Ph. D. Thesis

The 2-stage Assembly Scheduling Problem

by

Carla Talens Fayos

supervised by

PhD. Víctor Fernández-Viagas Escudero

PhD. Paz Pérez González

Sevilla, Spain

November 2022

A mis abuelos,
Juan y Matilde

Acknowledgements

This thesis is the result of all the work carried out over the last four years. It would not have been possible without the support and guidance of different people whom I would like to thank in these lines.

Paz and Víctor, my warmest thanks for all the support and motivation during all these years. Thanks for sharing your knowledge and experience with me, and also thanks for all the advice. I am sure that without your supervision I would not have been able to get to where I am today.

I also want to be thankful with Jose and all the members of the Industrial Management Research Group of the University of Seville for giving me the opportunity to carry out this thesis and be part of the group. All the knowledge acquired during these years is also thanks to all of you.

I have also received help and knowledge from other leading researchers during my research stays. I am sincerely grateful to Antonio Costa, Jorge M.S. Valente and Rubén Ruiz for the support and availability during my stays with them. Here, I would like to highlight the role of Rubén Ruiz, since he was my professor in the Master's degree and thanks to him I started this thesis and my research career.

En estas líneas también quiero dar las gracias a mi familía, porque, a pesar de la distancia durante estos años, siempre los he sentido cerca. Gracias a mi madre, por todo el esfuerzo, apoyo y dedicación para haber llegado hasta aquí. Gracias a mi hermano, por estar siempre. Gracias a mi abuela, por mimarme siempre, y a mi abuelo porque, allá donde esté, seguro que estará orgulloso de mí.

También quiero dar las gracias a Fernando, por aparecer y, sobre todo, permanecer. Y, por último, agradecer a mis amigas todo el apoyo durante estos años.

Abstract

Today, manufacturing systems face new global challenges that require flexibility and quick reactions. To remain competitive, companies must also focus on the minimisation of time, as it leads to cost reductions, shorter delivery times, and a more agile and flexible response. In this context, improving the scheduling is of great importance. This is why, in this doctoral thesis, different ways to improve the scheduling of the considered problem are discussed in this thesis.

More specifically, we address the 2-stage assembly scheduling problem. This problem consists of two stages: the components of the product are processed in the first stage and then assembled in the second stage. In the first stage, we consider dedicated parallel machines and, in the second stage, one or several parallel assembly machines. The purpose of this thesis is to make several contributions in different aspects of the problem. First, we review in detail the problem under consideration. Second, there are some less studied variants of the problem that remain unanalysed and are considered in this thesis. Furthermore, there is no a common set of instances in which the different optimization methods developed to solve the problem can be evaluated and compared. Finally, additional constraints are also considered for the 2-stage assembly scheduling problem, as they have never been addressed.

Contents

Part I

Preliminaries

Chapter 1

Introduction

1.1 Motivation for the thesis

Global competition that makes the market more competitive and dynamic requires manufacturing companies to be flexible and react quickly to a constantly changing market (Sanchez, 1995; Manzini et al., 2004). To survive in this environment, companies do not have to focus only on the quality of their products, since there is another key factor in many environments. Consider how important time is in many industries regarding its minimisation in transportation, manufacturing, services, or communications, among others. For a company involving these procedures, time is of great importance if it wants to gain a competitive advantage with respect to competitors. Basically, time reduction is important for two reasons: on the one hand, it usually leads to cost reductions for the company and, on the other hand, it allows both shorter delivery times to customers and a more agile and flexible response. In this regard, production management is a managerial process in which a large number of decisions are made over time to ensure the delivery of goods with the highest quality, the lowest cost, and the lowest lead time (Framinan et al., 2014). These decisions differ, among other issues, in their impact on the company, their scope, and the time period for taking them. They range from strategic, high-impact, long-range decisions, such as deciding if a new product is manufactured in a certain factory or not, to short-term, low-level, small-impact decisions, such as the order in which a product will be manufactured in a certain machine in the shop floor. Among these decisions, the scheduling process plays an important role in the operational phase of manufacturing systems in terms of

rapid response to the dynamic market and the improvement of system performance (Alemão et al., 2021).

Manufacturing scheduling establishes the schedules of the resources along the horizon under consideration in order to fulfil the customers' requests. Scheduling is carried out as a part of a more complex process. On the one hand, it uses as input the results of a planning plan, where the set of products to be manufactured by the company along with their real (or expected) demand, among other issues, are included. On the other hand, the execution of a schedule may need to be reviewed or modified due to the often highly variable conditions on the shop floor. Consequently, manufacturing scheduling is integrated into a set of managerial decisions known as production management.

In real manufacturing scenarios, the difficulty of scheduling problems becomes extremely complex, since schedulers must consider the specific constraint and the objective of the shop when determining the best schedule for the shop floor (Framinan et al., 2014). Another important aspect is that most scheduling decisions must be made in short time intervals due to the rapid response required by the manufacturing system. Thus, the development of fast and efficient procedures is of great importance in solving this manufacturing scheduling problem.

In fact, industries adopt several processing layouts to manufacture their products, such as e.g., parallel machines, flowshop, jobshop, etc. Among the different layouts, in this thesis, we address the 2-stage Assembly Scheduling Problem (2-ASP in the following), where many products made up of several components are manufactured, the components have to be produced in the previous stages and then assembled in a later stage. This problem is receiving an increasing attention from researchers due to different reasons: First, the 2-stage assembly layout has many applications in industry (Sheikh et al., 2018), such as 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). Second, while the two-stage assembly problem with only one machine in the second stage has been widely discussed in the literature (see the recent review on the topic by

Framinan et al., 2019), to the best of our knowledge, the problem considering several identical assembly machines is common in companies, although it has not received attention so far. Therefore, the state-of-the-art regarding solution methods for the problem under consideration is either at an early stage or very scarce, and efficient approximate algorithms for this problem need to be found.

After addressing both problems presented above, we also focus on a quite common unavailability constraint, where all machines stop periodically after a given time interval and jobs cannot be processed during these periods in which the machines are not available. In many companies, we can find that resources are not continuously available due to different causes (see, e.g., Ji et al., 2007; Low et al., 2010b; Perez-Gonzalez et al., 2020; Yu et al., 2014), such as end of shifts, hours/days off, periodic maintenance activities, etc. Due to the importance of this constraint in real manufacturing, it has been widely studied for different layouts, e.g., single machine scheduling problem (see e.g., Ángel-Bello et al., 2011; Yazdani et al., 2018; Perez-Gonzalez and Framinan, 2018); parallel machines scheduling problem (see e.g. Kaabi and Harrath, 2019); and flowshop scheduling problem (see e.g. Perez-Gonzalez et al., 2020), among others.

Due to the reasons introduced above and the analysis of the state-of-the-art for the problem under consideration (see Chapter 4), there is an opportunity to improve the current state-of-the-art in the problems addressed in this thesis by designing a standard and representative benchmark to test the different algorithms; proposing new approximate solution procedures for the most common objectives and comparing them with the state-of-the-art algorithms; and, finally, considering different scheduling constraints to capture more realistic situations.

Therefore, in an attempt to accomplish the previous objectives, in this thesis, we focus on the following scheduling problems:

- The 2-stage assembly scheduling problem (or 2-ASP) with total completion time objective (i.e., the sum of the completion time of all jobs).
- The 2-stage assembly scheduling problem with periodic maintenance (or 2-ASP-pm) with makespan (i.e., the maximum completion time) and total completion time objective.

1.2 Objectives and outline of the thesis

As stated in the previous section, the goal of this thesis is to provide further insights into the 2-ASP. After reviewing the literature related to the 2-ASP, we can identify different weaknesses and areas for improvement. First, we can find an extensive literature related to the problem with only one assembly machine in the second stage; however, to the best of our knowledge, there are very few works where the problem with several machines in the second stage is addressed. Furthermore, with respect to the objectives, the makespan is the most common objective considered in this layout, remaining unanalysed others such as the total completion time. Second, there is no a common set of instances in which the different algorithms developed to solve the problem can be evaluated and compared. And finally, the 2-ASP with additional constraints has not yet been addressed, specially, the problem where resources are not continuously available. To make different contributions in these areas, the following general research objectives are defined:

GO1 To review the 2-ASP literature in order to identify the problems addressed in this thesis, the most relevant objectives, and the most interesting constraints.

GO2 To design a new benchmark of hard and exhaustive instances for testing the efficient approximate algorithms in the literature.

GO3 To propose faster and more efficient approximate algorithms to solve the 2-ASP with total completion time objective, based on the conclusions obtained from GO1.

GO4 To demonstrate the efficiency and good performance of the solution procedures developed in GO3.

GO5 To extend the goals GO3 and GO4 to some constrained 2-ASP based on real manufacturing environments.

To achieve these objectives, the structure of the thesis, graphically summarised in Figure 1.1, is organised in four parts as follows:

- Part I: Preliminaries. This part is divided into three chapters. In Chapter 1, we introduce this thesis, describe its main objectives, and discuss its main contributions. In Chapter 2, we state the problems under consideration and present their

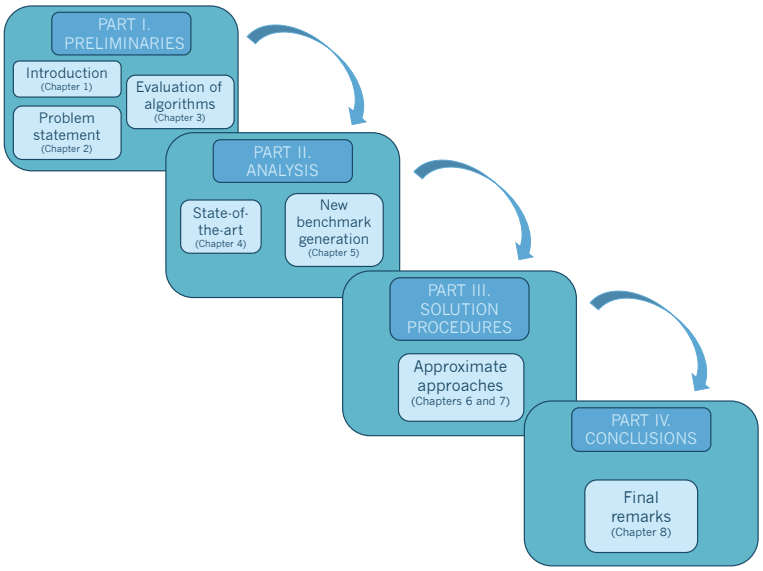

Figure 1.1: Structure of the thesis.

mathematical models, and in Chapter 3, we discuss the performance indicators used to evaluate the solution procedures, the experimental conditions, and the different sets of instances used in the computational experiments to compare the solution procedures presented throughout this thesis.

- Part II: Analysis. In this part, we analyse the problem in detail in two chapters. In Chapter 4, handling GO1, the main contributions in the literature are reviewed for the traditional problem (the 2-ASP) and for the 2-ASP with periodic maintenance (2-ASP-pm). In Chapter 5, we analyse the relationship between the 2-ASP and other related scheduling problems and propose a comprehensive benchmark for the 2-ASP addressing GO2.

- Part III: Solution Procedures. Here, we propose new efficient algorithms to solve 2-ASP with various objectives, as well as with other scheduling constraints. This part is divided into three chapters and deals with the general objectives GO3, GO4, and GO5. More specifically, in Chapter 6, we propose two efficient constructive heuristics to minimise the total completion time. In Chapter 7, we address the problem with periodic maintenance and the objective of minimising the makespan by proposing several efficient constructive and

composite heuristics.

- Part IV: Conclusions. Finally, in this part, we discuss the conclusions and results obtained and present future research lines.

Chapter 2

Problem statement

2.1 Introduction

In this chapter, we define and model the problems under study and also explain the different layouts related to the problems addressed in this thesis. More specifically, in Section 2.2 we describe the problem under consideration and analyse its different variants. Additionally, we present the notation for the assembly problems. Then, in Section 2.3 we present the mathematical models for the 2-ASP with one assembly machine and several machines in the second stage. Finally, in Section 2.4, we present the mathematical model for the 2-ASP-pm with one assembly machine in the second stage.

2.2 Problem description and notation

Among the different assembly scheduling problems, in this thesis, we first focus on the so-called 2-stage assembly scheduling problem (2-ASP). In this problem, there are m_1 ($m_1 > 1$, fixed) dedicated parallel machines (DPm_1) in the first stage (to manufacture each operation of job j on machine i, denoted by $\{O_{ij}^1\}$) and m_2 ($m_2 \geq 1$, fixed) parallel assembly machines (to assemble each assembly operation of job j, denoted by $\{O_j^2\}$) in the second stage. A job j has a processing time p_{ij} on machine i in the first stage and an assembly processing time at_j in the second stage. According to Framinan et al. (2019), this layout is denoted by $DPm_1 \rightarrow 1$, when there is one single assembly machine, or by $DPm_1 - > Pm_2$, when there are m_2 identical parallel machines in

the assembly stage. These assembly layouts are related to several traditional layouts from the literature as follows: (1) the layout of customer order with dedicated parallel machines, denoted by $DPm \rightarrow 0$, is tantamount to the one under consideration if the processing times of the jobs in the assembly stage are zero; (2) the single machine layout is similar to $DPm \rightarrow 1$ if the processing times in the second stage are much higher than those in the first stage; (3) similarly, the layout of traditional parallel machines is connected to $DPm \rightarrow Pm$; (4) finally, the 2-machine flowshop layout, denoted $F2$ is a particular case of the 2-ASP problem if $m_1 = 1$ and $m_2 = 1$. The layouts of the problems under study and related problems are shown in Figures 2.1 and 2.2.

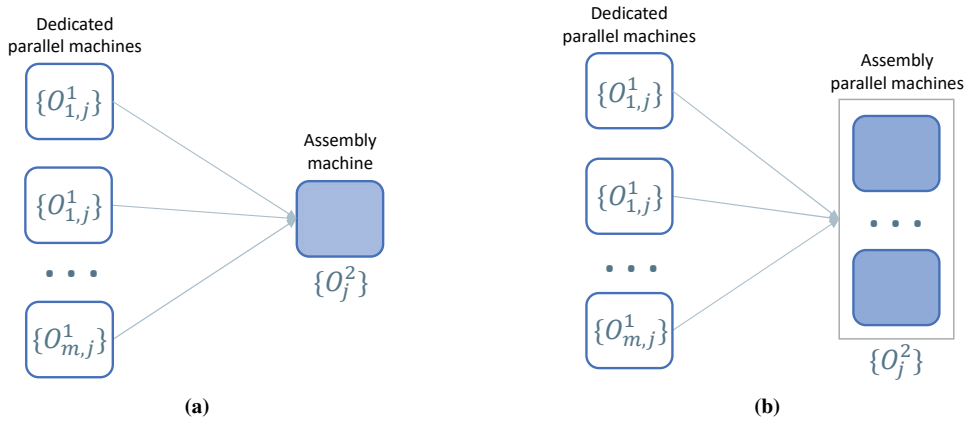

Figure 2.1: Layouts of the considered problems. Figure 2.1a: $DPm \rightarrow 1$ layout. Figure 2.1b: $DPm_1 \rightarrow Pm_2$ layout.

In this thesis, among the objectives addressed in the 2-stage assembly scheduling problems, we consider that with the objective of minimising the total completion time. The decision problem consists on scheduling the jobs in the two stages, so the sum of the completion times of the jobs is minimised. The problem with one assembly machine (labelled as SA in the following) is denoted as $DPm \rightarrow 1||\sum C_j$ by Graham et al. (1979) and Framinan et al. (2019). The problem 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.

After addressing both problems presented above, we also consider the 2-stage assembly scheduling problem with periodic maintenance constraint and the objective of minimising the makespan. This constraint is defined in the related literature as

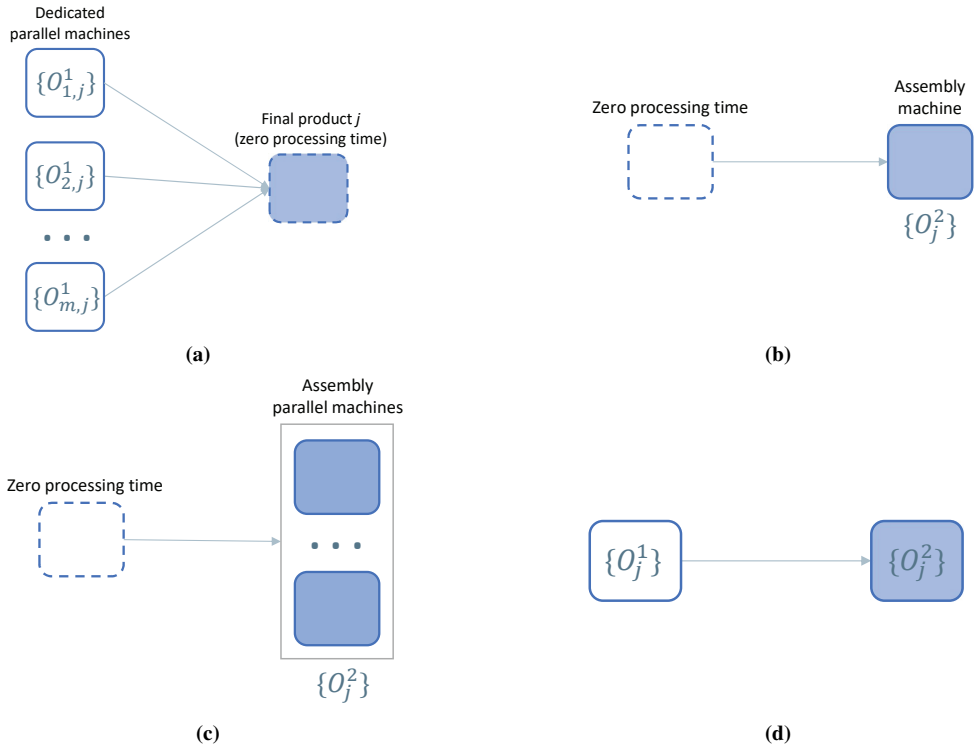

Figure 2.2: Layouts of the related problems. Figure 2.2a: $DPm \rightarrow 0$ layout. Figure 2.2b: 1 layout. Figure 2.2c: Pm layout. Figure 2.2d: $F2$ layout.

periodic maintenance (see, e.g., Perez-Gonzalez and Framinan, 2018; Perez-Gonzalez et al., 2020). Regarding the periodic maintenance constraint, we consider that jobs are non resumable (*nr*), i.e., preemption is not allowed and then a job must be completely processed in a unique availability period, being this feature simply denoted by *nr-pm* (see e.g., Low et al., 2010a; Perez-Gonzalez et al., 2020). Following the notation by Graham et al. (1979), the problem under consideration can be denoted by $DPm \rightarrow 1|nr-pm|C_{\max}$.

The $DPm \rightarrow 1|nr-pm|C_{\max}$ problem consists of scheduling n jobs in a two-stage layout. The scheduling horizon is formed by availability periods, denoted by bins[1], of length T, where jobs are sequenced, and in the non-availability periods of t time

[1]In the literature, the availability periods are traditionally denoted by batches, blocks, working shifts or bins.

units where the machines are not available to process any job. Each job has $m + 1$ non-resumable operations. In the first stage, there are m $(i = 1, ..., m)$ dedicated parallel machines, in which the first m operations are carried out. Every job has to be processed in every machine i, being p_{ij} the processing time of job j in machine i. In the assembly stage, there is one machine that executes the last $(m + 1)th$ operation and whose processing time is denoted by at_j.

As mentioned previously, the problems under study, the 2-ASP and 2-ASP-pm, have never been addressed so far. Therefore, it is necessary to review the literature on related problems (see Chapter 4), considering the related layouts explained above.

For the 2-ASP ($DPm \rightarrow 1||\sum C_j$ or SA and $DPm \rightarrow Pm||\sum C_j$ or MA), the following related problems are identified:

- the $DPm \rightarrow 0||\sum C_j$ problem, denoted CO, is tantamount to the SA/MA problem.
- the $1||\sum C_j$ and the $Pm||\sum C_j$ problem, denoted SM and PM, respectively, are similar to the SA and MA problems, respectively.
- the $F2||\sum C_j$ problem, denoted $F2$, is a particular case of the SA problem if $m_1 = 1$ and $m_2 = 1$. Additionally, since the $F2||\sum C_j$ problem is strongly NP-hard (Garey et al., 1976), the problem under consideration is also strongly NP-hard.

For the 2-ASP-pm ($DPm \rightarrow 1|nr - pm|C_{\max}$), we identify the following related problems:

- the $1|nr - pm|C_{\max}$ problem is a special case of the 2-ASP-pm when $m = 0$. Since this problem is NP-hard (Lee, 1996), the 2-ASP-pm can also be classified as NP-hard.
- the $DPm \rightarrow 1||C_{\max}$ problem is equivalent to the 2-ASP-pm if the availability period is high enough.
- the $F2|nr - pm|C_{\max}$ problem, to which our problem can be reduced when $m = 1$.

In Table 2.1, the different problems considered in this thesis, their contributions, and the sections where they are addressed are shown. Now, let us introduce a common

Problem	Benchmark	Exact methods	Approximate methods	Computational evaluation
$DPm_1 \rightarrow 1\|\|\sum C_j$	Section 5.3	Section 2.3	Section 6.2 and 6.3	Section 6.5
$DPm_1 \rightarrow Pm_2\|\|\sum C_j$	Section 5.3	Section 2.3	Section 6.2 and 6.3	Section 6.5
$DPm_1 \rightarrow 1\|nr-pm\|C_{\max}$	-	Section 2.4	Section 7.4 and 7.5	Section 7.6

Table 2.1: Summary of the problems addressed in this thesis and their contributions.

j	p_{1j}	p_{2j}	p_{3j}	at_j
1	12	13	14	14
2	12	13	10	11
3	9	11	18	13
4	3	12	9	14

Table 2.2: Processing times of the jobs in both stages.

example to explain the three scheduling problems under study. In Table 2.2 we show the processing times of four jobs in a layout composed of three dedicated parallel machines in the first stage and one assembly machine. In Figure 2.3, a Gantt chart is shown, where four jobs are scheduled in a layout $DPm \rightarrow 1$. It also indicates how the makespan and total completion time objectives are calculated, where C_1, C_2, C_3 and C_4 are the completion times of each job (see Equation 2.3). The case with several assembly machines in the second stage ($m_2 \geq 2$) is shown in Figure 2.4. It can be observed that, as there is an additional assembly machine, jobs 2 and 4 can be assembled earlier. In Figure 2.5, the four jobs are scheduled in $DPm \rightarrow 1$ with periodic maintenance. It can be seen that each job that does not fit in the current bin, it is assigned to a new bin.

