differences between the two heuristics compared in each hypothesis. The results are shown in Table 9. It can be checked that, for both SA and MA, all the hypotheses are rejected, indicating that there are statistically significant differences between the heuristic considered, and validating the composition of the Pareto set.

4.3 Upper bounds

Finally, the experimentation carried out in this section is aimed towards the generation of reference upper bounds to be used by practitioners and researchers in order to compare their proposals. In order to obtain these bounds, the iterated greedy algorithm by Ruiz and Stützle (2007) has been run following the same procedure as Vallada et al. (2015) and Fernandez-Viagas and Framinan (2020). The stopping criterion used is $n \cdot m/2 \cdot 600/1000$ seconds, where m is equal to $m_1 + 1$. More specifically, this algorithm is run 20 times for each instance and the best value is taken as the reference upper bound of the instance. Then, the upper bounds are the minimum value between the solution provided by the IG and all the experimentation carried out in this section. The upper bounds are published as on-line materials in http://grupo.us.es/oindustrial/en/research/results/.

5 Conclusions

This paper considers the 2-stage assembly scheduling problem with several dedicated parallel machines in the first stage and one assembly machine in the second stage, denoted as SA, and also with several assembly machines in the second stage, denoted as MA. Due to the absence of a commonly accepted set of instances ensuring that the instances are representative of the problem

Table 8: Performance of the heuristics in benchmarks $\mathcal{B}_1$ and $\mathcal{B}_2$.