2.3 Mathematical model of the 2-ASP with total completion time objective

A solution for this problem can be given by a sequence of jobs indicating the order in which the jobs are processed (see e.g., Al-Anzi and Allahverdi, 2006a and Al-Anzi and Allahverdi, 2012). Therefore, given a sequence, let $[j]$ denote the job processed

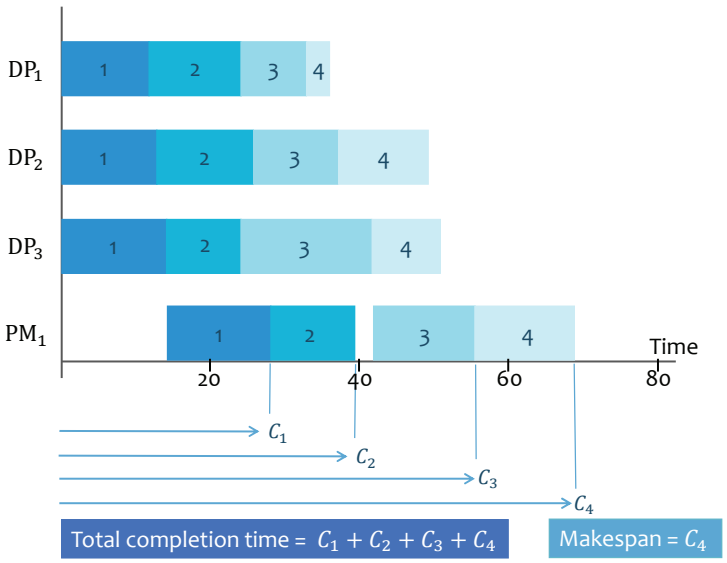

Figure 2.3: Gantt chart of $DPm \rightarrow 1$ layout with different objectives.

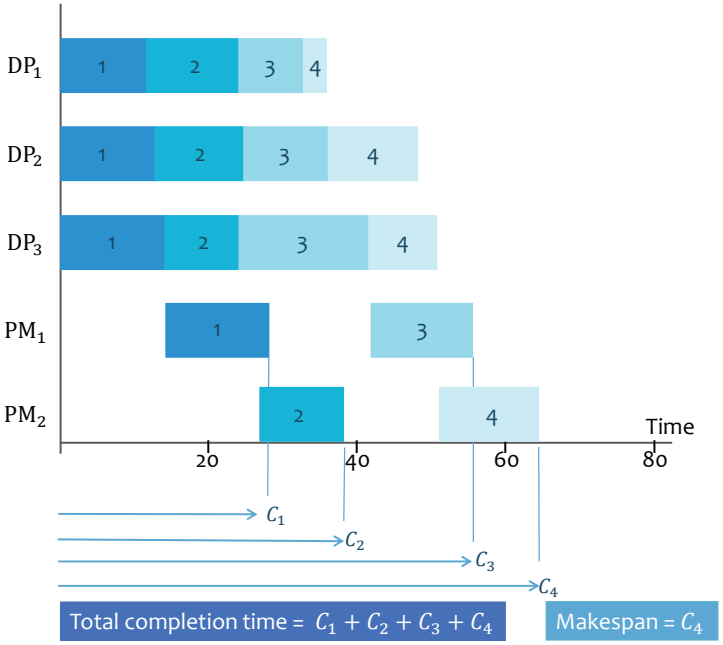

Figure 2.4: Gantt chart of $DPm \rightarrow Pm$ layout with different objectives.

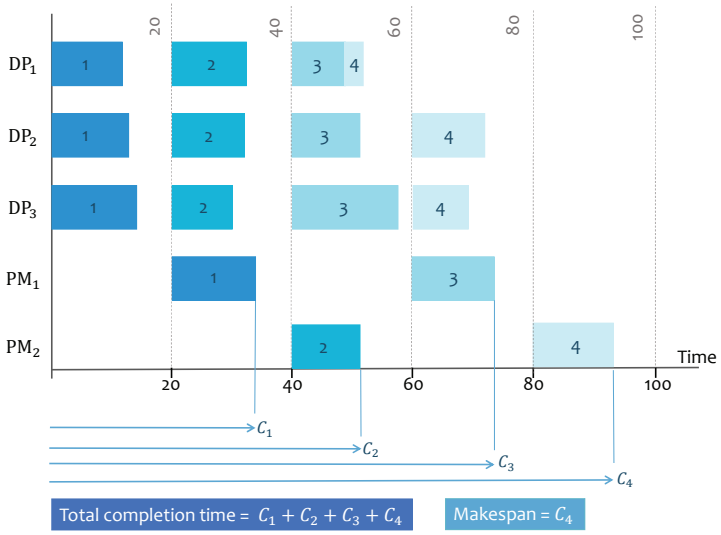

Figure 2.5: Gantt chart of $DPm \rightarrow 1$ layout with periodic maintenance and different objectives.

in position j in the sequence. The maximum completion time of job in position j in the first stage, denoted by $C1_j$, can be computed as follows:

$$C1_j = \max_{i=1,\ldots,m_1} \left\{ \sum_{k=1}^{j} p_{i[k]} \right\} \tag{2.1}$$

The completion time of each job in position j is recursively computed by Equation (2.2) for each assembly machine $i = 1, \ldots, m_2$, using the variable $C_{i,j-1}$ to denote the completion time in machine i of the jobs in the sequence until job in position j:

$$C_{ij} = \begin{cases} \max\{C_{i^*,j-1}, C1_j + at_{[j]}\} & \text{if } i = i^* \text{ with } i^* = \arg\min_{i=1,\ldots,m_2}\{C_{i,j-1}\} \\ C_{i,j-1} & \text{otherwise} \end{cases} \tag{2.2}$$

where $C_{i,0} = 0, \ \forall i = 1, \ldots, m_2$. Then, C_j the completion time of the job processed in position j of the sequence can be computed by Equation (2.3).

$$C_j = C_{i^*,j} \text{ if } i = i^* \text{ with } i^* = \arg\min_{i=1,\ldots,m_2}\{C_{i,j-1}\} \tag{2.3}$$

The mathematical model of the $DPm \rightarrow 1||\sum C_j\ (SA)$ variant is presented below

and the notation is shown in Table 2.3. The objective function and constraints can be written as follows:

Indexes	Definition
j	Index for jobs: $j \in \{1, \ldots, n\}$.
k	Index for the position in the sequence: $k \in \{1, \ldots, n\}$.
i	Index for machines in the first stage: $i \in \{1, \ldots m_1\}$.
Parameters	**Definition**
p_{ij}	Processing time of job j in machine $i \in \{1, \ldots, m\}$.
at_j	Processing time of job j in machine $m + 1$.
Variables	**Definition**
X_{jk}	1 if job j is placed in position k of the sequence at the first stage; 0, otherwise.
$C1_{ik}$	Completion time in machine i of the first stage of job in position k.
$C1_k$	Completion time at the first stage of job in position k.
C_k	Job completion time, i.e., the time at which job in position k leaves the second stage.

Table 2.3: Notation of the $DPm \rightarrow 1||\sum Cj$ problem.

$$\text{Min} \quad \sum C_j$$

subject to:

$$\sum_{k=1}^{n} X_{jk} = 1, \quad 1 \leq j \leq n \tag{2.4}$$

$$\sum_{j=1}^{n} X_{jk} = 1, \quad 1 \leq k \leq n \tag{2.5}$$

$$C1_{i,1} = \sum_{j=1}^{n} X_{j,1} p_{ij}, \quad 1 \leq i \leq m_1 \tag{2.6}$$

$$C1_{ik} = C1_{i,k-1} + \sum_{j=1}^{n} X_{jk} p_{ij}, \quad 1 \leq i \leq m_1, 2 \leq k \leq n \tag{2.7}$$

$$C1_k \geq C1_{ik}, \quad 1 \leq i \leq m_1, 1 \leq k \leq n \tag{2.8}$$

$$C_1 = C1_1 + \sum_{j=1}^{n} X_{j,1} at_j \tag{2.9}$$

$$C_k \geq C_{k-1} + \sum_{j=1}^{n} X_{jk} at_j, \quad 2 \leq k \leq n \tag{2.10}$$

$$C_k \geq C1_k + \sum_{j=1}^{n} X_{jk} at_j, \quad 2 \leq k \leq n \tag{2.11}$$

$$X_{jk} \in \{0, 1\}, \quad 1 \leq j \leq n, 1 \leq k \leq n \tag{2.12}$$

$$C1_{ij}, C1_j, C_j \geq 0, \quad 1 \leq i \leq m_1, 1 \leq j \leq n \tag{2.13}$$

Constraints (2.4) guarantee that each job is only placed in one position at the first stage and constraints (2.5) ensure that each position at the first stage is occupied by only one job. Constraints 2.6 and 2.7 calculate the completion time of each component of the job in position k at each machine i in the first stage, while constraints (2.8) compute the completion time of each job in position j at the first stage. Constraints (2.9) are used to obtain the completion time of the job scheduled in the first position in the second stage. The sets of constraints (2.10) and (2.11) calculate the completion time of each job in position k in the second stage. The set of constraints (2.12) guarantees that X_{jk} is a binary variable and, finally, constraints (2.13) assures that all variables are non-negatives.

The mathematical model of the $DPm \rightarrow Pm||\sum C_j$ (MA) variant is presented below, and the notation is shown in Tables 2.3 and 2.4. The objective function and constraints can be written as follows:

$$\text{Min} \quad \sum C_j$$

subject to:

$$\sum_{k=1}^{n} X_{jk} = 1, \quad 1 \leq j \leq n \tag{2.14}$$

$$\sum_{j=1}^{n} X_{jk} = 1, \quad 1 \leq k \leq n \tag{2.15}$$

$$\sum_{q=1}^{m_2} Z_{kq} = 1, \quad 1 \leq k \leq n \tag{2.16}$$

Indexes	Definition
h	Index for jobs: $h \in \{1, \ldots, n\}$.
i	Index for machines in the first stage: $i \in \{1, \ldots, m_1\}$.
q	Index for machines in the second stage: $q \in \{1, \ldots, m_2\}$.
Parameters	**Definition**
p_{ij}	Processing time of job j in machine $i \in \{1, \ldots, m_1\}$.
at_j	Processing time of job j in the second stage.
Variables	**Definition**
Z_{qk}	1 if job in position k in the first stage is assigned to machine q in the second stage. It is equals zero if the job is not assigned to assembly machine q.
$C2_{qk}$	Completion time of job in position k in the first stage on machine q at the second stage. It is equals zero if the job is not assigned to assembly machine q.
CT_{qk}	Completion time of the last job before job in position k, which has been assigned on assembly machine q. Note that the last job assigned to machine q is not necessarily the job in position $k-1$.

Table 2.4: Notation of the $DPm_1 \rightarrow Pm_2 || \sum Cj$ problem. See also Table 2.3.

$$C1_{i,1} \geq \sum_{j=1}^{n} X_{j,1} \cdot p_{ij}, \quad 1 \leq i \leq m_1 \tag{2.17}$$

$$C1_{ik} \geq C1_{i,k-1} + \sum_{j=1}^{n} X_{jk} \cdot p_{ij}, \quad 1 \leq i \leq m_1, 2 \leq k \leq n \tag{2.18}$$

$$C1_k \geq C1_{ik}, \quad 1 \leq i \leq m_1, 1 \leq k \leq n \tag{2.19}$$

$$CT_{qk} \geq C2_{q(h-1)}, \quad 1 \leq q \leq m_2, 2 \leq k \leq n, 2 \leq h \leq k \tag{2.20}$$

$$C2_{q,1} \geq Z_{q,1} \cdot (C1_1 + \sum_{j=1}^{n} X_{j,1} \cdot at_j), \quad 1 \leq q \leq m_2 \tag{2.21}$$

$$C2_{qk} \geq Z_{qk} \cdot (C1_k + \sum_{j=1}^{n} X_{j,k} \cdot at_j), \quad 1 \leq q \leq m_2, 2 \leq k \leq n \tag{2.22}$$

$$C2_{qk} \geq Z_{qk} \cdot (CT_{qk} + \sum_{j=1}^{n} X_{j,k} \cdot at_j), \quad 1 \leq q \leq m_2, 2 \leq k \leq n \tag{2.23}$$

$$X_{jk}, Z_{qk} \in \{0,1\}, \quad 1 \leq q \leq m_2, 1 \leq j \leq n, 1 \leq k \leq n \tag{2.24}$$

$$C1_{ij}, C1_j, C_{qk}, C2_{qk} \geq 0, \quad 1 \leq i \leq m_1, 1 \leq q \leq m_2, 1 \leq j \leq n \tag{2.25}$$

Constraints (2.14) guarantee that each job is only placed in one position at the first stage and constraints (2.15) ensure that each position at the first stage is occupied by only one job. Similarly, constraints (2.16) ensure that every job is assigned to exactly one machine in the second stage. Constraints (2.17) and (2.18) calculate the completion time of each job in position k at each machine i in the first stage and constraints (2.19) compute the completion time of each job in position k at the first stage. Constraints (2.20) assure that each job in position k can be assembled on machine q only when the previous job processed in that machine has finished. The sets of non-linear constraints (2.21), (2.22) and (2.23) compute the final completion time of the job in position k assembled on machine q in the second stage. Constraints (2.24) state that X_{jk} and Z_{jq} are binary variables and, finally, constraints (2.25) assure that all variables are non-negatives values.

To transform this problem into a MILP, constraint (2.21) should be replaced with constraints 2.13a-2.13h, and (2.22) and (2.23) with constraints 2.14a-2.14j:

$$C2_{q,1} = dd_{q,1} + ee_{q,1} \quad (2.13a)$$

$$dd_{q,1} \geq M \cdot Z_{q,1} + C1_1 - M \quad (2.13b)$$

$$dd_{q,1} \leq C1_1 \quad (2.13c)$$

$$dd_{q,1} \leq M \cdot Z_{q,1} \quad (2.13d)$$

$$ee_{q,1} \geq M \cdot Z_{q,1} + \sum_{j=1}^{n} X_{1,k} \cdot at_1 - M \quad (2.13e)$$

$$ee_{q,1} \leq \sum_{j=1}^{n} X_{1,k} \cdot at_1 \quad (2.13f)$$

$$ee_{q,1} \leq M \cdot Z_{q,1} \quad (2.13g)$$

$$dd_{q,1}, ee_{q,1} \geq 0 \quad (2.13h)$$

$$C2_{qk} = dd_{qk} + ee_{qk} \quad (2.14a)$$

$$dd_{qk} \geq M \cdot Z_{qk} + C1_k - M \quad (2.14b)$$

$$dd_{qk} \geq M \cdot Z_{qk} + CT_{qk} - M \quad (2.14c)$$

$$dd_{qk} \leq C1_k \quad (2.14d)$$

$$dd_{qk} \leq CT_{qk} \quad (2.14e)$$

$$dd_{qk} \leq M \cdot Z_{qk} \quad (2.14f)$$

$$ee_{qk} \geq M \cdot Z_{qk} + \sum_{j=1}^{n} X_{jk} \cdot at_j - M \quad (2.14g)$$

$$ee_{qk} \leq \sum_{j=1}^{n} X_{jk} \cdot at_j \quad (2.14h)$$

$$ee_{qk} \leq M \cdot Z_{qk} \quad (2.14i)$$

$$dd_{qk}, ee_{qk} \geq 0 \quad (2.14j)$$

Note that in above, M represents a huge positive number and dd_{qk} and ee_{qk} can be defined as auxiliar variables.

2.4 Mathematical model of the 2-ASP-pm with makespan objective

The mathematical model of the $DPm \rightarrow 1|nr - pm| \sum C_{\max}$ problem is presented below and the notation is shown in Table 2.5. The objective function and constraints can be written as follows:

Indexes	Definition
j	Index for jobs: $j \in \{1, \ldots, n\}$.
k	Index for the position in the sequence: $k \in \{1, \ldots, n\}$.
i	Index for machines: $i \in \{1, \ldots, m+1\}$.
z	Index for bins: $z \in \{1, \ldots, Z\}$.
Parameters	**Definition**
T	Availability period or length of the bins.
t	Unavailability period.
p_{ij}	Processing time of job j in machine $i \in \{1, \ldots, m\}$.
at_j	Processing time of job j in machine $m+1$.
M	Big number. It can be computed as $M = \sum_{j=1}^{n}(\sum_{i=1}^{m} p_{ij}) + at_j$.
L	Maximal number of bins.
Variables	**Definition**
X_{jk}	1 if job j is placed in position k of the sequence at the first stage; 0, otherwise.
β_{ikz}	1 if job in position k is assigned to bin z in machine i; 0, otherwise.
$C1_{ik}$	Completion time in machine i of the first stage of job in position k.
$C1_k$	Completion time at the first stage of job in position k.
C_k	Job completion time, i.e., the time at which job in position k leaves the second stage.
$C_{\max}$	Completion time of the last job in the second stage or maximum completion time.

Table 2.5: Notation of the $DPm \rightarrow 1|nr - pm|C_{\max}$ problem.

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

subject to:

$$\sum_{k=1}^{n} X_{jk} = 1, \quad 1 \leq j \leq n \tag{2.15}$$

$$\sum_{j=1}^{n} X_{jk} = 1, \quad 1 \leq k \leq n \tag{2.16}$$

$$C1_{i,1} \geq \sum_{j=1}^{n} X_{j,1} \cdot p_{ij}, \quad 1 \leq i \leq m \tag{2.17}$$

$$C1_{ik} \geq C1_{i,k-1} + \sum_{j=1}^{n} X_{jk} \cdot p_{ij}, \quad 1 \leq i \leq m, 2 \leq k \leq n \tag{2.18}$$

$$C1_k \geq C1_{ik}, \quad 1 \leq i \leq m, 1 \leq k \leq n \tag{2.19}$$

$$C_1 \geq C1_1 + \sum_{j=1}^{n} X_{j,1} \cdot at_j \tag{2.20}$$

$$C_k \geq C_{k-1} + \sum_{j=1}^{n} X_{jk} \cdot at_j, \quad 2 \leq k \leq n \tag{2.21}$$

$$C_k \geq C1_k + \sum_{j=1}^{n} X_{jk} \cdot at_j, \quad 2 \leq k \leq n \tag{2.22}$$

$$C1_{ik} - zT \leq M(1 - \beta_{ikz}), \quad 1 \leq i \leq m, 1 \leq k \leq n, 1 \leq z \leq Z \tag{2.23}$$

$$C1_{ik} - \sum_{j=1}^{n} X_{jk} \cdot p_{ij} + M(1 - \beta_{ikz}) \geq T(z-1), \quad 1 \leq i \leq m, 1 \leq k \leq n, 1 \leq z \leq Z \tag{2.24}$$

$$C_k - zT \leq M(1 - \beta_{m+1,k,z}), \quad 1 \leq k \leq n, 1 \leq z \leq Z \tag{2.25}$$

$$C_k - \sum_{j=1}^{n} X_{jk} \cdot at_j + M(1 - \beta_{m+1,k,z}) \geq T(z-1), \quad 1 \leq k \leq n, 1 \leq z \leq Z \tag{2.26}$$

$$\sum_{z=1}^{Z} \beta_{ikz} = 1, \quad 1 \leq i \leq m, 1 \leq k \leq n \tag{2.27}$$

$$C_{\max} \geq C_j, \quad 1 \leq j \leq n \tag{2.28}$$

$$X_{jk} \in \{0, 1\}, \quad 1 \leq j \leq n, 1 \leq k \leq n \tag{2.29}$$

$$C1_{ij}, C1_j, C_j \geq 0, \quad 1 \leq i \leq m, 1 \leq j \leq n \tag{2.30}$$

Constraints (2.15) guarantee that each job is only placed in one position at the first stage and constraints (2.16) ensure that each position at the first stage is occupied by only one job. Constraints 2.17 and 2.18 calculate the completion time of each component of the job in position k at each machine i in the first stage, while constraint

(2.19) compute the completion time of each job in position j at the first stage. Constraints (2.20) are used to obtain the completion time of the job scheduled in the first position in the second stage. The set of constraints (2.21) and (2.22) calculates the completion time of each job in position k in the second stage. The set of constraints (2.23) and (2.24) controls that each operation starting in a bin also finishes within this bin in the first stage, while constraints (2.25) and (2.26) control the operation in the second stage. Constraints (2.27) ensure that each operation is scheduled in only one bin and constraints (2.28) state that the makespan is the maximum completion time. The set of constraints (2.29) guarantees that X_{jk} is binary variable and, finally, constraints (2.30) assures that all variables are non negatives.

Chapter 3

Conditions and characteristics for the evaluation of algorithms

3.1 Introduction

In this chapter, we introduce the characteristics and conditions of the different computational evaluations performed throughout this thesis. As explained in the general research objectives, several resolution methods are proposed to solve the problems under study throughout the thesis. Therefore, in the next chapters, common information is needed, such as the indicators used to measure the performance of the methods and the set of instances used to test the methods. For this reason, it has been decided to include this information in this chapter, which will make the following chapters easier to read. More specifically, in Section 3.2 we explain the performance indicators used to compare the different algorithms. Then, in Section 3.3, the sets of instances used to compare the different algorithms are presented. Finally, in Section 3.4, the experimental conditions that must be met are presented.

3.2 Performance indicators

When different approximate algorithms are evaluated, a trade-off between two aspects is needed: the quality of the solutions and the computational time required by the algorithm to get them. To select an algorithm from the set of algorithms available for the problem, both aspects should be weighted, and different decision intervals may

be required and different qualities of the solution can be accepted. However, in most cases, the decision maker has no knowledge of the precise trade-off. Therefore, in this thesis, we follow the idea of representing the algorithms along the above mentioned criteria (quality of solution and computational time) and obtain a set of efficient algorithms.

In this section, we present the performance indicators employed to represent the algorithms and compare the different results obtained from the different computational evaluations carried out in this thesis. Firstly, we use the Average Relative Percentage Deviation ($ARPD$) to measure the quality of the solutions of the different methods according to Equation 3.1:

$$ARPD_h = \frac{\sum_{\forall s} RPD_{hs}}{S},\ \forall\ s = 1, \cdots, S \tag{3.1}$$

where S is the total number of instances and RPD_{hs} computed as:

$$RPD_{hs} = \frac{OF_{hs} - MIN}{MIN} \cdot 100 \tag{3.2}$$

with OF_{hs} the value of the objective function (makespan or total completion time) obtained by method h ($h = 1, \cdots, H$) in instance s ($s = 1, \ldots, S$) and MIN the minimum known solution for each instance s. The computational effort is measured by means of the Average CPU ($ACPU$) time:

$$ACPU_h = \frac{\sum_{\forall s} T_{hs}}{S} \tag{3.3}$$

where T_{hs} is the time (in seconds) required by method h to obtain a solution for instance s. Furthermore, since the $ACPU$ indicator presents some problems if it is used to compare heuristics with different stopping criteria (Fernandez-Viagas and Framinan, 2015a), the Relative Percentage computation Time (labelled RPT) is computed, as indicated in Equation (3.4), in order to evaluate methods with different number of steps in their procedures.

$$RPT_{hs} = \frac{T_{hs}}{ACPU_s} \tag{3.4}$$

where $ACPU_s$ is the average CPU time obtained for instance s and is computed according to Equation 3.5:

$$ACPU_s = \frac{\sum_{\forall h} T_{hs}}{H} \tag{3.5}$$

Finally, the Average RPT ($ARPT$) can be defined as follows:

$$ARPT_h = \sum_{s=1}^{S} \frac{RPT_{hs}}{S} \tag{3.6}$$

3.3 Benchmarks

In this section, and for the reader's convenience, we present the characteristics of the sets of instances employed in this thesis. First, we present a set of instances adapted from the literature and used in Section 6.5.2 and 6.5.3. Then, after reviewing the literature in Section 4.2, we identify the need for a common benchmark. Therefore, in this section, the new benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$ are described, and, in Chapter 5, the generation process carried out to obtain them is detailed. These benchmarks are used in Part III to compare the new proposals with the existing algorithms and, thus, determine the state-of-the-art in a common framework.

As reviewed in Section 4.2, in the related literature, different sets of instances (see those from Al-Anzi and Allahverdi, 2006b, 2007; Allahverdi and Al-Anzi, 2009; Al-Anzi and Allahverdi, 2012, among others) are used for the problem under study and for related problems. In these testbeds, the processing times are generated in the same way, but each testbed has a different number of jobs and machines in the first stage. From the testbeds found in the literature, the following benchmarks are generated in order to evaluate and compare the different solution procedures:

- Benchmark $\mathscr{B}_0$: This testbed is generated following the procedure by Al-Anzi and Allahverdi (2012). We adapt it in order to consider the parameter m_2. Thus, the proposed testbed consists of 30 instances generated for each combination of n, m_1 and m_2. More specifically, the problem data are generated for $n \in \{30, 40, 50, 60, 70\}$, $m_1 \in \{2, 4, 6, 8\}$, and $m_2 \in \{2, 4, 6, 8\}$. The processing times of the jobs in the machines in the first stage are drawn from a $U \in [1, 100]$ distribution, while in the second stage the processing times are drawn from a $U \in [1, m_2 \cdot 100]$ distribution to balance both stages and have different scenarios regarding the relative processing times in each stage. In total, 2,400 instances are generated.

- Benchmark $\mathscr{B}_1$: This benchmark is generated, after an exhaustive study, in Chapter 5 for the $DPm \rightarrow 1||\sum C_j$ problem. It consists of 10 instances for

each combination of $n \in \{50, 100, 150, 200, 250, 300\}$ and $m_1 \in \{2, 4, 6, 8\}$. The procedure followed to generate the processing times is explained in the referred chapter. In total, this benchmark has 240 instances.

- Benchmark $\mathscr{B}_2$: This benchmark is generated, after an exhaustive study, in Chapter 5 for the $DPm \rightarrow Pm||\sum C_j$ problem. It consists of 10 instances for each combination of $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$, and $m_2 \in \{2, 4, 6, 8\}$. The procedure followed to generate the processing times is explained in the referred chapter. In total, this benchmark has 960 instances.

- Benchmark $\mathscr{B}_3$: This benchmark is generated to the mathematical model presented in Section 2.4. It consists of 10 instances for each combination of $n \in \{10, 20, 30, 40\}$, $m_1 \in \{2, 4, 6, 8\}$, and $T \in \{200, 300, 400, 500\}$. In total, this benchmark has 640 instances.

- Benchmark $\mathscr{B}_4$: This benchmark is generated following the characteristics of benchmark $\mathscr{B}_1$ for the $DPm \rightarrow 1|nr - pm|C_{\max}$ problem. It consists of 10 instances for each combination of $n \in \{50, 100, 150, 200, 250, 300\}$, $m_1 \in \{2, 4, 6, 8\}$, and $T \in \{200, 300, 400, 500\}$. In total, this benchmark has 960 instances.

Furthermore, the sets of instances used to evaluate the algorithms have to be different from those used to calibrate the parameters of the proposed algorithms to avoid an over-calibration. The calibration testbeds are as follows:

- Calibration benchmark $\mathscr{B}_{C0}$: 10 instances have been generated for $n \in \{30, 40, 50, 60, 70\}$, $m_1 \in \{2, 4, 6, 8\}$, and $m_2 \in \{2, 4, 6, 8\}$. The processing times in the first stage are generated according to $p_{ij} \in U[1, 100]$, while in the second stage, $at_j \in U[1, m_2 \cdot 100]$.

- Calibration benchmark $\mathscr{B}_{C1}$: 10 instances have been generated for $n \in \{50, 100, 150, 200, 25, 300\}$, $m_1 \in \{2, 4, 6, 8\}$ and $m_2 = 1$. The processing times of the jobs are drawn from a uniform distribution $U[1, 100]$ for the machines of the first stage, and from a $U[1, \alpha 100]$ distribution, with $\alpha = 2$ in the second stage (following the procedure designed in Chapter 5).

- Calibration benchmark $\mathscr{B}_{C3}$: 10 instances have been generated for $n \in \{50, 100, 150, 200, 25, 300\}$, $m_1 \in \{2, 4, 6, 8\}$, $m_2 = 1$ and $T \in \{200, 300, 400, 500\}$. The

processing times of the jobs are drawn from a uniform distribution $U[1, 100]$ for the machines of the first stage, and from a $U[1, 2 \cdot m_2 \cdot 100]$ distribution in the second stage.

3.4 Experimental conditions

In this thesis, several approximate methods are proposed and compared with other algorithms from the literature. To have a fair comparison among all methods, all selected algorithms in this thesis are again fully re-coded in C# and tested under the same conditions, which means:

- Using the same computer, which means the same processor speed, bus speed, memory speed, and size. More specifically, a cluster of 12 Intel Core i7-3770 PC with 3.4 GHz and 16 GB RAM is used.
- Using the same programming language (C# under Visual Studio 2019) and compiler.
- Using the same operating system.
- Using the same libraries and common functions.
- Using the same set of instances in each comparison.

Furthermore, for each instance, five runs are carried out to better fit the computational time of each heuristic. Then, the average values of the indicators (quality of the solutions and computational effort) are computed.

Part II

Analysis

Chapter 4

State of the art

4.1 Introduction

In this chapter, we perform a comprehensive review of the literature for the problems under consideration to fulfill Objective GO1. The structure of this chapter, shown in Figure 4.1, is organised as follows.

In Section 4.2, we review and 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 the related scheduling problems (i.e., with slightly different constraints or objective functions), which can be easily adapted to the problem under study.

Section 4.3 analyses the literature related to the 2-ASP. Despite its applicability in real life, there are few references that address the variant with several identical machines in the second stage ($DPm \rightarrow Pm||\sum C_j$ or MA). In this section, we review the existing solution procedures for the $DPm \rightarrow 1||\sum C_j$ variant (or SA) and for other related scheduling problems, namely the Customer Order scheduling problem ($DPm \rightarrow 0||\sum C_j$).

Finally, in Section 4.4, we review the literature related to the periodic maintenance constraint (2-ASP-pm). Due to the importance of this constraint in real manufacturing, it has been widely studied for different layouts, e.g., single machine scheduling problem; parallel machines scheduling problem; and flowshop scheduling problem, among others. However, to the best of our knowledge, this is the first time that the $DPm \rightarrow 1|nr-pm|C_{\max}$ problem is addressed. Therefore, we review the literature of the related problems, more specifically, $1|nr-pm|C_{\max}$, $F2|nr-pm|C_{\max}$ and

$DPm \rightarrow 1||C_{\max}$.

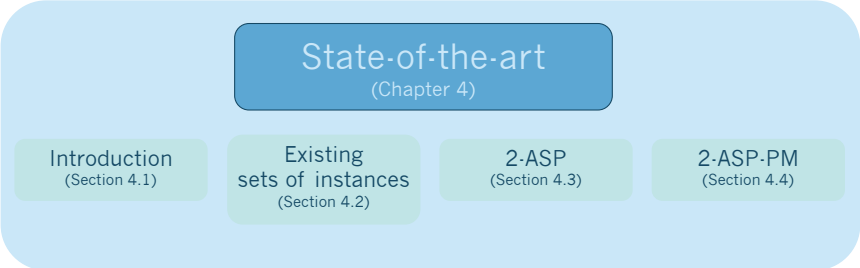

Figure 4.1: Structure of the Chapter 4.

4.2 Existing sets of instances

In this section, we review the literature related to the sets of instances used in the literature to test the different methods for solving the problem considered in this thesis. As introduced before, existing testbeds from related scheduling problems are also incorporated into the analysis, as they could be easily adapted to the problem under study.

In Table 4.1, we summarise 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 columns, 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

Table 4.1: Sets of instances generated to solve related problems.

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

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 scarcely 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 4.2) in order to incorporate what the respective authors consider as dominance between the two stages [1]. 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, the case in which the workload in 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

[1] 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 4.2: References with the generation of the processing times in different classes.

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).

4.3 The 2-ASP

As mentioned in Section 4.1, there are few references that address the assembly scheduling problem with several machines in the second stage ($DP_m \rightarrow Pm||\sum C_j$). These references are the works by Sung and Kim (2008) and Al-Anzi and Allahverdi (2012). Sung and Kim (2008) developed a heuristic, denoted SAK from now on, applying a processing-time-based pairwise exchange mechanism, while in Al-Anzi and Allahverdi (2012), a mathematical model of the problem with two assembly machines was proposed, together with three new metaheuristics. Due to the scarce literature handling the $DP_m \rightarrow Pm||\sum C_j$ problem, we review the existing literature of the $DP_m \rightarrow 1||\sum C_j$ and $DP_m \rightarrow 0||\sum C_j$ problems.

Regarding the $DP_m \rightarrow 1||\sum C_j$ problem, the first reference that addresses it is Tozkapan et al. (2003), where the authors proved that permutation schedules are optimal for the $DP_m \rightarrow 1$ problem and proposed two heuristics, labelled $TCK1$ and $TCK2$ in the following, to find an upper bound for their branch and bound algorithm. Al-Anzi and Allahverdi (2006a) also addressed this problem and derived a number of theoretical properties. They proposed three simple constructive heuristics ($S1$, $S2$, and $S3$) based on the idea of ordering the jobs according to the Shortest Processing Time (SPT) rule, and two additional constructive heuristics, labelled $A1$ and $A2$ in the following. Recently, Framinan and Perez-Gonzalez (2017b) developed a constructive heuristic, denoted FAP in the following, which outperforms the existing constructive heuristics and is based on the problem properties studied by Al-Anzi and Allahverdi (2006a). The last reference where this problem is considered is Lee (2018). Six

lower bounds were proposed and tested in a branch-and-bound algorithm. The author also proposed four greedy-type constructive heuristics, labelled $G1$, $G2$, $G3$, and $G4$. This problem has also been tackled with respect to different objectives: Lee et al. (1993) and Potts et al. (1995) addressed this problem regarding the minimisation of the makespan, while Al-Anzi and Allahverdi (2006b) and Allahverdi and Al-Anzi (2006) considered the minimisation of the maximum lateness; additional constraints such as setup times (Al-Anzi and Allahverdi, 2007), or additional stages for the transportation of components (Koulamas and Kyparisis, 2001 and Shoaardebili and Fattahi, 2015) have also been studied.

For the $DP_m \rightarrow 0||\sum C_j$ problem, Sung and Yoon (1998) proposed two constructive heuristics based on the SPT rule. The first one schedules the order with the smallest total processing time across all m machines, labelled $STPT$ in the following, and the second one selects the order with the smallest maximum amount of processing time on any of the m machines, denoted as $SMPT$. Leung et al. (2005) proposed 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) proposed a constructive heuristic and two specific local search mechanisms for the problem, labelled $SHIFT_k$ and $SHIFT_{k_{OPT}}$.

4.4 The 2-ASP-pm

In this section, we review the literature related to the periodic maintenance constraint. To the best of our knowledge, the $DPm \rightarrow 1|nr - pm|C_{\max}$ problem has never been addressed so far and no resolution methods have been designed to solve it. Therefore, we review the existing literature of the related problems presented in Chapter 2.

Regarding the $1|nr - pm|C_{\max}$ problem, with a single machine, it was firstly addressed by Ji et al. (2007). The authors proposed a dispatching rule, named LPT, which first sorts the jobs in descending order of their processing times and then assigns them according to the First Fit (or FF, see the explanation in Section 7.2) policy. Low et al. (2010a) proposed a Particle Swarm Optimization (PSO) algorithm, using different dispatching rules to generate the initial population. Hsu et al. (2010) addressed a different version of the problem where the machine stops for maintenance after T units of time or after processing A jobs. This problem is equivalent to

$1|nr-pm|C_{\max}$ if $A = n$. Firstly, the authors presented a Binary Integer Programming model and, secondly, they proposed two heuristics. The first heuristic sorts the jobs according to LPT and assigns them using the Best Fit (or BF, see the explanation in Section 7.2) policy. The second heuristic sorts the jobs in the so-called butterfly order (given the jobs in LPT, select first the largest one, then the smallest, the second largest, the second smallest, and so on), and then applies the BF bin packing policy. In Low et al. (2010b), the problem with flexible maintenance was addressed, where the maintenance has to be executed within a given time window. The authors proposed six constructive heuristics obtained by combining a sequencing priority list (random order, SPT order and LPT order) with the FF or BF policies. Yu et al. (2014) designed three constructive heuristics, labelled *LS*, *LPT* and *MLPT*, concluding that *MLPT* is the best among them. Finally, Perez-Gonzalez and Framinan (2018) designed a heuristic, denoted here by *PGF*, using a given bin packing assignment policy as an operator to be applied to permutation sequences, and carried out a computational evaluation with all the previous procedures.

With respect to the $DPm \rightarrow 1||C_{\max}$ problem, Lee et al. (1993) considered the $DP2 \rightarrow 1||C_{\max}$ problem by proposing a branch-and-bound solution scheme and three Johnson-based heuristics, labelled LCL_1, LCL_2 and LCL_3, which reduces the problem to a modified 2-machine flowshop scheduling problem. Sun et al. (2003) considered the same problem and proposed different heuristic algorithms based also on Johnson's rule. Lin et al. (2006) addressed the same problem, assuming that the assembly machine processes batches of specific jobs and that a (non-sequence dependent) setup is required every time a bin is formed. The authors proved that this problem is strongly NP-hard, detected some few polynomially solvable cases, and provided heuristics for the problem. Potts et al. (1995) addressed the problem with m machines in the first stage and proposed a heuristic, which is an extension of the LCL_3. Allahverdi and Al-Anzi (2006) addressed the same problem with setup times and proposed three different approximate algorithms. Komaki and Kayvanfar (2015) studied the problem with release times, developed several heuristics (based on Johnson's rule) and also proposed a meta-heuristic algorithm. Finally, regarding the problem with an intermediate stage for collection and transportation (denoted as $DPm \rightarrow F2||C_{\max}$), Koulamas and Kyparisis (2001) analysed the worst-case ratio bound for several heuristics and proposed a heuristic based on compact vector summation techniques. In Komaki et al. (2017), the authors proposed an Improved

Discrete Cuckoo Optimization Algorithm, a lower bound, and a series of dispatching rules for the same problem.

Regarding the flowshop scheduling problem with maintenance, Perez-Gonzalez et al. (2020) developed specific heuristics with different computational complexity, employed an iterated greedy algorithm, and carried out an extensive computational experiment to establish the efficiency of the proposed heuristics. Recently, Zhao et al. (2020) studied the problem with blocking and minimisation of the makespan.

4.5 Conclusions of the chapter

In this chapter, on the one hand, the literature related to the existing sets of instances of related problems has been analysed, and, on the other hand, the literature of the existing approximate methods for the related problems of the two problems under study, 2-ASP and 2-ASP-pm, has been reviewed. After the first analysis of the existing testbeds carried out in Section 4.2, 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 the majority of the literature 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 parameters (number of machines in both stages) for each number of jobs (see e.g., 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 the 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.

Therefore, we find appropriate to design and propose two comprehensive benchmarks for the 2-stage multi-machine scheduling problem with total completion time criterion, one for each variant considered (see Objective GO2). All the analysis and methodology are presented in Chapter 5.

Regarding the review in Section 4.3, it can be seen that despite the existence of some solution procedures for the 2-ASP, the performance of the adaptation of procedures from related problems has not been tested so far. For these reasons, we find pertinent to address the $DP_m \rightarrow Pm||\sum C_j$ problem and propose some heuristic algorithms in Chapter 6 (see Objectives GO3 and GO4).

From the analysis of the literature related to the 2-ASP-pm (Section 4.4, we can conclude that, while the periodic maintenance constraint has been widely studied for other scheduling problems, it has never been considered in the 2-stage assembly scheduling problem so far. To cover this opportunity, in Chapter 7, we adapt the existing approximate algorithms from related problems and propose new algorithms to solve the 2-ASP-pm (see Objective GO5) with the minimisation of the makespan.

Chapter 5

New benchmark generation

5.1 Introduction

Following the recommendation identified in Section 4.5, in this chapter, we generate a new benchmark for the 2-ASP with total completion time minimisation. More specifically, in Section 5.2, we explain the reasons for justifying the need for a common benchmark to solve this problem. Then, the methodology adopted to select the most suitable instances is explained in the previous chapter (see Section 5.3), and the experiments needed to apply the methodology are included in Section 5.4. Finally, we discuss the conclusions in Section 5.5.

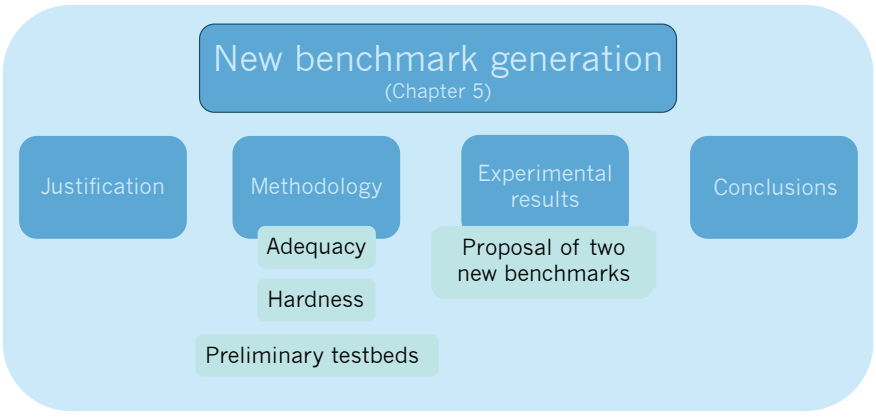

Figure 5.1: Structure of Chapter 5.

5.2 Justification

According to the conclusions obtained after reviewing the literature in Section 4.2, the issues to justify the proposal of a new benchmark for the problem are the following:

- Every time a new approximate method for the problem is proposed, it is tested in 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.
- 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 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 5.3.1 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 chapter is aimed to tackle these issues. More specifically, the contribution is twofold: First, an analysis of the context is performed with a computational experiment to determine the relationship among our variants and the related scheduling problems. Second, using this information, we propose two comprehensive benchmarks for 2-ASP with the total completion time criterion (one for each variant considered, SA and MA).

5.3 Methodology

In Section 4.2, 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 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, leads us to obtain a more discriminant benchmark.

In this section, the procedure generated to design the new benchmarks and satisfy the previous characteristics is described. Figure 5.2 shows the outline of the procedure. First, 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 5.3.1). 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 5.3.2). 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 $\mathscr{B}_1$ and $\mathscr{B}_2$ in Figure 5.2, are explained in Section 5.3.3. Finally, after the methodology has been determined, the experimental analysis is carried out in Section 5.4.

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). 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 4.2. As can be observed, the use of a sequence provides a generic solution for all the considered scheduling problems.

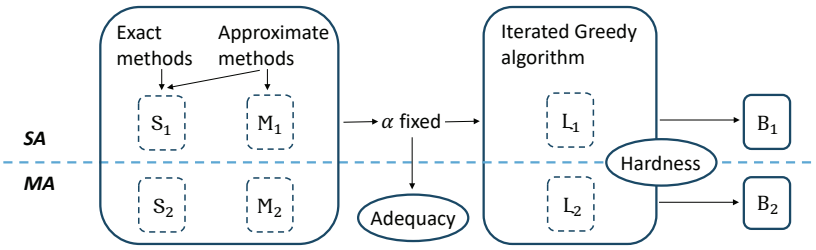

Figure 5.2: Diagram of the methodology designed to propose the new benchmarks.

5.3.1 Adequacy

In order to guarantee the adequacy of the generated benchmarks for each variant, *SA* and *MA*, the balance between stages is analysed to determine the relationship among the problems. As usual in the problem under consideration (see Table 4.1) 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, to control the different scenarios of balance / unbalance mentioned above, the processing times are generated using $at_j \sim U[1, \alpha \cdot m_2 \cdot 100]$, where α is a parameter that represents the existing relationship between the workload of the two stages. Based on preliminary results in which a wider range of α values is 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 for inclusion in the benchmarks, avoiding those that can be efficiently solved by methods designed for *CO*, *SM*, and *PM*.

To accomplish with adequacy, small instances are first solved using exact methods (explained below), developed for *CO*, *SM*, *PM*, *SA*, and *MA*. Second, 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 where 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 the problem $\mathcal{T}$, with $\mathcal{T} \in \{SA, MA, CO, PM, SM\}$. Then, this solution is evaluated for other related problems 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 $\mathscr{T}$, $S_{\mathscr{T}}$, evaluated for the problem $\mathscr{T}'$. 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 they 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 presented in Section 2.3.
- For *CO*: an adaptation of the MILP model presented in Section 2.3, 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_{MA}, proposed in Chapter 6.
- 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.

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 5.1: Processing times of the jobs in both stages.