		$\mathcal{B}_1$									$\mathcal{B}_2$								
		n						Average			n						Average		
Problem	Heuristic	50	100	150	200	250	300	$ARPD$	$ARPT'$	$ACPU$	50	100	150	200	250	300	$ARPD$	$ARPT'$	$ACPU$
	$TCK1$	10.13	10.49	10.99	11.22	11.47	11.06	10.89	0.003	0.004	7.54	9.81	10.49	10.94	11.43	11.39	10.27	0.003	0.004
	$\boldsymbol{TCK2}$	5.77	5.84	5.76	5.63	5.82	5.49	**5.72**	**0.001**	**0.002**	4.17	5.32	5.63	5.54	5.91	5.84	5.40	0.001	0.002
	$A1$	33.02	35.06	36.72	37.99	38.35	38.09	36.54	0.003	0.003	24.53	30.96	33.57	35.43	36.85	37.34	33.11	0.003	0.004
	$A2$	7.35	7.46	7.64	7.97	7.75	7.71	7.65	0.003	0.003	5.35	6.90	7.38	7.81	7.98	8.00	7.24	0.003	0.004
	$\boldsymbol{S1}$	33.49	35.29	37.64	39.22	38.63	38.95	37.20	0.000	0.000	26.82	32.45	35.69	37.67	37.98	38.49	**34.85**	**0.000**	**0.000**
	$S2$	33.53	36.17	37.79	38.88	38.87	38.94	37.36	0.000	0.000	26.59	32.80	35.57	36.87	37.63	38.18	34.61	0.000	0.000
SA	$\boldsymbol{S3}$	8.60	9.24	10.10	10.19	10.09	9.75	**9.66**	**0.000**	**0.000**	6.57	8.79	9.62	9.86	10.21	10.25	9.22	0.000	0.000
	$G1$	6.44	6.81	7.29	7.69	7.53	7.41	7.20	0.002	0.003	6.41	8.49	9.08	9.64	9.95	9.95	8.92	0.003	0.003
	$G2$	33.03	35.55	36.93	38.13	38.36	38.20	36.70	0.003	0.003	24.85	31.28	33.81	35.69	37.17	37.61	33.40	0.003	0.003
	$G3$	33.98	34.93	36.82	38.02	39.09	38.78	36.94	0.002	0.003	25.21	31.01	34.09	35.63	36.93	37.78	33.44	0.003	0.003
	$G4$	6.64	6.88	7.29	7.67	7.51	7.39	7.23	0.002	0.003	6.42	8.49	9.09	9.65	9.96	9.95	8.93	0.003	0.003
	FAP	3.92	3.38	2.81	2.59	2.59	2.44	2.96	1.332	9.012	2.15	1.60	1.36	1.30	1.34	1.29	1.51	1.316	8.987
	ECT	6.44	6.81	7.29	7.69	7.53	7.41	7.20	0.474	3.095	6.41	8.49	9.08	9.64	9.95	9.95	8.92	0.488	3.156
	$\underline{STPT/SMPT}$	11.93	12.42	12.68	12.60	12.51	12.12	**12.38**	**0.000**	**0.000**	6.80	8.81	9.05	9.43	9.83	9.66	**8.93**	**0.000**	**0.000**
CO	$SHIFT_k$	3.91	5.51	6.55	6.98	7.07	7.08	6.18	1.737	11.825	2.87	6.19	7.61	8.48	9.09	9.21	7.24	1.776	12.011
	$SHIFT_{k_{OPT}}$	3.89	5.48	6.58	6.87	7.01	7.00	6.14	1.751	11.827	2.14	5.00	6.43	7.19	8.00	8.09	6.14	1.880	12.228
	SAK	10.09	10.91	11.56	11.50	11.50	11.32	11.15	5.138	35.860	4.06	5.91	6.14	6.47	6.96	6.79	6.06	5.163	35.284
	$\boldsymbol{CH_{MMA}}$	0.75	0.59	0.74	0.83	0.87	0.83	**0.77**	**0.003**	**0.003**	1.19	0.68	0.89	0.90	0.97	0.95	**0.93**	**0.003**	**0.004**
	$\boldsymbol{BSCH_V\ (x{=}2)}$	0.27	0.14	0.10	0.07	0.07	0.06	**0.12**	**0.494**	**2.544**	0.44	0.20	0.21	0.21	0.21	0.20	0.25	0.528	2.698
	$BSCH_V\ (x{=}n/10)$	0.32	0.17	0.10	0.07	0.08	0.06	0.13	1.080	5.645	0.46	0.17	0.22	0.20	0.22	0.20	0.25	1.152	5.973
	$BSCH_V\ (x{=}5)$	0.32	0.15	0.09	0.07	0.07	0.05	0.12	1.600	8.257	0.46	0.16	0.21	0.24	0.22	0.20	0.25	1.707	8.741
MA	$BSCH_V\ (x{=}10)$	0.31	0.17	0.11	0.07	0.08	0.05	0.13	2.166	10.983	0.46	0.17	0.24	0.21	0.20	0.21	0.25	2.309	11.628
	$BSCH_V\ (x{=}15)$	0.28	0.14	0.10	0.08	0.08	0.05	0.12	2.780	13.827	0.47	0.18	0.22	0.22	0.20	0.21	0.25	2.962	14.638
	$\boldsymbol{BSCH_V\ (x{=}n)}$	0.31	0.16	0.10	0.04	0.06	0.06	0.12	4.801	25.214	0.50	0.19	0.19	0.19	0.17	0.17	**0.23**	**5.053**	**26.401**
	$\boldsymbol{BSCH_{MMA}\ (x{=}n)}$	0.25	0.12	0.08	0.03	0.04	0.02	**0.09**	**0.900**	**4.963**	0.44	0.19	0.27	0.30	0.26	0.31	0.30	0.907	5.028
	$BSCH_{MMA}\ (x{=}n{+}n/2)$	0.24	0.12	0.08	0.04	0.04	0.02	0.09	2.724	15.419	0.46	0.18	0.23	0.24	0.24	0.24	0.27	2.735	15.575

		$\mathcal{B}_1$			
H_i	Hypothesis	p-value	Mann-Whitney	$\alpha/(k-i-1)$	Holm's Procedure
H_1^1	$STPT/SMPT{=}S1$	0.000	R	0.0125	R
H_2^1	$CH_{MMA}{=}G1$	0.000	R	0.0167	R
H_3^1	$BSCH_{MMA}$ ($x{=}n$)$=BSCH$ ($x{=}n/10$)	0.000	R	0.0250	R
H_4^1	$BSCH_{MMA}$ ($x{=}n{+}n/2$)$=BSCH_V$ ($x{=}15$)	0.000	R	0.0500	R
		$\mathcal{B}_2$			
H_i	Hypothesis	p-value	Mann-Whitney	$\alpha/(k-i-1)$	Holm's Procedure
H_1^2	$STPT/SMPT{=}S2$	0.000	R	0.0100	R
H_2^2	$STPT/SMPT{=}S3$	0.000	R	0.0125	R
H_3^2	$TCK2{=}A2$	0.000	R	0.0167	R
H_4^2	$CH_{MMA}{=}ECT$	0.000	R	0.0250	R
H_5^2	$BSCH_V$ ($x{=}2$)$=BSCH_{MMA}$ ($x{=}n$)	0.027	R	0.0500	R