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 5.1. 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 5.3a). Then, the 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 from the original instance (see Figure 5.3b), 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 5.3c), with the value of the objective function $\sum C_j^{MA}(S_{PM})$=142.

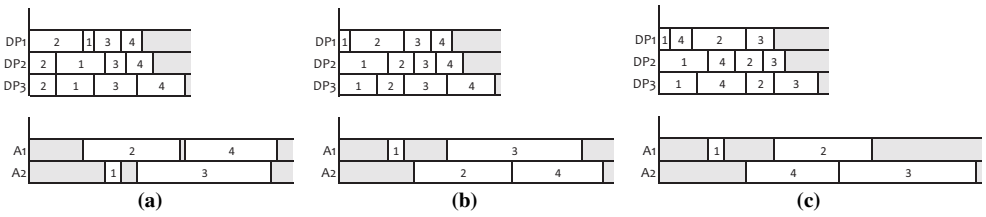

Figure 5.3: Gantt charts of the different optimal sequences evaluated for the *MA* variant. Figure 5.3a Gantt chart of the optimal sequence of the *MA* variant, S_{MA}. Figure 5.3b Gantt chart of the optimal sequence of the *CO* problem, S_{CO}, when it is evaluated for *MA*. Figure 5.3c Gantt chart of the optimal sequence of the *PM* problem, S_{PM}, when it is evaluated for *MA*.

5.3.2 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 selected metaheuristic 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 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 an 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 calculated following Equation (5.1), 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 (5.1) 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 prior to job j given in the order defined by $p_{i[k]}$. On the other hand, the processing time is added in the second stage. Note that since our proposed lower bound for $DPm_1 \rightarrow 1||\sum C_j$ does not consider the waiting time for the job 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. Also, since the total completion time cannot increase with the number of machines in the second stage, this LB is tighter for the MA variant.

$$LB = \sum_{j=1}^{n} \left(max \left\{ \sum_{k=1}^{j} \lceil w_{[k]} \rceil, \max_{i=1,\ldots,m_1} \left\{ \sum_{k=1}^{j} p_{i[k]} \right\} \right\} + at_j \right) \tag{5.1}$$

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

$$RPD_{M_{\mathcal{T}}}(\mathcal{T}') = \frac{\sum C_j^{\mathcal{T}'}(h) - MIN}{MIN} \tag{5.2}$$

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. Instances with the highest values of *RPD* will be selected to be part of the new benchmark.

5.3.3 Preliminary testbeds and selection procedure

In this section, the procedure to select the final sets of instances, denoted as $\mathscr{B}_1$ and $\mathscr{B}_2$, is explained. The selection procedure is carried out using different preliminary testbeds: $\mathscr{S}_1$, $\mathscr{M}_1$ and $\mathscr{L}_1$ for *SA* including small, medium and large size instances, respectively; and $\mathscr{S}_2$, $\mathscr{M}_2$ and $\mathscr{L}_2$ 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 $\mathscr{S}_1$ and $\mathscr{S}_2$, $n \in \{30, 40, 50, 60, 70\}$ for $\mathscr{M}_1$ and $\mathscr{M}_2$, and $n \in \{50, 100, 150, 200, 250, 300\}$ for $\mathscr{L}_1$ and $\mathscr{L}_2$. Although the maximum number of jobs considered in the literature is equal to 200 (see Table 4.1 in Section 4.2), 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 $\mathscr{S}_1$ and $\mathscr{S}_2$, and $m_1 \in \{2, 4, 6, 8\}$ (based on the literature, see Table 4.1 in Section 4.2) in $\mathscr{M}_1$, $\mathscr{M}_2$, $\mathscr{L}_1$, $\mathscr{L}_2$.

- Number of machines in the second stage: $m_2 = 1$ in $\mathscr{S}_1$, $\mathscr{M}_1$ and $\mathscr{L}_1$, $m_2 \in \{2, 3\}$ in $\mathscr{S}_2$, $m_2 \in \{2, 4, 6, 8\}$ in $\mathscr{M}_2$ and $\mathscr{L}_2$. In the literature (see Table 4.1 in Section 4.2) the maximum value of m_2 is 4, so here it has been extended.

- Number of replicates: 30 is established for $\mathscr{S}_1$ and $\mathscr{S}_2$, 10 for $\mathscr{M}_1$ and $\mathscr{M}_2$, and finally 1,000 for $\mathscr{L}_1$ and $\mathscr{L}_2$.
- Values of parameter α: $\alpha \in \{1, 2, 3\}$ for $\mathscr{S}_1$, $\mathscr{S}_2$, $\mathscr{M}_1$ and $\mathscr{M}_2$ (see Section 5.3.1). For $\mathscr{L}_1$ and $\mathscr{L}_2$, α is equal to 2, according to the results obtained in Section 5.4.

Taking into account these parameters, $\mathscr{S}_1$ and $\mathscr{S}_2$ have 540 and 1,080 instances, respectively; $\mathscr{M}_1$ and $\mathscr{M}_2$ 600 and 2,400 instances, respectively; and $\mathscr{L}_1$ and $\mathscr{L}_2$ 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: To obtain the adequacy of the instances, the most suitable values of α should be identified for *SA* and *MA*.

- For the *SA* variant, $\mathscr{S}_1$ is solved using exact methods. Additionally, $\mathscr{S}_1$ and $\mathscr{M}_1$ are solved with the approximate methods. All the methods have been previously described in Section 5.3.1. 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 $\mathscr{S}_1$. 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 $\mathscr{S}_1$ and $\mathscr{M}_1$. 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 $\mathscr{S}_2$ and $\mathscr{M}_2$. 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 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, $\mathscr{L}_1$ and $\mathscr{L}_2$, 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 5.3.2, the instances are solved by the IG and the lower bound for each one is computed. For each instance size, the *Average RPD* obtained is sorted in decreasing order, and the first 10 instances are selected to be part of the new benchmark. The instances selected for each variant form the benchmark $\mathscr{B}_1$ for *SA*, with 240 large size instances (Section 5.4.1), and $\mathscr{B}_2$ for *MA*, with 960 large size instances (Section 5.4.2).

As $\mathscr{B}_1$ and $\mathscr{B}_2$ are selected from the testbeds $\mathscr{L}_1$ and $\mathscr{L}_2$, 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. Furthermore, since the benchmarks consist of a high number of instances, their exhaustiveness is also fulfilled. The instances files of benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$ are published as additional material at the following link:

5.4 Experimental results

In this section, the results obtained by applying the procedure of Section 5.3 are presented. Section 5.4.1 presents the results for *SA*, and Section 5.4.2 for *MA*. In each case, the ANOVA shows the statistically 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 Tukey's HSD test are presented in a simplified way. All detailed results are available in Anova & Tukey Test Experimental Results files at http://grupo.us.es/oindustrial/en/research/results/.

5.4.1 Results for the *SA* variant

This section shows the results provided for the *SA* variant. Regarding adequacy, the preliminary testbed $\mathscr{S}_1$ has been solved using exact methods, and both $\mathscr{S}_1$ and $\mathscr{M}_1$ using approximate methods. Table 5.2 shows the conclusions provided by the Tukey's HSD test for each value of α. On the one hand, 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 $\mathscr{S}_1$. When the approximate methods are applied to $\mathscr{S}_1$ and $\mathscr{M}_1$, 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, hypothesis $H_2 : SA = SM$ is similar when comparing *SA* with *SM*. Therefore, in Table 5.2 it is shown whether 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 $\mathscr{S}_1$. 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 of 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 $\mathscr{S}_1$. Therefore, instances generated with $\alpha = 3$ are not suitable for *SA*. In conclusion, the instances generated with $\alpha = 1$ and $\alpha = 3$ are not suitable (unbalanced) for the *SA* variant, and $\alpha = 2$ is consistently balanced due to the rejection of the hypotheses in all cases.

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

Table 5.2: Conclusions from HSD Tukey tests for *SA*.

Regarding the hardness, testbed $\mathscr{L}_1$ has been solved by IG. Then, the $ARPD_{IG}^{SA}$ with respect to the lower bound has been computed. In Table 5.3, the values of $ARPD$ of the 10 hardest instances of each combination are shown in the rows labelled as

		n						
	m_1	50	100	150	200	250	300	Average
Selected	2	114.54	109.90	112.70	108.23	102.77	101.73	108.31
instances	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	2	62.71	69.24	72.26	73.39	73.38	74.07	70.84
instances	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 5.3: Values of $ARPD$ for the 10 hardest instances and for the total instances of each combination of parameters in testbed $\mathscr{L}_1$.

"Selected instances", while in the rows labelled as "Total instances" the values of $ARPD$ of the total instances of each combination are computed.

5.4.2 Results for the *MA* variant

In this section, the results for the *MA* variant are shown. Similarly to the previous section, the adequacy is analysed. The preliminary testbed $\mathscr{S}_2$ is solved using exact and approximate methods and $\mathscr{M}_2$ using approximate methods. Table 5.4 has the same structure as Table 5.2. 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 $\mathscr{S}_1$. Although the hypothesis is rejected in the rest 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 of 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 $\mathscr{S}_2$. Therefore, instances generated with $\alpha = 3$ are not suitable for *MA*. In the same way as in the previous case, instances generated with $\alpha = 1$ and $\alpha = 3$ are unbalanced for the *MA* variant, where $\alpha = 2$ is again the suitable value.

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

Table 5.4: Conclusions from HSD Tukey tests for *MA*

Regarding the hardness of the instances, the testbed $\mathscr{L}_2$ has been solved by IG and the same procedure as in the previous section is followed. Table 5.5 has a structure similar to Table 5.3, 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 5.3, the hardest instances of $\mathscr{L}_1$ and $\mathscr{L}_2$ form the two new proposed benchmarks, whose parameters can be summarised as follows:

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

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 IG 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

			n						
	m_1	m_2	50	100	150	200	250	300	Average
Selected	2	2	98.15	100.64	105.70	102.85	95.26	96.52	99.85
instances		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	2	2	53.76	63.33	67.67	69.58	70.54	71.64	66.09
instances		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 5.5: Values of $ARPD$ for the 10 hardest instances and for the total instances of each combination of parameters in testbed $\mathscr{L}_2$.

amenable for statistical analysis, since the levels of the parameters are equidistant and all the levels have been combined to generate the instances.

5.5 Conclusions of the chapter

In this chapter, we consider the 2-stage assembly scheduling problem with several dedicated parallel machines in the first stage and one assembly machine in the second stage (SA) and also with several assembly machines in the second stage (MA). Due to the absence of a commonly accepted set of instances in the literature ensuring that the instances are representative of the problem under study, there is a need of hard instances specifically designed for variants *SA* and *MA*, which has been covered in this chapter. Therefore, two new benchmarks of instances are designed (see GO2) 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 have been generated using a parameter α, depending on the relationship between stages. 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 values 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 for each instance has been computed and the IG algorithm has been applied to solve the testbeds. Then, instances whose solution founded by the IG is further from the previous lower bound are selected to form two new benchmarks of 240 instances for *SA*, and 960 instances for *MA*. With this methodology, the characteristics of adequacy and empirical hardness are ensured. The exhaustiveness and amenability are also fulfilled by considering a large number of instances, equidistant levels, and by combining all the levels of parameters to obtain the instances.

Part III

Solution procedures

Chapter 6

Heuristic algorithms for the 2-ASP

6.1 Introduction

In this chapter, we address the 2-ASP with the objective of minimising the total completion time. More specifically, in Section 6.2, we propose a fast constructive heuristic that applies to our problem some dominance properties by Framinan and Perez-Gonzalez (2017b) derived for the case where there is only one assembly machine. In light of the excellent results of this heuristic in negligible computation times, in Section 6.3, it is embedded into a beam-search-based constructive heuristic. This type of heuristic has recently been applied by Fernandez-Viagas and Framinan (2017); Fernandez-Viagas et al. (2016) and Fernandez-Viagas et al. (2018) for related flowshop scheduling problems. In addition, we introduce the idea of keeping only the most promising nodes in each iteration to boost its performance, an idea successfully employed for different scheduling problems (see, e.g., Della Croce and T'kindt, 2003, Valente and Alves, 2008, and Valente, 2010). Then, in Section 6.4, the state-of-the-art algorithms are adapted and implemented, and in Section 6.5, an extensive computational evaluation is performed to show that the proposals are found to be more efficient than the existing solution procedures for the problem. Finally, the conclusions are discussed in Section 6.6.

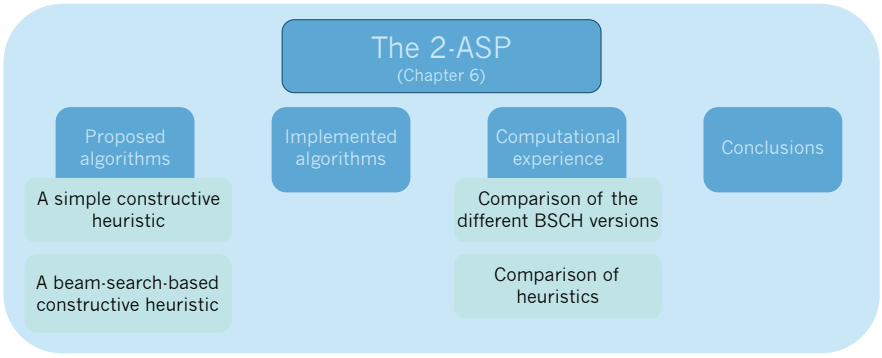

Figure 6.1: Structure of Chapter 6.

6.2 A simple constructive heuristic

The first proposal for solving the 2-ASP is a simple constructive heuristic. The idea of this heuristic, labelled CH_{MA} in the following, consists of iteratively constructing a sequence by selecting one job among the unscheduled jobs and adding it at the end of the partial sequence, an idea that has been addressed for different scheduling problems by Framinan and Perez-Gonzalez (2017b,a) and Fernandez-Viagas and Framinan (2017) with excellent results. A key issue for the performance of this type of heuristics is the development of a problem-specific indicator ψ that adequately represents the suitability of an unscheduled job to be appended since, once a job is added to the sequence, its position cannot be modified in subsequent iterations.

Thence, the proposed algorithm starts with a set $\mathscr{U}$ containing all (unscheduled) jobs and an empty schedule $\mathscr{S}$. For each iteration $k \in \{1, \cdots, n\}$, each unscheduled job $\omega_j \in \mathscr{U}$ $(j = 1, \ldots, n-k+1)$ is analysed as a candidate to be added to the last position in S, and its suitability is measured by computing the indicator ψ_j. Then, the job in $\mathscr{U}$ with the lowest value of ψ_j is selected.

In our problem, two main aspects are considered to assess the suitability of appending a candidate job ω_j at the end of the partial sequence, i.e.,:

1. The case in which its inclusion implies that the first available machine in the second stage has to wait for processing the candidate job. Therefore, the idle time induced in the assembly stage by the insertion of job ω_j is computed. Let us IT_{ω_j} denote the idle time induced by scheduling job ω_j at the end of the sequence which is computed as follows:

$$IT_{\omega_j} = \max\left\{C1_{\omega_j} - (C_{\omega_j} - at_{\omega_j}), 0\right\} \tag{6.1}$$

Note that $C_{\omega_j} - at_{\omega_j}$ computes the workload of the first available machine in the second stage before scheduling job ω_j (i.e., the machine in the second stage where the candidate job will be processed). If the idle time induced by scheduling a job is greater than 0, then the completion times of the components in the first stage dominate the completion time of this job when it is evaluated as candidate to be scheduled. Otherwise, the completion time of the candidate job is largely influenced by the second stage. As we are minimising the total completion time, it is clear that the lower the idle time caused by a job, the more suitable it is to be scheduled.

2. The contribution of the candidate job to the total completion time. As can be seen, the idle time takes into account the suitability of the job until its processing in the second stage starts. In addition, its contribution to the total completion time would depend on the processing time in the second stage, i.e., at_{ω_j}, which roughly measures the influence of the second stage. We weight this influence according to the number of assembly machines (m_2) since, with a higher number of machines in the second stage, the next jobs have a lower probability of not being affected by the second stage. Furthermore, to take into account the fact that, the higher the number of components in the first stage, the higher is its influence on the completion time, CT_{ω_j} the expected contribution of job ω_j to the total completion time is measured as follows:

$$CT_{\omega_j} = \frac{at_{\omega_j}}{m_1 \cdot m_2} \tag{6.2}$$

By taking into account these two aspects, we will ensure that the jobs to be first sequenced are those with lower values of idle time and assembly time. Therefore, the indicator ψ_{ω_j}, which estimates the suitability of appending a candidate job ω_j at the end of $\mathscr{S}$, is computed as follows:

$$\psi_{\omega_j} = a \cdot IT_{\omega_j} + CT_{\omega_j} \tag{6.3}$$

where a is a parameter to weight the influence of the two terms and that would be determined via calibration of the algorithm.

Note that the complexity of this heuristic is $O(n \cdot (n-k) \cdot m_1) \sim O(n^2 \cdot m_1)$, since the main loop in the algorithm performs n iterations. In each iteration, $n-k$ jobs are evaluated, each evaluation consisting on obtaining the maximum processing time in the first stage, so each machine m_1 is evaluated. The pseudo-code of the proposed heuristic is shown in Algorithm 1.

6.3 A beam-search-based constructive heuristic

The heuristic proposed in Section 6.3 provides excellent results with negligible CPU times (see Section 6.5), proving that the indicator ψ_j properly captures the suitability of a job to be appended. Therefore, we embed the components of this indicator into a new beam-search-based constructive heuristic for the problem, labelled $BSCH_{MA}$. This type of heuristic has been considered to solve different scheduling problems, such as in Sotskov et al. (1996) (where some constructive heuristics based on insertion techniques are combined with beam search for the permutation flowshop scheduling problem), Erenay et al. (2010) (for the single machine bicriteria scheduling problem), and Fernandez-Viagas and Framinan (2017) (also for the permutation flowshop scheduling problem). In this type of heuristics, a number of candidate nodes, denoted by a parameter x, are maintained in each iteration. In iteration k, each node l ($l \in \{1, \cdots, x\}$) is formed by a set of k scheduled jobs, denoted as partial sequence $\mathscr{S}_k^l$, and a set of unscheduled jobs, $\mathscr{U}_k^l$ with $n-k+1$ jobs. Then, all unscheduled jobs in $\mathscr{U}_k^l$ are inserted in position $k+1$ of $\mathscr{S}_k^l$, thus obtaining $x \cdot (n-k+1)$ candidate nodes. Out of these nodes, the x most suitable ones are selected as candidates for the next iteration. To deal with that, the ideas behind the indicator ψ_j could be used in this heuristic. However, an additional complication arises because candidates from different nodes may have to be compared. More specifically, the heuristic may have to deal with one of the following situations:

- If the candidate jobs have been obtained by appending different jobs in $\mathscr{U}_k^l$ to the same node l, their partial sequences $\mathscr{S}_k^l$ will be exactly alike with the exception of the last job appended. So, the comparison can be done in reference to the completion time or the idle time caused by the added job.

- If the candidate jobs have been obtained from different nodes, the unscheduled jobs and the scheduled jobs are different for each candidate node. In this case,

Algorithm 1 Pseudo-code of the proposed heuristic CH_{MA}

procedure CH_{MA}
  All jobs are initially unscheduled
  $\Pi := \emptyset$;
  Completion times on stages 1 and 2:
  $C1_i := 0 \quad i = 1, \ldots, m_1$
  $C2_i := 0 \quad i = 1, \ldots, m_2$
  $i^* := \arg\min_{1 \leq i \leq m_2} C2_i$;
  Obtain a sequence $\mathscr{U} := \{\omega_1, \ldots, \omega_n\}$ by applying algorithm S2;
  for $j = 1$ to n **do**
    for each $\omega_j \in \mathscr{U}$ **do**
      Compute the completion times in the first stage after selecting ω_j as candidate:
      $C_1(\omega_j) := \max_{1 \leq i \leq m_1} \{C1_i + p_{i\omega_j}\}$
      Compute the idle time induced if job ω_j is inserted at the end of the partial sequence:
      $IT_{\omega_j} = \max\left\{C_1(\omega_j) - C2^*_{\rangle}, 0\right\}$
      Compute the expected contribution to the total completion time induced when job ω_j is inserted at the end of the partial sequence:
      $CT_{\omega_j} = \frac{at_{\omega_j}}{m_1 \cdot m_2}$
      Compute ψ_{ω_j}:
      $\psi_{\omega_j} := a \cdot IT_{\omega_j} + CT_{\omega_j}$
    end for
    $r := \arg\min_{1 \leq k \leq n-j+1} \psi_k$;
    Append ω_r at the end of Π, i.e., $\Pi := (\pi_1, \ldots, \pi_{j-1}, \omega_r)$;
    Extract ω_r from $\mathscr{U}$, i.e., $\mathscr{U} := \{\omega_1, \ldots, \omega_{r-1}, \omega_{r+1}, \ldots, \omega_{n-j+1}\}$;
    Update values of the constructive sequence:
    $C1_i := C1_i + p_{i\omega_r}$
    $C2_i := \max\left\{C2_{i^*}, \max_{1 \leq i \leq m_1} C1_i\right\} + at_{\omega_r}$
  end forreturn $C2_i$
end procedure

the comparison should take into account that the previous scheduled jobs at each candidate node are different, so this fact has to be considered when developing the indicator for suitability.

To explain the design of the heuristic in detail, we first denote by $\int_{jk}^{l}$ the jth scheduled job of node l in iteration k and by $\sqcap_{jk}^{l}$ the jth unscheduled job of selected node l in iteration k. As shown in Algorithm 2, the heuristic consists of the following steps:

Step 1: Generate the initial x nodes: All jobs are initially sorted according to Algorithm S2 by Al-Anzi and Allahverdi (2006a), as it is done in CH_{MA}. The first x nodes are obtained by assigning the job in position l of the sorted list to the first position of the partial sequence $\int_{11}^{l}$ of the selected node l. The list of unscheduled jobs of this selected node l is formed by the rest of the jobs.

Step 2: Generate candidate nodes: At iteration k, $n-k+1$ candidate jobs are obtained by appending each job in $\mathscr{U}_k^l$ at the end of the partial sequence of each selected node $l \in \{1, \cdots, x\}$.

Step 3: Evaluate candidate nodes: In this step, two aspects are considered: first, the influence from the selected node and, second, the influence from the inserted job. The former is computed as the forecast index, F_{kl}, which is explained in Step 6.3, and the latter is due to the insertion of the new job, $\sqcap_{jk}^{l}$, at the end of the partial sequence, which is measured by CT_{jkl}, analogously as Equation (6.2), and by IT_{jkl}, which denotes the idle time incurred when inserting job $\sqcap_{jk}^{l}$ in the selected node and is computed analogously as Equation (6.1). Note that these two last components are taken from the heuristic in Section 6.2, while F_{kl} is a component specifically designed to allow the comparison of candidates from different nodes.