Table 9: Mann-Whitney's procedure. (R indicates that the hypothesis can be rejected).

under study, there is a need of hard instances specifically designed for the variants *SA* and *MA*. Therefore, two new benchmarks of instances are designed according to the following characteristics found in the literature: empirical hardness, adequacy, exhaustiveness, and amenability for statistical analysis.

Regarding the procedure followed to design the benchmarks, first, different scenarios, depending on the relation between stages have been generated using a parameter α. Two preliminary testbeds for each variant have been generated and both exact and approximate methods have been applied to identify the adequacy of the instances of the problem under study. Next, with the most suitable value of α identified in the previous analysis, two additional preliminary testbeds, with 24,000 and 96,000 instances, respectively, have been generated to determine the empirical hardness of the testbeds. A lower bound of each instance has been computed and the iterated greedy algorithm has been applied to solve the testbeds. Then, the instances whose solution founded by the IG is further from the theoretical lower bound are selected to form the two new benchmarks of 240 instances for *SA*, and 960 instances for *MA*. With this methodology, it is ensured that the characteristics of adequacy and empirical hardness are achieved. The exhaustiveness and amenability are also fulfilled by considering a large number of instances, equidistant levels, and by combining all the levels of the parameters to obtain the instances.

All the heuristics designed to solve the variants *SA* and *MA* and the related problems *CO*, *SM* and *PM* have been tested and compared in the new benchmarks. On the one hand, the results obtained show that most of the heuristics designed to solve the SA variant perform slightly better, in average, solving the MA variant than the SA variant. On the other hand, the beam search-based constructive heuristics designed in Talens et al. (2020) yields the best results in terms of $ARPD$, for both benchmarks. Finally, this computational experimentation has led us to determine the best heuristics to solve each one of the considered variants. For SA, the group of the most efficient heuristics is formed by the dispatching rules $STPT/SMPT$ and $S3$, and the heuristics $TCK2$, CH_{MMA}, $BSCH_V$ (x=2) and $BSCH_{MMA}$ (x=n); and for MA, by the dispatching rules $S1$ and $STPT/SMPT$, and the heuristics CH_{MMA} and $BSCH_V$ (x=2) and $BSCH_V$ (x=n). Regarding the most efficient heuristics for the MA variant, there are some differences with respect to the conclusions obtained in the recent work by Talens et al. (2020), showcasing the importance of the set of instances selected in a state-of-the-art study.

Note that, in designing the benchmarks, the total completion time has been adopted as objective, since it was, by far, the most studied criterion for the 2-stage assembly problems. Although this does not preclude using the benchmarks for different criteria, in view of the high influence of the due dates in scheduling problems, new instances considering due-date oriented objective functions (as e.g $\sum(w_j)U_j$ and $\sum(w_j)T_j$) could be proposed, by either adding the due dates of each job to the present instances or generating a completely new set benchmark. Furthermore, the relationship between the problem under consideration and related problems would be an interesting research line to explore when different data or constraints are used, specially regarding the influence of using different distribution of processing times in the first stage, or the addition of setup times.

References

Ahmadi, R., Bagchi, U., and Roemer, T. A. (2005). Coordinated scheduling of customer orders for quick response. *Naval Research Logistics*, 52(6):493–512.

Al-Anzi, F. S. and Allahverdi, A. (2006a). A Hybrid Tabu Search Heuristic for the Two-Stage Assembly Scheduling Problem. *International Journal of Operations Research*, 3(2):109–119.