Therefore, at each iteration k, the following indicator is used to compute the suitability of appending an unscheduled job $\sqcap_{jk}^{l}$ in a selected node l:

$$B_{jkl} := F_{kl} + a' \cdot IT_{jkl} + CT_{jkl} \tag{6.4}$$

In Equation (6.4) the parameter a' has been considered in order to balance the

idle time and the completion time of the new inserted job and its calibration is addressed in Section 6.5.1.

Step 4: Select the best x candidate nodes: The x candidate nodes with the lowest values of B are selected and these nodes will form the nodes of the next iteration, i.e., in iteration k all the combinations of j and l are tested and those achieving the lowest values of B_{jkl}, as defined in Equation (6.4), are selected. The rest of candidate nodes are discarded and the best candidate nodes are defined as the selected nodes for the next iteration. At each iteration k, the combination of l and j of the l'th best B_{jkl} ($l^{'} \in \{1, \ldots, x\}$) are denoted by $branch[l']$ and $job[l']$, respectively.

Step 5: Update forecast index: The forecast index F is defined in order to compare candidate nodes obtained from different nodes and, therefore, composed by different un- and scheduled jobs. F represents the completion time of the last scheduled job at each candidate node and it is computed by Equation (6.5):

$$F_{k,l'} = F_{k-1,branch[l']} + b \cdot \psi_{job[l'],k,branch[l']} \tag{6.5}$$

where parameter b is designed to balance the influence of the last scheduled job in the completion time and $\psi_{job[l'],k,branch[l']}$ is defined by Equation 6.6 (similar as in Equation (6.3)) and computed when job $\sqcap^{l}_{jk}$ is appended. The calibration of b is discussed in Section 6.5.1. The pseudocode of the algorithm is shown in Algorithm 2.

$$\psi_{job[l'],k,branch[l']} = a' \cdot IT_j + CT_j \tag{6.6}$$

Apart from the weights a' and b (which are determined during the calibration of the algorithm), the $BSCH_{MA}$ has only one parameter (x, the beam width). It can be seen that, for $x = 1$, $BSCH_{MA}$ is tantamount to CH_{MA}. Note that the complexity of this heuristic is given by Equation 6.7, since the main loop in the algorithm performs n iterations. In each iteration, $n - k + 1$ jobs are evaluated in each node x. This evaluation consists of obtaining the maximum processing time in the first stage, so each m_1 is evaluated (complexity $n^2 \cdot x \cdot m_1$), and assigning the selected job to the first available assembly machine, so each m_2 is evaluated ($O(n^2 \cdot x \cdot m_2)$). In addition, for each node x, the rest of nodes are

evaluated in order to select the best x combinations of nodes and jobs for the next iteration ($O(n^2 \cdot x^2)$).

$$O(max\{n^2 \cdot x \cdot m_1, n^2 \cdot x \cdot m_2, n^2 \cdot x^2\}) \tag{6.7}$$

6.3.1 Variable Beam Width

To the best of our knowledge, beam-search-based heuristics employed in the scheduling literature always use a constant beam width x. However, it is expected that the number of nodes analysed in each iteration has a large influence on the performance of this heuristic. To test this hypothesis, we carry out this study and analyse the behaviour of the $BSCH_{MA}$ when x may take different values. More specifically, we will test the following variants:

- Constant Beam Width: $BSCH_{MA}$ is tested with different values of x. In this manner, the influence of the beam width on the performance of the heuristic can be analysed.
- Ascending Beam Width: In this version, the heuristic, denoted as $BSCH_{ASC}$, starts selecting x nodes and the beam width increases by one unit as the beam search advances. The search is stronger in each iteration since the number of selected nodes is higher.
- Descending Beam Width: This version, labelled as $BSCH_{DESC}$, starts by selecting a number of nodes equal to $x + n - 1$, and the beam width decreases one by one as the number of iterations increases. Therefore, in the last iteration, x nodes are considered.
- V-shaped Beam Width: In this version, the beam width is modified taking a V-shape. Initially, x nodes are considered and, for each iteration k, the number of nodes is decreased one by one while $k \leq \frac{n}{2}$ and then it increases until $k = n$. We denote this version as $BSCH_V$.
- Peak-shaped Beam Width: The pattern of this version, labelled as $BSCH_P$, is completely opposed to the previous one. The initial beam width is x, and then it increases one by one whereas $k \leq n \cdot \frac{2}{3}$ and then it decreases also one by one until the last iteration.

Algorithm 2 Pseudo-code of the proposed beam-search-based constructive heuristic.

procedure BSCH_MA(x)
  Obtain a sequence $\Omega := (\omega_1, \ldots, \omega_n)$ by applying algorithm S2;
  Update $\mathscr{U}_1^l(u_{1,1}^l = \omega_l)$ and $\mathscr{S}_1^l(s_{1,1}^l = \varnothing)$.
  for $l = 1$ to x **do** $F_{1,l} = \psi_{\omega[l],0,l}$
  end for
  for $k = 2$ to n **do**
    // Candidate Nodes Creation
    Determination of IT_{jkl}, CT_{jkl};
    // Candidate Nodes Evaluation
    $B_{jkl} := F_{kl} + a' \cdot IT_{jkl} + CT_{jkl}\ \forall\ l = 1, \cdots, x$ and $\forall\ j = 1, \cdots, n-k+1$;
    // Candidate Nodes Selection
    for $l' = 1$ to x **do**
      Determination of the l'-th best candidate node according to non-decreasing B_{jkl} in iteration k. Denote by $branch[l']$ and $job[l']$ the value of l and j respectively, of that candidate.
    end for
    // Forecasting Phase
    for $l' = 1$ to x **do**
      Update $\mathscr{S}_{k+1}^{l'}$ and $\mathscr{U}_{k+1}^{l'}$ by removing job $\sqcap_{job[l'],k}^{branch[l']}$ from $\mathscr{U}_k^{branch[l']}$ and including in $\mathscr{S}_k^{branch[l']}$.
    $F_{k+1,l'} = F_{k,branch[l']} + b \cdot \psi_{job[l'],k,branch[l']}$;
    end for
  end for
  // Final evaluation
  Evaluate the total completion time of the scheduled jobs of each selected node and return the least one.
end procedure

We design and implement different versions, which are evaluated considering the next values of x, $x \in \{2, 5, 10, 15, \frac{n}{10}, n, n + \frac{n}{2}, 2n\}$.

6.4 Implemented heuristics

In order to determine the performance of the proposed heuristics (CH_{MA} and the best variants of the beam-search-based heuristic), these have been compared with existing heuristics for the problem, as well as with heuristics adapted from similar problems. More specifically, the heuristics used for the comparison are the following:

- Heuristics from the $DPm \rightarrow Pm||\sum_j C_j$ problem:
  - New heuristics proposed in Section 6, i.e., heuristic CH_{MA} in Section 6.2 and the variants of the $BSCH$ proposed in Section 6.3 that are in the Pareto frontier found in 6.5.2: $BSCH_{V(x=2)}$, $BSCH_{V(x=n/10)}$, $BSCH_{V(x=5)}$, $BSCH_{V(x=10)}$, $BSCH_{V(x=15)}$, $BSCH_{MA(x=n)}$ and $BSCH_{MA(x=n+n/2)}$.
  - SAK (Sung and Kim, 2008): This heuristic sorts the jobs in non decreasing order of $psum_j = \sum_{i=1,\ldots,m_1} p_{ij} + at_j$. Set k=1 and m=k+1, it exchanges the kth job and the mth job. If the total completion time is improved, it keeps the exchange. If not, $m = m$+1.
  - $NSDE$ (Al-Anzi and Allahverdi, 2012): This algorithm has been coded and run, but it has been discarded due to its computational effort is far from the rest of the heuristics and the quality of its solution is poor since it is specifically designed for the problem with two assembly machines.
- Heuristics adapted from the $DPm \rightarrow 1||\sum_j C_j$ problem. Since this problem is closely related to the one addressed in this chapter, it is interesting to test whether heuristics specifically designed for the problem with one assembly machine can be adapted to the problem under consideration. These adaptations are:
  - $TCK1$ and $TCK2$ (Tozkapan et al., 2003): The original $TCK1$ constructs m_1+1 indices for each job, according to $PTF_{ij} = t_{ij}$ and $PTS_j = at_j$. So, m_1+1 sequences are obtained by sorting the jobs in non decreasing order of these indicators, and the sequence with the lowest TCT is selected. The index PTS_j has been adapted to our problem so that $PTS_j = at_j/m_2$.

Similarly, $TCK2$ computes three indices for each job, so three sequences are obtained by sorting the jobs in non decreasing order of these indices, and the sequence yielding the lowest TCT is selected. The indices have been adapted to our problem taking into account m_2, i.e.,: $MPT_j = min\{p_{1j}, p_{2j}, \cdots, p_{m_1 j}, at_j/m_2\}$; $APT_j = \frac{1}{m_1+m_2}\sum_{i=1}^{m_1} p_{ij} + at_j/m_2$; and $MXPT_j = max\{p_{1j}, p_{2j}, \cdots, p_{m_1 j}, at_j\}$.

– $A1$ and $A2$ (Al-Anzi and Allahverdi, 2006a): These algorithms construct a sequence by iteratively appending a job at the end of a partial sequence. For algorithm $A1$, the job is chosen so that the following indicator is minimised:

$$A1_j = \max_{i=1,\ldots,m_1}\left\{\sum_{r=1}^{j-1} p_{i[r]} + p_{ij}\right\} \tag{6.8}$$

Note that this indicator does not require any adaptation to our problem. However, for algorithm $A2$ the indicator is adapted by dividing the assembly time by the number of assembly machines m_2, so the modified index is:

$$A2_j = \max_{i=1,\ldots,m_1}\left\{\sum_{r=1}^{j-1} p_{i[r]} + p_{ij}\right\} + \frac{at_j}{m_2} \tag{6.9}$$

– $S1$, $S2$ and $S3$ (Al-Anzi and Allahverdi, 2006a): $S1$ sorts the jobs in non decreasing order of at_j. Heuristic $S2$ is obtained by sorting the jobs in non decresing order of $\max_{i=1,\ldots,m_1}\{p_{ij}\}$ and, finally, heuristic $S3$ orders the jobs in non decreasing order of $\max_{i=1,\ldots,m_1}\{p_{ij}\} + at_k$. As with the previous heuristics, $S1$ and $S3$ have been adapted by dividing at_j by m_2.

– $G1$, $G2$, $G3$ and $G4$ (Lee, 2018): Each of these heuristics constructs a sequence by inserting the job with the smallest value of one of the following indicators: $G1_j = C_{[j]} - C_2^*$; $G2_j = C1_j^* - C1_{j-1}^*$; $G3_j = C1_j^* - C_2^*$ and $G4_j = C_{[j]} - C1_j^*$. These indicators can be used for our problem in a straightforward manner.

– FAP (Framinan and Perez-Gonzalez, 2017b): This heuristic appends one by one the unscheduled jobs at the end of a partial sequence by computing an estimate of the completion times of the unscheduled jobs which takes into account which stage is more important. This estimate has been adapted considering the number of assembly machines so, if the first

stage is dominant, then $FAP_l = C1^*_j + \frac{n-j+1}{n}(C_{1\bullet} + \frac{at_\bullet}{m_1+m_2})$. Otherwise, $FAP_l = \frac{at_{\omega_l}}{m_1+m_2} + \frac{n-j+1}{n}(C_{1\bullet} + \frac{at_\bullet}{m_1+m_2})$, where $C_{1\bullet}$ is the completion time in the first stage of an average artificial job composed of the unscheduled jobs, and $at_\bullet$ is the processing time of such artificial job in the second stage.

- Heuristics adapted from $DPm \rightarrow 0||\sum_j C_j$ problem. As mentioned above, the Customer Order (CO) scheduling problem is identical to the problem considered if the processing times of the second stage are zero. Therefore, it is also of interest to test how the adaptation of their best methods perform in our case. The most relevant methods for the order scheduling problem are:
  - $STPT$ (Sung and Yoon, 1998): A sequence is constructed sorting the jobs in ascending order of their sum of their processing times on the m_1 machines. In our case, $m_1 + m_2$ machines are considered.
  - $SMPT$ (Sung and Yoon, 1998): A sequence is constructed sorting the jobs in ascending order of their maximum processing time on the m_1 machines. As with the previous heuristic, $m_1 + m_2$ machines are considered.
  - ECT (Ahmadi et al., 2005; Leung et al., 2005): In this heuristic, the order with the earliest completion time is selected as the next to be sequenced. This heuristic does not require adaptation, as for each order, the completion time is computed according to Equation (2.3).
  - SFT_k and SFT_{k_O} (Framinan and Perez-Gonzalez, 2017a): SFT_k obtains iteratively a partial sequence using the ECT heuristic. Then, the jobs are iteratively removed from their position and re-inserted. The procedure is repeated until the so-obtained partial sequence does not returns a lower total completion time. SFT_{k_O} restarts the reinsertion phase whenever a better sequence is found and repeats the process until no further improvement is found.

6.5 Computational experience

In this section, we analyse the efficiency of the heuristics proposed in Section 6.2 and 6.3 compared with the implemented heuristics described in Section6.4. In Section 6.5.1 we perform a design of experiments to set up proper values for parameters a for

CH_{MA}, a' and b for $BSCH_{MA}$. In Section 6.5.2, the different versions of the beam-search heuristics presented in Section 6.3.1 are compared to obtain the best variants. Finally, in Section 6.5.3 and 6.5.4 the proposed heuristics are compared with the existing heuristics for the problem and for related problems in the benchmarks $\mathscr{B}_0$, $\mathscr{B}_1$ and $\mathscr{B}_2$. The experiments have been carried out comparing the algorithms by means of the indicators established in Section 3.2, using the sets of instances designed in Section 3.3 and following the conditions detailed in Section 3.4. To obtain a better estimation of the performance of all algorithms and based on the computational experiments carried out by Rad et al. (2009), Fernandez-Viagas and Framinan (2017) and Valente and Alves (2008), a total of 10 replicates are carried out for each instance and the results are averaged.

6.5.1 Experimental parameter tuning

In this section, a factorial design of experiments is performed to find the best values of the parameters of the two heuristics presented in Section 6.2 and 6.3. After some preliminary tests, it has been identified that the best values for the different parameters are in the ranges detailed below. More specifically, the following values are tested:

- Parameter a for the CH_{MA} heuristic described in Section 6.2. The following levels for a are tested: $a \in \{1, 2, 5, 10, 15, 20, 50, 200\}$.
- Parameters a' and b for the $BSCH_{MA}$ heuristic described in Section 6.3. The following levels for each parameter are tested (in total, there are 42 combinations):
  - $a' \in \{1, 2, 5, 10, 15, 20\}$
  - $b \in \{0, 1, 2, 3, 4, 5, 6\}$

To determine the best combination of parameters, the calibration benchmark $\mathscr{B}_{C0}$ presented in Section 3.3 is used. With this testbed, after proving that the normality and homoscedasticity assumptions are not fulfilled, a non-parametric Kruskal-Wallis test is performed. The results indicate that there are significant differences among the different values of parameter a. The best result is obtained for $a = 5$. Regarding the $BSCH_{MA}$ heuristic, it has been assumed that $x = n$ and the so-obtained results are compared. Wider ranks of the parameter a' have been tested, but the results have not

been improved. A non-parametric Kruskal-Wallis test is also carried out. The results indicate that there are significant differences between parameters a' and b since the significance of both parameters is equal to 0.000. The best combination is obtained for a'=2 and b=2. In Figures 6.2 and 6.3, the evolution of the $ARPD$ of the heuristics is shown according to the different experimental values of each parameter. It can be seen that the minimum $ARPD$ is obtained for $a = 5$ and for $a' = 2$ and $b = 2$. These values are used for the different versions of $BSCH$ in Section 6.5.2 and Section 6.5.3 regardless of the value of x.

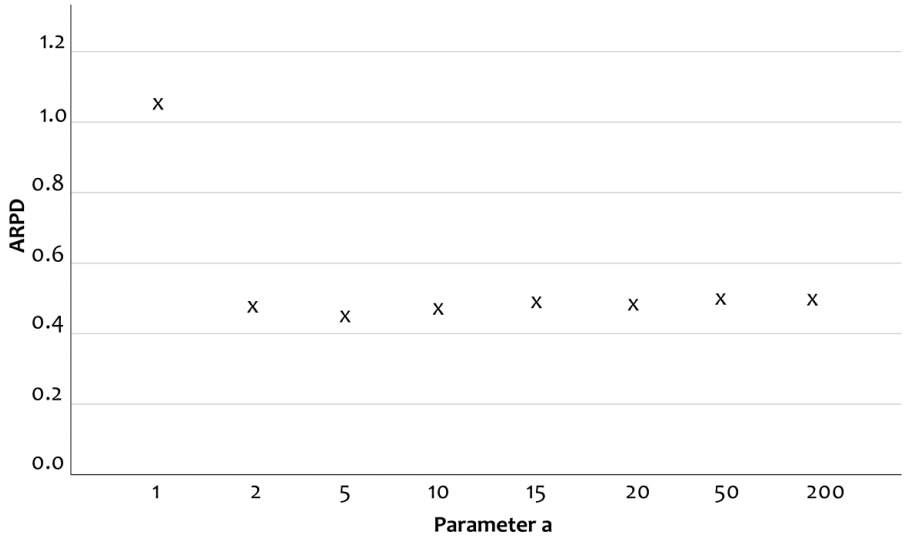

Figure 6.2: Evolution of $ARPD$ with different values of the parameter a.

6.5.2 Comparison of the different versions of $BSCH$ in benchmark $\mathscr{B}_0$

Prior to conducting a full comparison with the existing heuristics, the best variant of the beam-search-based heuristics is selected. To do so, the versions of $BSCH$ presented in Section 6.3.1 have been run on the 2,400 instances generated in Section 3.3. The results are summarised in Table 6.2 using the indicators defined in Section 3.2. The $ARPD$ values range from 1.320 ($BSCH_{MA(x=2)}$) to 0.523 ($BSCH_{MA(x=n+n/2)}$) whereas $ACPU$ values range from 0.997 to 0.004. Results are graphically shown in Figure 6.4 where the y-axis represents the $ARPD$ for each heuristic and the x-axis represents the ACPU.

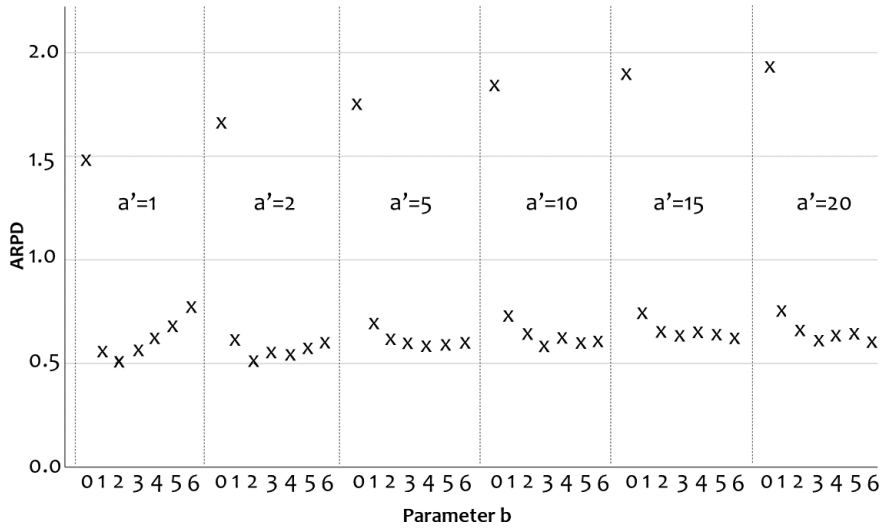

Figure 6.3: Evolution of $ARPD$ with different values of parameters a' and b.

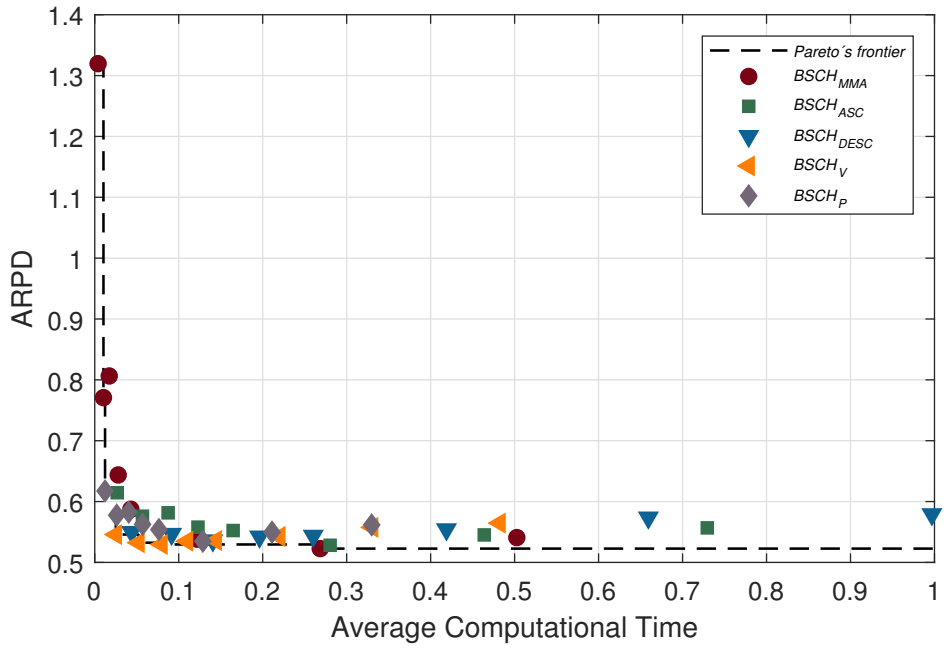

Figure 6.4: $ARPD$ versus average CPU times of the different versions of $BSCH$ with the Pareto frontier.

In view of these results, the following conclusions can be derived:

- $BSCH_{P(x=2)}$ with $ARPD = 0.617$ improves the variants $BSCH_{MA(x=2,x=5,x=n/10)}$ with $ARPD$ equal to 1.320, 0.806 and 0.771, respectively, using approximately a similar computational effort.
- $BSCH_{V(x=2)}$ with $ARPD = 0.546$ outperforms variants $BSCH_{MA(x=10)}$, $BSCH_{ASC(x=2)}$ and $BSCH_{P(x=5)}$, with $ARPD$ equal to 0.644, 0.550 and 0.578, respectively.
- $BSCH_{DESC(x=2)}$ with $ARPD = 0.550$ improves the variant $BSCH_{MA(x=15)}$ using the same $ACPU$.
- $BSCH_{V(x=5)}$ with $ARPD = 0.532$ outperforms variants $BSCH_{P(x=10)}$ and $BSCH_{ASC(x=5)}$, with $ARPD$ equal to 0.563 and 0.547, respectively, using the same computational effort.
- $BSCHV(x = n/10)$ with $ARPD = 0.523$ outperforms variants $BSCH_{DESC(x=5)}$ and $BSCH_{ASC(x=n/10)}$, with $ARPD$ equal to 0.547 and 0.536, respectively, using the same computational effort.
- For $x = n$, the variants $BSCH_{MA(x=n)}$ and $BSCH_{P(x=n)}$ yield a similar performance, being its $ARPD$ equal to 0.538 and 0.535, respectively. Moreover, the version $BSCH_{V(x=10)}$ obtains a similar $ARPD = 0.5349$ with less computational effort.
- The minimum $ARPD$ is achieved by $BSCH_{MA(x=n+n/2)}$, being the rest of the variants worse with respect to the quality of the solutions.
- The Pareto frontier (i.e., the efficient variants with respect to the quality of solutions and the computational effort) is formed by $BSCH_{MA(x=2)}$, $BSCH_{MA(x=5)}$, $BSCH_{P(x=2)}$, $BSCH_{V(x=2)}$, $BSCH_{V(x=5)}$, $BSCH_{V(x=n+/10)}$ and $BSCH_{MA(x=n+n/2)}$.
- The performance of $BSCH_{MA}$ worsens for $x = 2n$. For this value of x, this heuristic selects $2n$ candidates in each iteration, so there are more node candidates to be evaluated from the first iteration. Due to this poor result, it has not been considered.

i	H_i	p-value	Wilcoxon	$\alpha/(k-i-1)$	Holm's Procedure
1	$BSCH_{MA(x=10)} = BSCH_{ASC(x=2)}$	0.015	R	0.0083	-
2	$BSCH_{MA(x=10)} = BSCH_{V(x=2)}$	0.001	R	0.0100	R
3	$BSCH_{MA(x=5)} = BSCH_{P(x=2)}$	0.000	R	0.0125	R
4	$BSCH_{MA(x=10)} = BSCH_P(x=5)$	0.000	R	0.0167	R
5	$BSCH_{MA(x=10)} = BSCH_{V(x=2)}$	0.025	R	0.0125	R
6	$BSCH_{MA(x=15)}) = BSCH_{DESC(x=2)}$	0.000	R	0.0500	R

Table 6.1: Holm's procedure for comparing the different versions of $BSCH$.

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 Wilcoxon signed-rank test assuming a 0.95 confidence level, i.e., $\alpha = 0.05$. In Holm's test, the hypotheses are sorted in non-descending order of the p-values obtained in the Wilcoxon test. Each hypothesis is rejected if $p < \alpha/(k-i+1)$ where k is the total number of hypotheses. The results can be seen in Table 6.1, where R means that the hypothesis is rejected by Wilcoxon and/or Holm's procedure. As can be seen, hypothesis $BSCH_{(x=10)} = BSCH_{ASC(x=2)}$ is the only one that cannot be rejected by Holm's procedure, but it has to be noted that the $ARPD$ achieved by the latter version, equal to 0.6145, is considerably lower than the one obtained by $BSCH_{MA(x=10)}$. In summary, it can be concluded that the variants in the Pareto frontier in Figure 6.4 are efficient for the problem. However, as no variant obtains the best performance for all values of x, it can be also concluded that, if $x \leq n$ the best variant is $BSCH_V$, and if $x > n$, then $BSCH_{MA}$ is the most efficient one.

6.5.3 Comparison of heuristics in benchmark $\mathscr{B}_0$

These heuristics have been employed to solve the instances from the testbed in Section 3.3. The $ARPD$ and $ACPU$ are computed according to Eqs. (3.1) and (3.3), while the indicator $ARPT$ is computed using Equation (3.6). The detailed results of $ARPD$ in terms of $n \times m_1 \times m_2$ are shown in Tables 6.5- 6.10. The average results in terms of $ARPD$, $ACPU$, and $ARPT$ are shown in Table 6.3 and graphically in Figure 6.5. Note that, in this figure, the dispatching rules are not displayed in order to have a clearer interpretation of the results. In view of the results, a number of conclusions

x	2	5	n/10	10	15	n	n+n/2	n
				ARPD				
$BSCH_{MA}$	1.320	0.771	0.806	0.644	0.587	0.538	0.523	0.541
$BSCH_{ASC}$	0.550	0.547	0.536	0.542	0.552	0.528	0.545	0.557
$BSCH_{DESC}$	0.550	0.547	0.536	0.542	0.544	0.554	0.574	0.579
$BSCH_V$	0.546	0.532	0.523	0.535	0.536	0.542	0.558	0.565
$BSCH_P$	0.617	0.578	0.582	0.563	0.554	0.535	0.550	0.562
				ACPU				
$BSCH_{MA}$	0.004	0.010	0.017	0.028	0.043	0.122	0.269	0.503
$BSCH_{ASC}$	0.027	0.057	0.087	0.123	0.165	0.280	0.464	0.723
$BSCH_{DESC}$	0.044	0.092	0.141	0.196	0.260	0.419	0.659	0.997
$BSCH_V$	0.025	0.052	0.080	0.110	0.144	0.220	0.330	0.482
$BSCH_P$	0.648	0.608	0.613	0.593	0.585	0.566	0.581	0.592

Table 6.2: Summary of results of the different versions of $BSCH$.

can be drawn:

- CH_{MA} ($ARPD$=1.874) clearly outperforms heuristics $A1$, $A2$, $G1$, $G2$, $G3$ and $G4$ using a similar computational effort. It can be seen that CH_{MA} obtains very good results, evaluating only one job at each iteration and, consequently, consuming less computational time. Furthermore, CH_{MA} obtains a similar $ARPD$ to FAP, but our proposal requires much less CPU time.

- $S2$ and $TCK2$ with $ARPD$ equal to 15.253 and 7.525, respectively, are the best dispatching rules. Although the quality of the solution is low, these rules obtain a solution very fast.

- $BSCH_V$ ($x = 2$) with $ARPD$=0.546 outperforms CH_{MA}, with $ARPD$ equal to 1.874, using the same computational effort as can be checked in Figure 6.5.

- $BSCH_{MA(x=n)}$ with $ARPD$ =0.5382 outperforms SFT_k, SFT_{k_O} and SAK with $ARPD$ equal to 14.175, 9.323 and 12.679, respectively. Moreover, it can be pointed out that this version of the $BSCH_{MA}$ obtains the best result in terms of quality of the solution, $ARPD$.

- Taking into account these results and those obtained in the previous section,

Heuristic	$ARPD$	$ACPU$	$ARPT$	Heuristic	$ARPD$	$ACPU$	$ARPT$
CH_{MA}	1.874	0.0019	625.07	SAK	12.680	0.3596	102534.03
FAP	1.664	0.0114	3358.46	$STPT$	15.873	0.0000	1.64
$G1$	17.403	0.0019	615.74	$SMPT$	21.671	0.0000	2.36
$G2$	6.142	0.0018	605.78	ECT	17.403	0.0368	10559.23
$G3$	6.817	0.0018	590.66	SFT_k	14.175	0.1226	34958.62
$G4$	17.465	0.0018	583.61	SFT_{k_O}	9.323	0.1590	45640.85
$A1$	6.029	0.0018	595.67	$BSCH_{V(x=2)}$	0.546	0.0251	7482.71
$A2$	9.257	0.0018	593.95	$BSCH_{V(x=5)}$	0.532	0.0522	15574.12
$S1$	19.274	0.0000	1.00	$BSCH_{V(x=n/10)}$	0.530	0.0796	23690.42
$S2$	15.253	0.0000	2.13	$BSCH_{V(x=10)}$	0.535	0.1100	32796.73
$S3$	16.649	0.0000	2.30	$BSCH_{V(x=15)}$	0.536	0.1442	43052.43
$TCK1$	15.313	0.0008	242.55	$BSCH_{MA(x=n)}$	0.538	0.1220	36705.90
$TCK2$	7.525	0.0003	109.24	$BSCH_{MA(x=n+n/2)}$	0.523	0.2685	78987.16

Table 6.3: Summary of results of the different heuristics.

the group of the most efficient heuristics is formed by the dispatching rule $S2$, the existing and adapted heuristic $TCK2$ and the proposed versions of the beam-search-based constructive heuristic: $BSCH_{V(x=2)}$, $BSCH_{V(x=n/10)}$, $BSCH_{V(x=5)}$, $BSCH_{V(x=10)}$, $BSCH_{V(x=15)}$, $BSCH_{MA(x=n)}$ and $BSCH_{MA(x=n+n/2)}$.

In order to check the statistical significance of these results, Holm's procedure is used as in the previous computational experience. However, each hypothesis is now analysed using a non-parametric Mann-Whitney test assuming a 95% confidence level (i.e., $\alpha = 0.05$) to establish de p-value of each hypothesis. The results are shown in Table 6.4. As can be seen, each p-value is 0.000, so all hypotheses can be rejected. In summary, it can be concluded that the proposed heuristics outperform the existing algorithms for the problem under consideration, as well as the adaptations of efficient algorithms for related problems.

6.5.4 Comparison of heuristics in benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$

The objective of this section is to analyse the actual state-of-the-art heuristics in the two new benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$, generated in Chapter 5. This experimentation can help us to make a solid comparison of the existing approximate methods and identify the most efficient ones to solve the *SA* and *MA* variants.

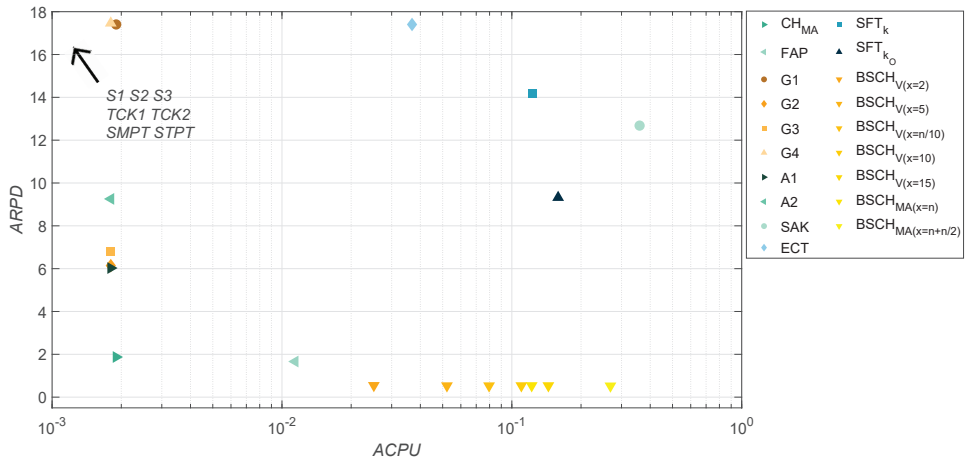

Figure 6.5: $ARPD$ versus $ACPU$. $ACPU$ (x-axis) is shown in logarithmic scale.

i	H_i	p-value	Mann-Whitney	$\alpha/(k-i-1)$	Holm's Procedure
1	$CH_{MA} = A1$	0.000	R	0.0100	R
2	$S2 = S3$	0.000	R	0.0125	R
3	$TCK2 = TCK1$	0.000	R	0.0167	R
4	$BSCH_{V(x=2)} = CH_{MA}$	0.000	R	0.0250	R
5	$BSCH_{MA(x=n)} = SFT_{k_O}$	0.000	R	0.0500	R

Table 6.4: Mann-Whitney's procedure. (R indicates that the hypothesis can be rejected)

To evaluate the efficiency of the different heuristics, the $ARPD$ values are computed following the Equation (3.2), where $\sum OF_{hs}$ is the value of the objective function found by each heuristic h and MIN is the minimum known solution for each instance. Moreover, since there are heuristics with different number of steps in their procedure, the $ARPT$ indicator is also computed according to Equation (3.6).

Table 6.11 shows the average results, and is organised as follows: column 1 and column 2 indicate the problem and the heuristics for each variant. Results for the benchmark of SA, $\mathscr{B}_1$, 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, $\mathscr{B}_2$, are presented in columns 11 to 18, and the columns are organised in the same

n	m_1	m_2	$A1$	$A2$	$S1$	$S2$	$S3$	$TCK1$	$TCK2$	$G1$	$G2$	$G3$
30	2	2	9.20	9.17	24.30	11.09	12.68	18.99	7.20	16.13	9.41	10.46
		4	5.14	9.24	19.52	7.74	12.18	14.94	5.79	18.96	5.21	6.07
		6	3.71	7.00	18.13	6.30	10.91	13.37	5.62	18.70	3.83	4.63
		8	3.15	6.78	15.25	5.16	10.56	13.19	6.12	18.68	3.18	3.53
	4	2	7.19	10.43	22.19	18.59	20.86	15.86	8.69	16.02	7.35	7.04
		4	5.44	7.76	17.84	17.28	17.44	13.59	6.89	15.87	5.25	4.51
		6	3.28	7.50	15.81	14.54	15.25	12.08	6.32	15.21	3.54	3.72
		8	1.91	6.35	14.47	12.69	13.35	11.28	5.79	13.44	2.06	2.17
	6	2	5.53	8.91	19.79	17.96	18.17	13.71	8.59	13.18	5.56	5.70
		4	4.52	7.89	15.99	16.14	16.60	11.75	8.35	14.85	4.77	4.28
		6	2.75	6.69	15.42	14.14	13.87	11.05	6.27	13.22	2.95	2.81
		8	2.53	6.06	13.36	12.56	12.26	9.74	5.28	11.52	2.78	3.07
	8	2	3.96	7.97	17.79	17.11	16.97	11.74	7.87	12.50	4.12	4.18
		4	3.25	7.30	14.83	14.80	15.29	10.23	8.32	13.79	3.14	3.81
		6	2.60	6.00	14.85	13.55	14.45	9.44	6.00	12.56	2.61	3.04
		8	2.20	5.77	12.99	11.79	12.46	8.73	6.12	11.42	2.14	2.12
40	2	2	11.10	10.59	26.79	12.54	15.05	22.53	8.78	18.96	11.16	11.71
		4	9.01	9.08	24.09	10.61	12.55	19.51	7.05	20.26	9.34	9.74
		6	5.99	7.71	18.97	7.57	11.22	16.29	5.81	20.76	6.25	6.26
		8	4.68	7.25	19.24	6.52	10.12	16.38	5.67	18.92	4.92	5.34
	4	2	8.26	10.86	22.33	22.14	22.67	17.33	8.78	16.85	8.30	8.80
		4	4.80	9.35	20.05	18.35	19.37	15.37	8.39	18.68	4.95	6.31
		6	3.47	8.22	18.04	15.38	15.36	12.90	5.49	15.74	3.75	4.12
		8	3.13	6.82	16.02	15.06	14.10	12.52	6.34	14.83	3.35	3.65
	6	2	5.73	9.81	19.99	18.53	19.77	15.30	9.23	14.79	5.73	6.22
		4	3.47	8.60	16.98	16.47	17.43	13.03	6.99	14.99	3.70	4.66
		6	2.64	7.31	16.33	15.28	17.58	11.61	7.07	15.40	2.70	4.64
		8	2.72	7.40	14.61	15.03	15.25	11.38	6.23	14.96	3.16	3.85
	8	2	5.05	10.40	17.29	17.49	18.41	13.15	9.28	14.24	4.99	6.25
		4	4.01	8.50	17.74	16.35	17.37	12.39	7.52	14.37	4.02	4.62
		6	3.12	7.07	14.69	13.65	14.20	11.23	7.10	13.14	3.16	3.98
		8	3.03	6.75	14.40	14.38	13.77	10.27	6.53	13.22	2.84	3.56
50	2	2	13.42	11.22	28.55	14.33	16.29	23.57	8.47	20.92	13.40	13.74
		4	8.71	9.03	22.35	10.50	14.14	20.24	6.35	23.42	9.13	9.29
		6	7.54	8.46	21.27	9.46	11.63	18.07	5.81	22.76	7.89	8.14
		8	5.94	8.14	20.89	7.96	11.96	16.28	5.99	21.62	6.09	6.40
	4	2	8.32	11.42	23.29	20.18	21.99	19.14	9.04	17.25	8.57	9.56
		4	6.77	10.20	21.29	19.80	20.41	16.66	7.10	18.42	6.92	8.20
		6	4.87	9.35	19.31	17.30	18.02	14.80	7.02	18.21	5.20	6.14
		8	4.28	8.10	17.96	15.63	17.06	14.32	6.74	17.88	4.16	5.06
	6	2	7.09	10.77	20.26	18.94	20.44	15.98	9.23	16.39	6.93	8.58
		4	4.61	9.90	18.71	17.78	19.04	13.70	7.95	16.46	4.96	5.34
		6	4.63	8.57	17.24	16.73	17.60	13.42	7.40	16.21	4.48	4.40
		8	2.80	7.60	15.45	13.80	14.99	11.65	6.98	14.65	3.13	4.01

Table 6.5: RPD of heuristics per levels of n, m_1 and m_2 (I).

n	m_1	m_2	$A1$	$A2$	$S1$	$S2$	$S3$	$TCK1$	$TCK2$	$G1$	$G2$	$G3$
50	8	2	5.72	11.00	18.08	18.57	19.01	14.36	8.97	14.56	5.76	7.41
		4	5.28	10.14	17.74	16.34	17.33	13.31	8.09	15.61	5.55	6.45
		6	3.46	8.54	15.83	14.88	15.60	11.60	7.16	14.35	3.49	4.54
		8	3.62	8.01	14.41	13.77	15.30	10.65	6.44	14.91	3.57	3.74
60	2	2	12.88	9.47	25.62	14.37	14.00	21.64	8.33	19.13	12.91	12.72
		4	10.08	9.34	24.79	12.03	12.76	20.60	6.78	22.50	10.29	11.91
		6	7.82	8.75	22.70	9.36	12.28	19.44	6.34	22.51	8.01	8.94
		8	6.61	9.40	20.37	8.25	12.60	18.21	6.25	22.84	6.97	7.41
	4	2	12.45	12.17	23.34	22.17	23.11	19.50	10.19	18.76	12.26	11.77
		4	6.27	10.78	20.02	18.65	19.48	17.02	7.78	19.33	6.45	7.64
		6	5.86	9.59	19.02	16.76	18.05	15.62	7.33	19.64	6.46	6.80
		8	4.22	8.13	18.88	17.09	17.44	15.28	6.01	18.48	4.61	5.39
	6	2	7.95	11.19	20.84	19.67	20.34	17.11	9.40	16.08	7.80	8.79
		4	5.52	10.12	19.47	17.52	18.56	14.74	8.35	17.67	5.51	6.76
		6	5.89	9.35	17.01	16.42	18.26	14.12	7.13	16.81	5.72	5.89
		8	3.56	8.01	16.74	15.72	16.27	13.00	6.39	16.70	3.56	4.14
	8	2	6.22	10.98	20.51	18.22	19.04	14.90	8.88	15.01	6.33	8.25
		4	5.78	10.76	17.78	16.53	20.04	13.59	9.12	17.19	5.89	5.92
		6	3.61	9.07	16.11	15.86	17.49	12.43	7.60	16.04	3.32	4.87
		8	3.46	7.94	14.70	14.96	15.51	12.05	6.45	15.14	3.55	4.18
70	2	2	15.07	10.94	28.62	15.98	14.85	25.38	8.08	20.12	15.02	16.04
		4	12.71	10.56	25.65	13.94	13.77	22.80	6.84	24.38	13.03	14.30
		6	9.59	9.10	22.06	11.16	13.66	20.37	6.69	24.97	9.82	10.47
		8	7.10	9.14	20.80	8.81	12.64	19.04	6.23	23.07	7.47	8.25
	4	2	10.22	12.18	23.65	21.75	23.24	19.17	8.44	18.22	10.11	10.79
		4	6.92	10.94	21.36	19.15	20.09	17.40	7.46	19.73	6.91	7.81
		6	6.16	9.57	20.20	17.99	18.02	17.22	7.03	18.99	6.31	7.26
		8	3.85	9.06	18.49	16.42	17.73	15.63	7.16	19.13	4.16	5.32
	6	2	6.18	11.64	20.34	18.38	19.60	16.27	9.24	15.66	6.10	8.94
		4	7.02	10.75	18.22	17.43	19.38	15.34	7.80	17.86	7.27	7.12
		6	5.86	10.62	17.71	16.20	18.16	14.34	7.06	17.98	6.04	6.50
		8	4.91	9.37	16.77	15.12	18.13	13.25	7.24	17.74	5.09	5.49
	8	2	6.30	11.69	19.49	18.13	20.81	15.71	9.81	15.93	6.47	8.10
		4	4.70	9.45	17.65	16.97	18.99	13.65	8.14	16.18	4.79	6.11
		6	4.95	9.56	18.13	16.37	17.43	13.57	7.04	16.44	4.74	5.91
		8	4.67	9.71	15.41	14.19	15.30	11.01	7.09	14.50	4.15	6.11
Average			5.85	9.08	19.06	15.05	16.44	15.10	7.35	17.18	5.96	6.65

Table 6.6: RPD of heuristics per levels of n, m_1 and m_2 (II).