Al-Anzi, F. S. and Allahverdi, A. (2006b). Empirically discovering dominance relations for scheduling problems using an evolutionary algorithm. *International Journal of Production Research*, 44(22):4701–4712.

Al-Anzi, F. S. and Allahverdi, A. (2007). A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. *European Journal of Operational Research*, 182(1):80–94.

Al-Anzi, F. S. and Allahverdi, A. (2012). Better Heuristics for a Two-Stage Multi- Machine Assembly Scheduling Problem to Minimize Total Completion Time Better Heuristics for a Two-Stage Multi-Machine Assembly Scheduling Problem to Minimize Total Completion Time. *International Journal of Operations Research*, 9:66–75.

Al-Anzi, F. S. and Allahverdi, A. (2013). An artificial immune system heuristic for two-stage multi-machine assembly scheduling problem to minimize total completion time. *Journal of Manufacturing Systems*, 32(4):825–830.

Allahverdi, A. (2000). Minimizing mean flowtime in a two-machine flowshop with sequence-independent setup times. *Computers and Operations Research*, 27(2):111–127.

Allahverdi, A. and Al-Anzi, F. (2012). A new heuristic for the queries scheduling problem on distributed database systems to minimize mean completion time. In *Proceedings of the 21st International Conference on Software Engineering and Data Engineering, SEDE 2012*.

Allahverdi, A. and Al-Anzi, F. S. (2006). A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. *Computers and Operations Research*, 33(4):1056–1080.

Allahverdi, A. and Al-Anzi, F. S. (2009). The two-stage assembly scheduling problem to minimize total completion time with setup times. *Computers and Operations Research*, 36(10):2740–2747.

Allahverdi, A. and Aydilek, H. (2015). The two stage assembly flowshop scheduling problem to minimize total tardiness. *Journal of Intelligent Manufacturing*, 26(2):225–237.

Blocher, J. D. and Chhajed, D. (2008). Minimizing customer order lead-time in a two-stage assembly supply chain. *Annals of Operations Research*, 161(1):25–52.

Cheng, T. E. and Wang, G. (1999). Scheduling the fabrication and assembly of components in a two-machine flowshop. *IIE Transactions*, 31(2):135–143.

Conway, R. W., Miller, L. W., and Maxwell, W. L. (1967). *Theory of scheduling.* Addison-Wesley Pub. Co.

Fattahi, P., Hosseini, S. M. H., and Jolai, F. (2013). A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 65(5):787–802.

Fernandez-Viagas, V. and Framinan, J. (2020). Design of a testbed for hybrid flow shop scheduling with identical machines. *Computers and Industrial Engineering*, pages 1–32.

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

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

Framinan, J. M. and Perez-Gonzalez, P. (2017a). New approximate algorithms for the customer order scheduling problem with total completion time objective. *Computers and Operations Research*, 78:181–192.

Framinan, J. M. and Perez-Gonzalez, P. (2017b). The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic. *Computers and Operations Research*, 88:237–246.

Framinan, J. M., Perez-Gonzalez, P., and Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research*, 273:401–417.

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

Hall, N. G. and Posner, M. E. (2001). Testing With Machine Scheduling Applications. *Operations Research*, 49(7):854–865.

Hatami, S., Ruiz, R., and Andrés-Romano, C. (2015). Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 169:76–88.

Holm, S. (1979). Board of the Foundation of the Scandinavian Journal of Statistics. 6(2):65–70.

Hwang, F. J. and Lin, B. M. (2012). Two-stage assembly-type flowshop batch scheduling problem subject to a fixed job sequence. *Journal of the Operational Research Society*, 63(6):839–845.

Hwang, F. J. and Lin, B. M. (2018). Survey and extensions of manufacturing models in two-stage flexible flow shops with dedicated machines. *Computers and Operations Research*, 98:103–112.

Lee, C.-Y., Cheng, T. C. E., and Lin, B. M. T. (1993). Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science*, 39(5):616–625.

Lee, I. S. (2018). Minimizing total completion time in the assembly scheduling problem. *Computers and Industrial Engineering*, 122:211–218.

Leung, J. Y., Li, H., and Pinedo, M. (2008). Scheduling orders on either dedicated or flexible machines in parallel to minimize total weighted completion time. *Annals of Operations Research*, 159(1):107–123.