$x = 2$

n	m_1	m_2	$G4$	ECT	$STPT$	$SMPT$	SFT_k	$SFT_{k_{OPT}}$	SAK	FAP	CH_{MA}	$BSCH_{MA}$
30	2	2	16.53	16.13	13.38	22.13	10.65	5.86	9.62	1.51	1.84	1.63
		4	19.12	18.96	18.02	23.04	14.32	6.81	13.17	1.32	1.39	1.06
		6	18.80	18.70	18.70	21.81	13.55	3.98	13.54	1.39	1.14	0.83
		8	18.71	18.68	18.51	20.78	13.84	4.14	13.35	1.18	1.19	0.72
	4	2	16.23	16.02	13.53	22.91	12.42	10.19	9.78	1.51	1.74	1.21
		4	15.95	15.87	13.99	20.86	11.80	6.52	10.17	1.33	1.52	0.75
		6	15.24	15.21	15.41	18.27	11.41	6.49	11.19	1.35	1.44	0.98
		8	13.48	13.44	13.02	15.40	10.13	5.50	9.25	1.08	1.10	0.66
	6	2	13.31	13.18	11.04	19.43	10.25	8.60	8.05	1.10	1.73	1.37
		4	14.91	14.85	13.97	18.41	11.07	7.12	9.92	1.44	1.64	1.04
		6	13.26	13.22	12.61	15.53	9.69	6.42	9.22	1.10	1.23	0.60
		8	11.53	11.52	11.00	13.46	8.48	5.38	7.86	1.38	1.14	0.75
	8	2	12.58	12.50	9.59	19.38	9.26	7.57	6.89	1.26	1.59	1.09
		4	13.82	13.79	12.75	17.42	10.55	6.79	9.25	1.28	1.68	0.97
		6	12.59	12.56	11.85	14.71	9.17	5.56	8.08	1.05	1.22	0.78
		8	11.44	11.42	11.18	13.30	8.47	5.50	8.02	1.06	1.10	0.60
40	2	2	19.30	18.96	16.44	25.81	14.74	10.09	12.76	1.80	2.32	1.41
		4	20.38	20.26	19.02	24.89	15.51	7.32	14.84	1.45	1.70	1.24
		6	20.80	20.76	20.15	24.47	16.18	4.29	15.93	1.72	1.68	0.87
		8	18.99	18.92	18.88	21.71	14.74	4.37	14.71	1.48	1.31	0.92
	4	2	16.99	16.85	13.01	23.65	13.37	11.23	9.95	1.48	2.05	1.07
		4	18.76	18.68	16.64	23.56	14.93	8.56	13.00	1.34	1.72	1.10
		6	15.77	15.74	14.87	18.54	12.39	7.57	11.99	1.34	1.46	0.68
		8	14.86	14.83	14.61	17.04	11.42	6.89	11.15	1.45	1.49	0.67
	6	2	14.88	14.79	11.85	21.97	12.21	9.72	8.88	1.31	1.87	1.13
		4	15.01	14.99	13.52	18.94	12.05	9.11	10.26	1.53	1.46	0.82
		6	15.43	15.40	15.22	18.60	12.15	7.58	11.49	1.30	1.45	0.90
		8	14.98	14.96	14.16	17.24	11.60	7.18	10.96	1.26	1.22	0.96
	8	2	14.32	14.24	11.18	20.01	11.68	10.69	8.25	1.20	1.99	1.41
		4	14.38	14.37	12.78	18.34	11.52	8.80	9.78	1.40	1.68	0.90
		6	13.16	13.14	12.19	16.11	10.35	7.25	9.16	1.13	1.24	0.84
		8	13.24	13.22	12.72	15.03	10.39	6.34	9.52	1.27	1.37	0.74
50	2	2	21.18	20.92	17.83	28.11	16.47	11.29	14.28	1.74	2.09	1.78
		4	23.52	23.42	22.21	29.15	18.85	7.99	18.31	1.81	1.89	1.10
		6	22.83	22.76	21.99	26.46	18.43	7.51	18.13	1.75	1.87	1.08
		8	21.65	21.62	21.42	24.06	17.36	5.38	17.44	1.48	1.43	1.11
	4	2	17.36	17.25	12.86	25.11	14.54	12.71	10.36	1.63	2.61	1.59
		4	18.48	18.42	16.80	23.20	15.01	10.70	13.70	1.49	1.92	1.47
		6	18.24	18.21	17.72	21.47	14.98	8.87	14.46	1.62	1.80	1.07
		8	17.91	17.88	17.29	20.28	14.48	7.54	14.14	1.55	1.67	0.84
	6	2	16.48	16.39	11.78	23.31	13.54	11.99	9.24	1.60	2.19	1.52
		4	16.48	16.46	14.98	20.79	13.73	10.27	11.90	1.35	1.48	1.07
		6	16.23	16.21	15.46	19.08	13.26	9.12	12.27	1.43	1.54	0.95
		8	14.66	14.65	14.46	16.86	11.91	8.57	11.36	1.49	1.57	0.97

Table 6.7: RPD of heuristics per levels of n, m_1 and m_2 (III).

x=2

n	m_1	m_2	$G4$	ECT	$STPT$	$SMPT$	SFT_k	$SFT_{k_{OPT}}$	SAK	FAP	CH_{MA}	$BSCH_{MA}$
50	8	2	14.59	14.56	10.82	20.67	12.31	11.49	8.77	1.35	1.99	1.51
		4	15.62	15.61	14.18	19.32	12.88	10.78	11.28	1.12	1.72	1.10
		6	14.36	14.35	13.74	16.91	12.00	9.55	11.11	1.34	1.65	0.79
		8	14.92	14.91	13.95	17.07	12.25	8.65	11.01	1.31	1.32	0.87
60	2	2	19.28	19.13	15.34	27.09	15.87	9.01	12.38	1.52	2.11	1.57
		4	22.61	22.50	21.33	27.94	18.78	9.49	18.06	1.82	2.08	1.31
		6	22.58	22.51	21.78	26.50	18.75	8.67	18.43	1.79	1.98	1.45
		8	22.89	22.84	22.42	25.81	19.08	7.01	18.67	1.65	1.80	1.34
	4	2	18.82	18.76	14.25	26.32	16.14	13.68	11.63	2.00	2.60	1.66
		4	19.39	19.33	17.60	23.81	16.38	12.04	14.78	1.56	2.05	1.28
		6	19.67	19.64	18.49	23.16	16.26	9.76	15.39	1.57	1.73	0.97
		8	18.51	18.48	17.73	21.29	15.26	8.49	14.86	1.52	1.61	1.05
	6	2	16.17	16.08	12.47	22.79	14.04	12.80	10.26	1.80	2.48	1.74
		4	17.69	17.67	16.37	22.48	14.96	11.69	13.38	1.51	1.82	1.34
		6	16.83	16.81	16.27	20.04	14.17	10.59	13.50	1.41	1.79	1.10
		8	16.71	16.70	16.26	19.19	13.70	8.92	12.97	1.27	1.54	0.71
	8	2	15.04	15.01	11.16	21.46	12.89	12.15	8.82	1.60	2.34	1.45
		4	17.20	17.19	15.71	21.65	14.56	11.42	13.04	1.54	1.93	1.40
		6	16.06	16.04	15.28	19.01	13.43	9.89	12.14	1.43	1.61	1.13
		8	15.15	15.14	14.13	17.34	12.40	9.34	11.49	1.45	1.54	0.97
70	2	2	20.32	20.12	16.70	27.63	16.64	11.13	14.01	2.14	2.64	1.92
		4	24.40	24.38	23.21	30.12	21.08	11.46	20.02	1.92	2.05	1.49
		6	25.02	24.97	24.58	29.15	21.02	8.82	21.27	1.83	1.93	1.23
		8	23.11	23.07	22.69	26.10	19.60	7.47	19.32	1.75	1.82	1.10
	4	2	18.31	18.22	13.27	26.13	15.85	13.75	11.07	2.01	2.55	1.89
		4	19.75	19.73	18.15	24.55	17.12	13.78	15.53	1.63	2.10	1.39
		6	19.02	18.99	17.92	22.48	16.02	11.60	15.51	1.69	1.87	1.07
		8	19.16	19.13	18.72	21.72	16.14	10.33	15.94	1.53	1.64	1.11
70	6	2	15.72	15.66	12.70	22.34	13.85	12.78	10.52	1.69	2.35	1.71
		4	17.87	17.86	15.35	22.61	15.48	11.88	12.74	1.50	2.10	1.41
		6	18.00	17.98	17.13	21.26	15.52	12.77	14.09	1.64	1.74	1.17
		8	17.74	17.74	17.23	20.32	15.12	11.22	14.25	1.71	1.72	1.11
	8	2	16.00	15.93	11.77	22.87	14.30	13.34	9.69	1.60	2.44	1.60
		4	16.18	16.18	14.13	21.07	13.99	11.74	11.67	1.41	1.72	1.28
		6	16.45	16.44	14.87	19.78	13.95	11.36	12.39	1.30	1.68	1.09
		8	14.46	15.57	15.83	17.70	13.24	8.92	13.30	3.20	1.58	0.89
Average			17.24	17.19	15.68	21.45	13.97	9.11	12.49	1.51	1.75	1.14

Table 6.8: RPD of heuristics per levels of n, m_1 and m_2 (IV).

			$x = n/10$	$x = 5$	$x = 10$	$x = 15$	$x = 10$	$x = 15$	$x = n$
n	m_1	m_2	$BSCH_{MA}$	$BSCH_{MA}$	$BSCH_{MA}$	$BSCH_{MA}$	$BSCH_V$	$BSCH_V$	$BSCH_{MA}$
30	2	2	1.17	0.73	0.42	0.29	0.58	0.45	0.37
		4	0.68	0.46	0.39	0.45	0.51	0.40	0.52
		6	0.56	0.46	0.47	0.51	0.44	0.51	0.39
		8	0.45	0.32	0.29	0.23	0.37	0.37	0.21
	4	2	1.03	0.68	0.67	0.49	0.42	0.57	0.61
		4	0.63	0.52	0.43	0.54	0.40	0.52	0.45
		6	0.91	0.56	0.38	0.46	0.51	0.56	0.46
		8	0.48	0.23	0.30	0.31	0.44	0.29	0.46
	6	2	0.78	0.69	0.60	0.64	0.72	0.50	0.65
		4	0.67	0.54	0.66	0.58	0.59	0.62	0.75
		6	0.46	0.48	0.40	0.41	0.48	0.54	0.49
		8	0.45	0.38	0.40	0.37	0.35	0.39	0.34
	8	2	0.94	0.45	0.48	0.59	0.62	0.61	0.78
		4	0.76	0.69	0.53	0.67	0.60	0.56	0.68
		6	0.55	0.41	0.52	0.52	0.47	0.42	0.45
		8	0.38	0.46	0.31	0.44	0.54	0.45	0.49
40	2	2	1.02	0.75	0.47	0.35	0.67	0.56	0.23
		4	0.75	0.56	0.49	0.66	0.43	0.51	0.39
		6	0.57	0.45	0.41	0.41	0.43	0.40	0.49
		8	0.49	0.63	0.54	0.44	0.45	0.47	0.33
	4	2	0.60	0.61	0.59	0.37	0.37	0.44	0.44
		4	0.67	0.54	0.45	0.56	0.57	0.66	0.47
		6	0.58	0.54	0.33	0.38	0.42	0.41	0.58
		8	0.47	0.33	0.41	0.35	0.40	0.39	0.45
	6	2	0.75	0.51	0.41	0.43	0.44	0.35	0.53
		4	0.61	0.57	0.37	0.32	0.50	0.39	0.45
		6	0.51	0.45	0.49	0.48	0.39	0.43	0.48
		8	0.54	0.42	0.32	0.31	0.41	0.42	0.43
	8	2	1.01	0.67	0.57	0.35	0.41	0.48	0.49
		4	0.67	0.58	0.56	0.52	0.39	0.36	0.47
		6	0.58	0.53	0.38	0.45	0.48	0.51	0.45
		8	0.51	0.42	0.34	0.42	0.36	0.39	0.43
50	2	2	0.64	0.64	0.56	0.27	0.52	0.51	0.08
		4	0.44	0.44	0.54	0.39	0.45	0.38	0.22
		6	0.58	0.58	0.43	0.25	0.59	0.50	0.32
		8	0.68	0.68	0.35	0.30	0.41	0.42	0.29
	4	2	0.79	0.79	0.55	0.36	0.30	0.32	0.22
		4	0.77	0.77	0.61	0.51	0.46	0.40	0.32
		6	0.56	0.56	0.42	0.38	0.35	0.31	0.39
		8	0.53	0.53	0.58	0.47	0.40	0.44	0.34
	6	2	0.66	0.66	0.50	0.36	0.47	0.36	0.35
		4	0.56	0.56	0.44	0.32	0.47	0.55	0.26
		6	0.59	0.59	0.42	0.41	0.48	0.44	0.33
		8	0.45	0.45	0.37	0.35	0.46	0.38	0.34

Table 6.9: RPD of heuristics per levels of n, m_1 and m_2 (V).

			$x = n/10$	$x = 5$	$x = 10$	$x = 15$	$x = 10$	$x = 15$	$x = n$
n	m_1	m_2	$BSCH_{MA}$	$BSCH_{MA}$	$BSCH_{MA}$	$BSCH_{MA}$	$BSCH_V$	$BSCH_V$	$BSCH_{MA}$
50	8	2	0.71	0.71	0.51	0.40	0.39	0.44	0.35
		4	0.75	0.75	0.60	0.44	0.55	0.35	0.37
		6	0.66	0.66	0.47	0.42	0.43	0.37	0.30
		8	0.49	0.49	0.43	0.29	0.46	0.50	0.40
60	2	2	0.76	0.81	0.55	0.44	0.42	0.36	0.23
		4	0.79	0.73	0.64	0.40	0.47	0.54	0.24
		6	0.71	0.63	0.64	0.39	0.54	0.48	0.28
		8	0.65	0.68	0.45	0.43	0.42	0.46	0.34
	4	2	0.79	0.75	0.40	0.39	0.38	0.37	0.12
		4	0.67	0.82	0.45	0.45	0.53	0.45	0.32
		6	0.64	0.48	0.61	0.51	0.56	0.43	0.32
		8	0.45	0.40	0.39	0.31	0.34	0.36	0.40
	6	2	0.79	0.77	0.46	0.41	0.31	0.31	0.23
		4	0.67	0.58	0.36	0.46	0.36	0.28	0.26
		6	0.42	0.50	0.42	0.49	0.42	0.50	0.38
		8	0.50	0.47	0.44	0.33	0.45	0.44	0.33
	8	2	0.80	0.78	0.52	0.47	0.35	0.26	0.29
		4	0.72	0.61	0.60	0.45	0.45	0.46	0.38
		6	0.53	0.50	0.36	0.35	0.34	0.35	0.31
		8	0.49	0.63	0.36	0.32	0.39	0.44	0.37
70	2	2	0.57	0.79	0.57	0.38	0.64	0.44	0.10
		4	0.58	0.75	0.55	0.45	0.55	0.54	0.18
		6	0.56	0.77	0.44	0.34	0.44	0.43	0.37
		8	0.51	0.80	0.47	0.36	0.54	0.52	0.36
	4	2	0.65	0.81	0.62	0.38	0.38	0.38	0.11
		4	0.55	0.61	0.36	0.35	0.30	0.36	0.31
		6	0.63	0.63	0.49	0.43	0.42	0.34	0.27
		8	0.52	0.55	0.43	0.34	0.30	0.29	0.30
	6	2	0.51	0.69	0.62	0.41	0.43	0.34	0.19
		4	0.59	0.70	0.40	0.44	0.37	0.32	0.26
		6	0.55	0.64	0.37	0.37	0.39	0.36	0.40
		8	0.59	0.63	0.62	0.39	0.43	0.32	0.47
	8	2	0.64	0.79	0.46	0.36	0.37	0.26	0.19
		4	0.58	0.67	0.51	0.48	0.43	0.44	0.29
		6	0.54	0.58	0.36	0.37	0.31	0.31	0.21
		8	0.53	0.71	0.50	0.43	0.37	0.35	0.28
Average			0.63	0.60	0.47	0.41	0.45	0.43	0.36

Table 6.10: RPD of heuristics per levels of n, m_1 and m_2 (VI).

way as for $\mathscr{B}_1$. The Pareto set for each variant is indicated in bold in Table 6.11.

Figures 6.6, 6.7 and 6.8 show the confidence intervals for the $ARPD$ values of the heuristics designed to solve the *SA* variant, providing good results solving benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$. In the case of Figures 6.9, 6.10 and 6.11, they show the confidence intervals for the values of $ARPD$ of the best heuristics, designed to solve the *MA* variant, solving both benchmarks. These figures show the values of $ARPD$ per levels of n in $\mathscr{B}_1$ and per levels of n and m_2 in $\mathscr{B}_2$, respectively. From the analysis of Table 6.11 and Figures 6.6-6.11, some comments can be made:

- The heuristics designed to solve the *SA* variant performs similarly in both benchmarks, $\mathscr{B}_1$ and $\mathscr{B}_2$. Regarding the results shown in Table 6.11, 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 Section 6.4 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 Figures 6.6, 6.7 and 6.8.

- The heuristics designed for *CO* are not suitable for solving *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 $SFT_{k_{OPT}}$ 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_{MA} yields a good performance in both benchmarks, with an $ARPD$ lower than 1 in both cases (see Table 6.11). Taking into account the results shown in Figures 6.9, 6.10 and 6.11, in terms of the number of jobs, CH_{MA} shows a considerable difference between $n = 50$ and $n = 100$ in benchmark $\mathscr{B}_2$, while it performs similarly for $n \geq 150$ in both benchmarks, $\mathscr{B}_1$ and $\mathscr{B}_2$. CH_{MA} 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 $BSCH_{MA(x=n)}$ for $\mathscr{B}_1$, and $BSCH_{V(x=2)}$ and $BSCH_{V(x=n)}$ for $\mathscr{B}_1$, 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 becomes similar. Note that, with respect to m_2, the values of $ARPD$ of $BSCH_{V(x=2)}$, $BSCH_{V(x=n)}$ and $BSCH_{MA(x=n)}$ increase, but in all cases it is less than 1. Therefore, it can be pointed out that these versions are the best heuristics to solve variants *SA* and *MA*.

- Comparing the results obtained from this evaluation and that carried out in Section 6.5.3, several changes can be observed in the relative performance of some heuristics, possibly due to the lack of adequacy in the testbed used in the referred work. In this evaluation, the heuristics $G1$ and $G4$ are more efficient than $G2$ and $G3$, while in the previous evaluation $G2$ and $G3$ yield better results. The heuristic $A2$ performs better than $A1$ in benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$, while in the previous experimentation the opposite happens. Regarding $S1$, $S2$ and $S3$, in benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$, $S3$ is more efficient than $S1$ and $S2$, being similar to the performance of the two latter. On the contrary, in the previous section, $S2$ yields a better result than $S1$ and $S3$, and $S3$ is also more efficient than $S1$.

Finally, to summarise the main aspects of these results, it is worth noting that, in terms of 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 a lower number of jobs. Note that this behaviour had not been previously detected, as the existing testbeds did not include instances with such large number of jobs. Taking into account this information, the methods recommended to solve the variants considered are as follows.

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

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

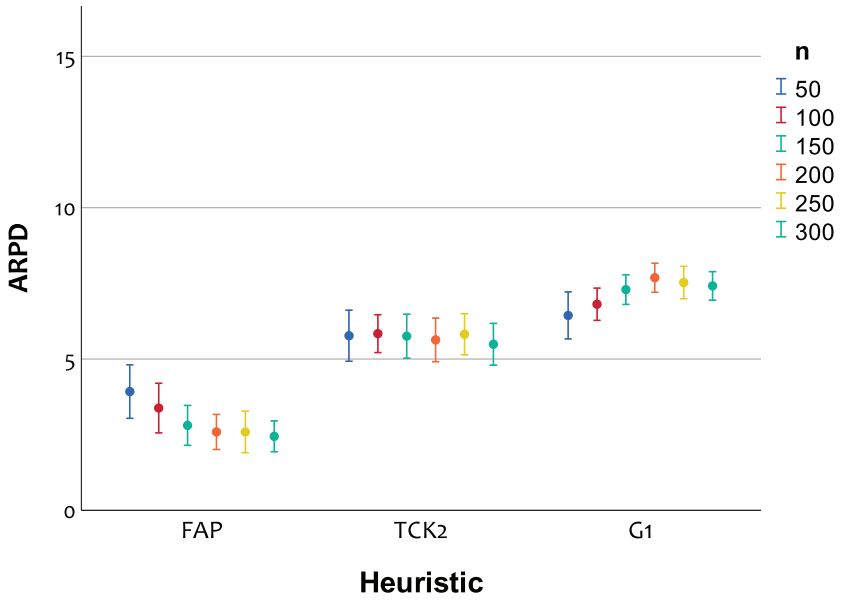

Figure 6.6: 95% Confidence Intervals for $ARPD$ of the best *SA* heuristics per levels of n in $\mathscr{B}_1$.

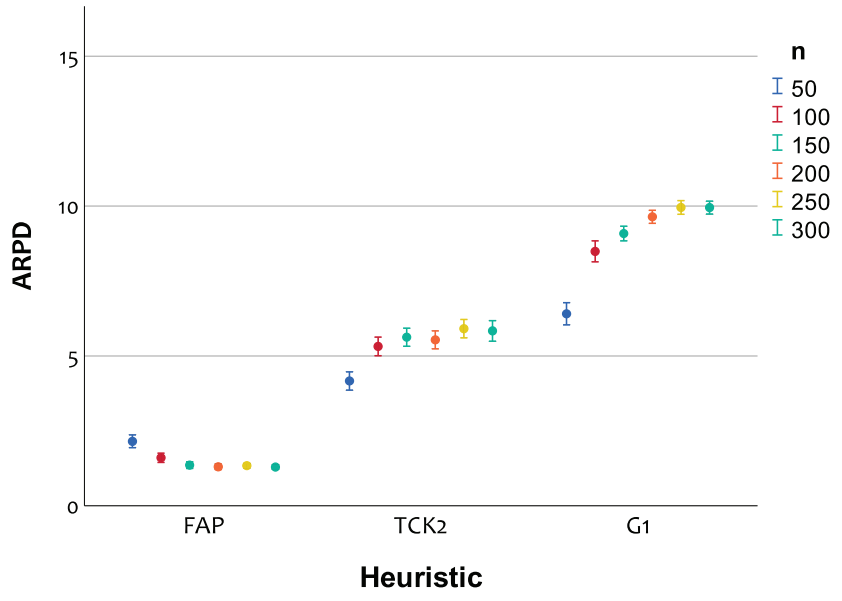

Figure 6.7: 95% Confidence Intervals for $ARPD$ of the best *SA* heuristics per levels of n in $\mathscr{B}_2$.