Leung, J. Y. T., Li, H., and Pinedo, M. (2005). Order scheduling in an environment with dedicated resources in parallel. *Journal of Scheduling*, 8(5):355–386.

Liao, C. J., Lee, C. H., and Lee, H. C. (2015). An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. *Computers and Industrial Engineering*, 88(313):317–325.

Lin, B. M., Lu, C. Y., Shyu, S. J., and Tsai, C. Y. (2008). Development of new features of ant colony optimization for flowshop scheduling. *International Journal of Production Economics*, 112(2):742–755.

Lin, W.-C. (2018). Minimizing the makespan for a two-stage three-machine assembly flow shop problem with the sum-of-processing-time based learning effect. *Discrete Dynamics in Nature and Society*, 2018.

Mozdgir, A., Fatemi Ghomi, S. M. T., Jolai, F., and Navaei, J. (2013). Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times. *International Journal of Production Research*, 51(12):3625–3642.

Navaei, J., Fatemi Ghomi, S. M. T., Jolai, F., Shiraqai, M. E., and Hidaji, H. (2013). Two-stage flow-shop scheduling problem with non-identical second stage assembly machines. *International Journal of Advanced Manufacturing Technology*, 69(9-12):2215–2226.

Nejati, M., Mahdavi, I., Hassanzadeh, R., and Mahdavi-Amiri, N. (2016). Lot streaming in a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint. *Journal of Industrial and Production Engineering*, 33(7):459–471.

Pan, Q.-K., Gao, L., Xin-Yu, L., and Framinan, J. (2019). Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing Journal*, 81.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems.* Springer Publishing Company, Incorporated, 3rd edition.

Potts, C. N., Sevast'janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., and Zwaneveld, C. M. (1995). The Two-Stage Assembly Scheduling Problem: Complexity and Approximation. *Operations Research*, 43(2):346–355.

Roemer, T. A. (2006). A note on the complexity of the concurrent open shop problem. *Journal of Scheduling*, 9(4):389–396.

Roemer, T. A. and Ahmadi, R. H. (1997). The Complexity of Scheduling Customer Orders. In *IN-FORMS conference 1997, Dallas.*

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Sheikh, S., Komaki, G., and Kayvanfar, V. (2018). Multi objective two-stage assembly flow shop with release time. *Computers and Industrial Engineering*, 124:276–292.

Shi, Z., Huang, Z., and Shi, L. (2018). Customer order scheduling on batch processing machines with incompatible job families. *International Journal of Production Research*, 56(1-2):795–808.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66.

Sung, C. S. and Kim, H. A. (2008). A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times. *International Journal of Production Economics*, 113(2):1038–1048.

Sung, C. S. and Yoon, S. H. (1998). Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines. *International Journal of Production Economics*, 54(3):247 – 255.

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

Talens, C., Fernandez-Viagas, V., Perez-Gonzalez, P., and Framinan, J. (2020). New efficient constructive heuristics for the two-stage multi-machine assembly scheduling problem. *Computers and Industrial Engineering*, 140.

T'kindt, V., Monmarcheé, N., Tercinet, F., and Lauügt, D. (2002). An Ant Colony Optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research*, 142(2):250–257.

Tozkapan, A., Kirca, Ö., and Chung, C. S. (2003). A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. *Computers and Operations Research*, 30(2):309–320.

Vallada, E., Ruiz, R., and Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677.

Wagneur, E. and Sriskandarajah, C. (1993). Openshops with jobs overlap. *European Journal of Operational Research*, 71(3):366–378.

Wu, C. C., Chen, J. Y., Lin, W. C., Lai, K., Liu, S. C., and Yu, P. W. (2018). A two-stage three-machine assembly flow shop scheduling with learning consideration to minimize the flowtime by six hybrids of particle swarm optimization. *Swarm and Evolutionary Computation*, 41(February):97–110.

Zhang, Y., Zhou, Z., and Liu, J. (2010). The production scheduling problem in a multi-page invoice printing system. *Computers and Operations Research*, 37(10):1814–1821.