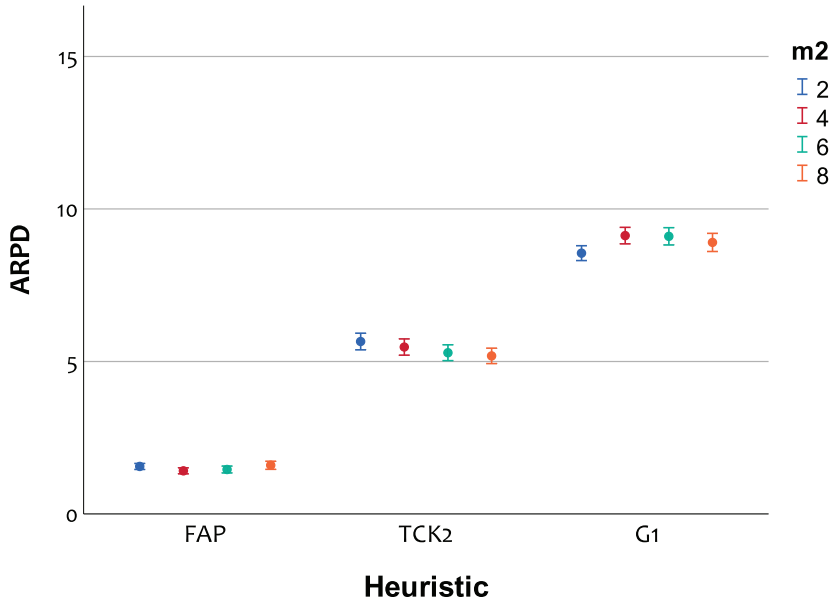

Figure 6.8: 95% Confidence Intervals for $ARPD$ of the best *SA* heuristics per levels of m_2 in $\mathscr{B}_2$.

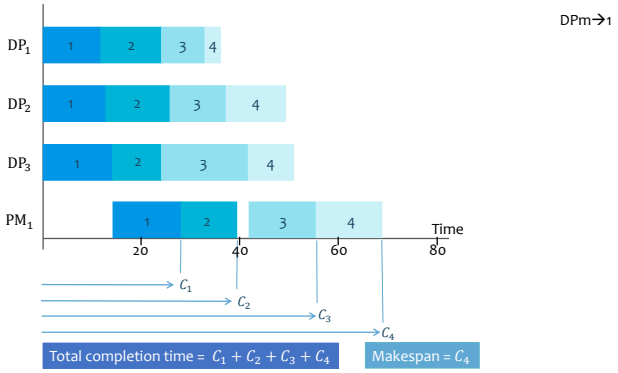

Figure 6.9: 95% Confidence Intervals for $ARPD$ of the best *MA* heuristics per levels of n in $\mathscr{B}_1$.

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 $p \leq /(k-i+1)$

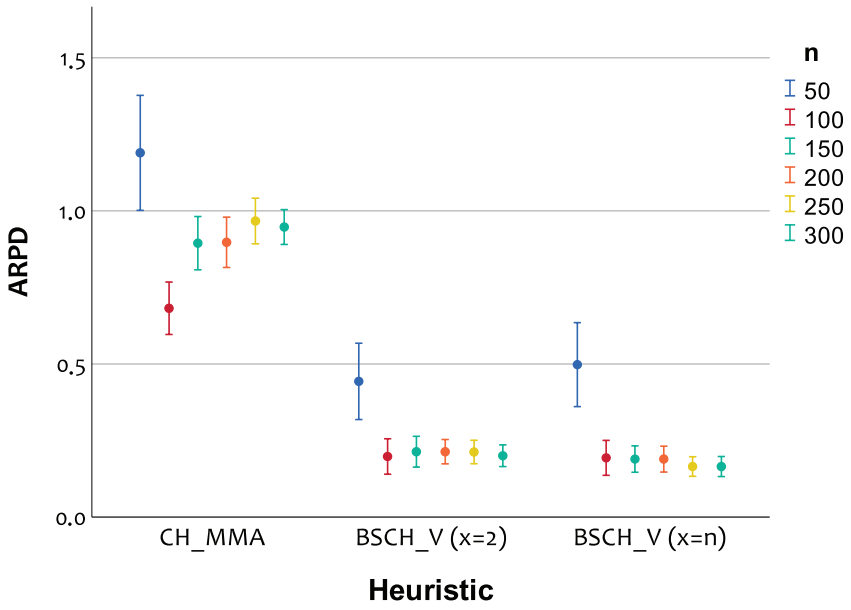

Figure 6.10: 95% Confidence Intervals for $ARPD$ of the best *MA* heuristics per levels of n in $\mathscr{B}_2$.

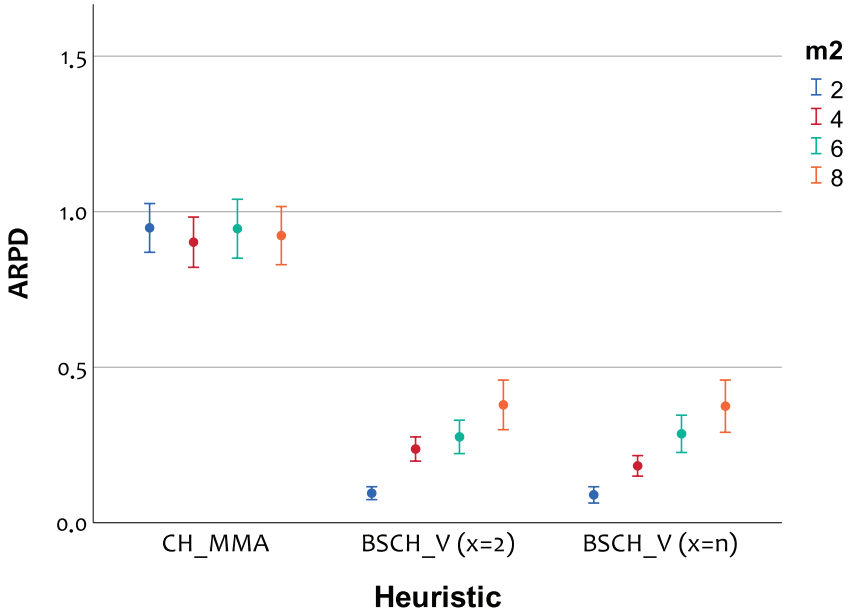

Figure 6.11: 95% Confidence Intervals for $ARPD$ of the best *MA* heuristics per levels of m_2 in $\mathscr{B}_2$.

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 6.12. It can be checked that, for both *SA* and *MA*, all the hypotheses are rejected, indicating that there are statistically significant differences between the considered heuristics, and validating the composition of the Pareto set.

6.5.5 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 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/.

6.6 Conclusions of the chapter

In this chapter, we have addressed the 2-stage multi-machine assembly scheduling problem with the objective of minimising the total completion time. We have presented two constructive heuristics (see GO3). The first algorithm, CH_{MA}, constructs a sequence by iteratively adding a job at the end of a partial sequence. The job is selected according to a problem-specific indicator that takes into account the idle time of the assembly machines in the second stage and the contribution of the job to the total completion time. Due to the good performance of this heuristic, the indicator has been embedded into a beam-search-based constructive heuristic, labelled $BSCH_{MA}$, which constructs several sequences at the same time, compares them, and selects the best x ones. Thereby, this second heuristic proposed combines the diversification of population-based algorithms and the speed of the constructive heuristic. Furthermore, we have implemented different variants of the $BSCH$, whose main difference is the

Table 6.11: Performance of heuristics on benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$.

		$\mathscr{B}_1$									$\mathscr{B}_2$								
		n						Average			n						Average		
Problem	Heuristic	50	100	150	200	250	300	$ARPD$	$ARPT'$	$ACPU$	50	100	150	200	250	300	$ARPD$	$ARPT'$	$ACPU$
SA	$TCK1$	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
	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
	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
	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
CO	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
	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**
	SFT_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
	$SFT_{k_{OPT}}$	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
MA	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
	$\boldsymbol{CH_{MA}}$	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
	$BSCH_{V(x=10)}$	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**
	$\boldsymbol{BSCH_{MA(x=n)}}$	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
	$BSCH_{MA(x=n+n/2)}$	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

		$\mathscr{B}_1$			
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_{MA}=$G1$	0.000	R	0.0167	R
H_3^1	$BSCH_{MA(x=n)}$ =$BSCH_{V(x=n/10)}$	0.000	R	0.0250	R
H_4^1	$BSCH_{MA(x=n+n/2)}$ = $BSCH_{V(x=15)}$	0.000	R	0.0500	R
		$\mathscr{B}_2$			
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_{MA}=ECT	0.000	R	0.0250	R
H_5^2	$BSCH_{V(x=2)}$=$BSCH_{MA(x=n)}$	0.027	R	0.0500	R

Table 6.12: Mann-Whitney's procedure. (R indicates that the hypothesis can be rejected).

way in which the beam width (x) is modified in each iteration.

Using testbed $\mathscr{B}_0$, the extensive computational experiments carried out show that the best $ARPD$ are found by variants $BSCH_V$ ($x \in \{2, n/10, 5, 10, 15, n\}$) and $BSCH_{MA}$ ($x \in \{n, n + n/2\}$). These variants have been compared with the CH_{MA} heuristic and with 18 existing heuristics for the problem under consideration and heuristics adapted from related scheduling problems. The results show that the proposed heuristics outperform the existing ones (see GO4).

Furthermore, all heuristics designed to solve variants *SA* and *MA* and related problems *CO*, *SM* and *PM* have been tested and compared in the new benchmarks $\mathscr{B}_1$ and $\mathscr{B}_2$. On the one hand, the results obtained show that most of the heuristics designed to solve the *SA* variant perform slightly better, on average, to solve the variant *MA* than the variant *SA*. On the other hand, the beam search-based constructive heuristics designed in 6.3 yield 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 of the variants considered. For *SA*, the group of the most efficient heuristics is formed by the dispatching rules $STPT/SMPT$ and $S3$, and the heuristics $TCK2$, CH_{MA}, $BSCH_{V(x=2)}$ and $BSCH_{MA(x=n)}$; and for *MA*, by the dispatching rules $S1$ and $STPT/SMPT$, and the heuristics CH_{MA} 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 Section 6.5.2, showcasing the importance of the set of instances selected in a state-of-the-art study.

Chapter 7

Heuristic algorithms for the 2-ASP-pm

7.1 Introduction

After reviewing the literature related to the 2-ASP-pm in Chapter 4, to the best of our knowledge, the $DPm \rightarrow 1|nr - pm|C_{\max}$ problem has never been addressed in the literature so far. To cover this opportunity, the contribution of this chapter, also shown in Figure 7.1, can be stated as follows:

- In Section 7.2, we explain the different simple bin-packing policies used in the construction of a schedule due to the relation of the problem to the bin-packing problem.
- In Section 7.3, the adapted methods are explained: in Section 7.3.1, all the dispatching rules adapted from the literature are detailed, while in Section 7.3.2 the heuristics adapted from the literature are presented.
- In Section 7.4, we design two constructive heuristics to efficiently solve the problem under study (see GO5). In this regard, it is well-known that constructive heuristics need no more than a few minutes to obtain good results (Pan et al., 2019). Furthermore, these methods can provide good initial solutions for metaheuristics (Framinan et al., 2014). For both reasons, following the idea of other papers considering periodic maintenance (Perez-Gonzalez and Framinan,

2018; Perez-Gonzalez et al., 2020), we propose two constructive heuristics as a first step to analyse their performance on the $DPm \rightarrow 1|nr - pm|C_{\max}$ problem. The first constructive heuristic uses knowledge of the problem domain to iteratively construct the solutions, while the second one extends this procedure by applying a partial local search mechanism at each iteration.

- In Section 7.5, we propose two composite heuristics by designing a novel local search mechanism. Note that, as commented before, constructive heuristics can often provide a reasonably good solution very quickly. Then, if improved solution quality is needed and time is allowed, then one way is to apply local search algorithms to the solution provided by a constructive heuristic(Liu and Reeves, 2001).
- Finally, in Section 7.7, two computational evaluations are carried out: on the one hand, 84 dispatching rules adapted from the literature are compared, and, on the other hand, the proposed heuristics are compared with seven approximate methods from the literature.

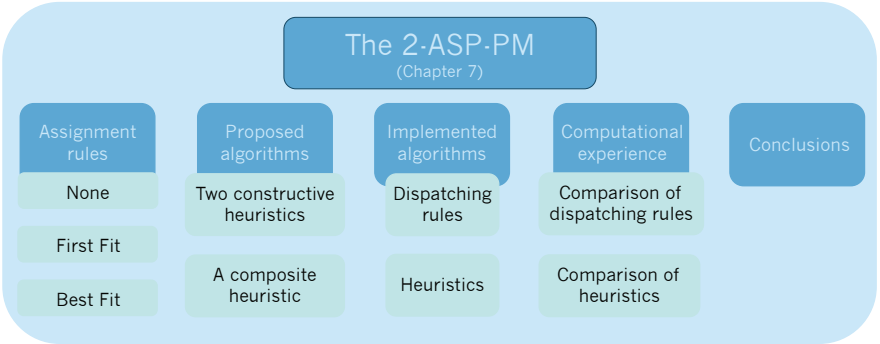

Figure 7.1: Structure of Chapter 7.

7.2 Assignment rules: Bin packing policies

In the problem under study, given a sequence, the construction of the schedule implies the assignment of each operation to a bin on each machine. The decision is not only to determine the job sequence, but also to assign such operations to the bins. Therefore,

a bin-packing policy is needed to generate a feasible solution of the problem at hand. In fact, in the process, the operation should *fit* in the bin, and the bin should be *feasible*. These concepts are formally defined in the following:

Definition 7.2.1 (Operation fits in bin z at the first stage) *The operation of the job j in a given sequence fits in a given bin z of a machine $i \in \{1, \ldots, m\}$ if $slack(z, i) \geq p_{ij}$. Let denote z as a feasible bin.*

Definition 7.2.2 (Operation fits in bin z at the second stage) *The operation of the job j in a given sequence fits in a given bin z of machine $m+1$ if* $\max\{C1_j, CT_{z,m+1}\} + at_j \leq z \cdot T$. *In this case, let denote z as a feasible bin.*

To evaluate a given sequence, the following different bin-packing policies can be applied in the construction of the schedule (Perez-Gonzalez and Framinan, 2018):

- Not applying any policy ($NONE$ or N): Given the sequence Π, the operation of job j in machine i is assigned to the same bin where the previous job was assigned, if the bin is feasible, or in the following bin in the opposite case.
- First Fit (FF) bin packing policy: Given the sequence Π, the operation of job j in machine i is assigned to the first feasible bin.
- Best Fit (BF) bin packing policy: Given the sequence Π, the operation of job j in machine i is assigned to the feasible bin where the operation fits with the minimum slack.

Depending on the bin packing policy selected to assign the operations to the bins, the schedule obtained could potentially be different, which consequently implies a different value of the objective function denoted as $C_{\max}^{NONE}$, $C_{\max}^{FF}$ and $C_{\max}^{BF}$, respectively.

For the sake of clarity, a numerical example involving six jobs to be scheduled in a 2-stage assembly system with $m = 3$ at the first stage is reported in the following. The processing times of the jobs are shown in Table 7.1 and $T = 25$ is considered. Given the sequence $\Pi = (1, 2, 3, 4, 5, 6)$, the different bin packing policies are applied, thus obtaining the Gantt charts shown in Figures 7.2, 7.3 and 7.4. If no bin packing policy is applied, as shown in Figure 7.2, when a job is too long to be included into the current bin, it is assigned to the next bin maintaining the order of jobs given by the

j	p_{1j}	p_{2j}	p_{3j}	at_j
1	12	13	14	14
2	16	13	15	11
3	5	11	13	13
4	3	12	9	13
5	16	10	17	7
6	11	10	3	5

Table 7.1: Processing times of the jobs in both stages.

sequence Π. In Figure 7.3, policy FF is applied and each job is assigned to the first feasible bin. Finally, in Figure 7.4, policy BF is applied and each job is assigned to the feasible bin with minimum slack. As commented before, by applying the policies FF and BF, jobs can be sequenced in each machine following a different order. Notably, by applying FF and BF, the obtained solutions entail a smaller number of bins than the one adopting the NONE policy. As a result, the machine workload (idle time) is higher (lower) than in the schedule pertaining to the NONE policy.

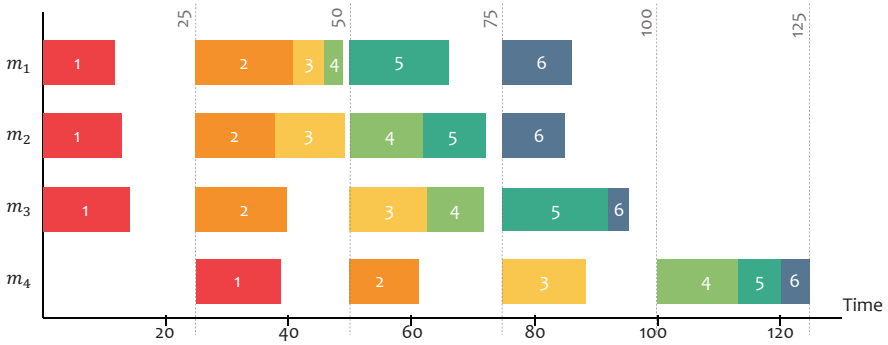

Figure 7.2: Gantt chart of the bin packing policy NONE.

7.3 Implemented heuristics

In this section, we deal with the methods that have been adapted and implemented from the related literature. For the sake of brevity, we refer the readers to their original papers for a full understanding of these algorithms.

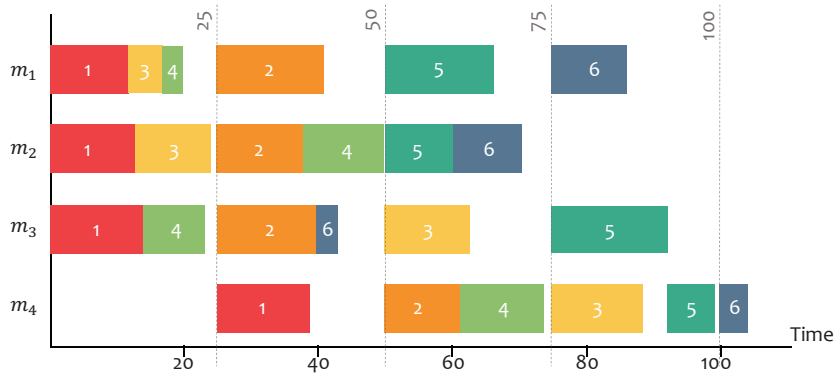

Figure 7.3: Gantt chart of the bin packing policy FF.

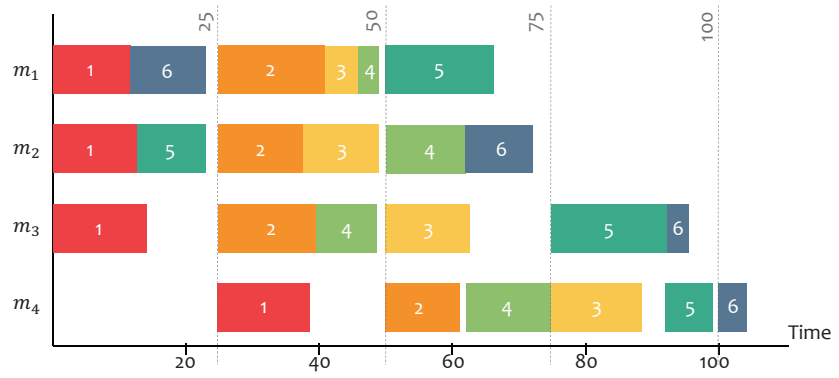

Figure 7.4: Gantt chart of the bin packing policy BF.

The existing methods can be classified in two groups according to the literature: dispatching rules and heuristics. They are explained in the following subsections.

7.3.1 Dispatching rules

Firstly, we adapt the dispatching rules proposed to solve the related problems. These methods consist of two phases: in the first phase, jobs are sorted according to a certain sorting criterion depending on the processing time of the jobs, and then, in the second phase, a specific bin packing policy is applied. In this section, we also consider in this group the methods from the literature for the $DP2 \rightarrow 1||C_{\max}$, $DPm \rightarrow 1|BS|C_{\max}$ and $DPm \rightarrow 1|r_j|C_{\max}$ problems, that apply Johnson's algorithm since they are based on the combination of two dispatching rules. In all the adaptations, each dispatching

rule is combined with FF, BF, and NONE bin policies to analyse the effect of the bin packing policies on the dispatching rules.

In Table 7.2, 7.3 and 7.4 all the adapted dispatching rules are summarised. The first two columns indicate the problem and the article from which each dispatching rule is adapted. The third column means the name[1] used for denoting each rule in this chapter. Then, the fourth column represents the index computed to sort the jobs or, if Johnson's rule is applied, the first index needed to apply this rule. The fifth column indicates the order in which jobs are sorted or, if Johnson's rule is applied, the second index needed to apply this rule. Finally, the last column means the bin packing policy applied in each case. Note that, all the dispatching rules from the $1|nr-pm|C_{\max}$ problem are adapted by computing ind_j (the index computed in the proposed methods in Section 7.4, see Equation 7.1). From the $DP2 \rightarrow 1||C_{\max}$ problem, m machines in the first stage are considered to adapt the different methods. Regarding the dispatching rules from the $DPm \rightarrow 1|r_j|C_{\max}$ problem, the methods shown in Table 7.2, 7.3 and 7.4 are adapted by eliminating the release dates from the original indexes. Finally, from the $DPm \rightarrow F2||C_{\max}$ problem, the rules are adapted by considering one machine in the second stage instead of a flowshop. In total, we obtain 84 dispatching rules.

7.3.2 Heuristics

Secondly, we adapt the existing heuristics in the literature to solve the related problems. These methods are summarised in Table 7.5, where it is also indicated their complexity. Their adaptations are briefly explained: from the $1|nr-pm|C_{\max}$ problem, the method PGF (Perez-Gonzalez and Framinan, 2018) is adapted by computing ind_j (see Equation 7.1 in Section 7.4) and applying the LPT rule to generate the initial solution. Then, one of the aforementioned policies (FF and BF) is applied to compute the objective function. From the $Fm|prmu, nr-pm|C_{\max}$ problem, the method $PFF1$ (Perez-Gonzalez et al., 2020) is adapted by generating the initial sequence by sorting the jobs in non-descending order of the index $X_j = C1_j + at_j + 2 \cdot IT_j$. Then, the algorithm is applied considering the problem as a flowshop with two stages. The adaptation of $PFF2$ and $PFF3$ implies the same heuristic since $PFF3$ is a generalization of $PFF2$. Finally, the authors propose two iterative heuristics by repeating

[1]Note that in the case of Komaki and Kayvanfar (2015), the original notation is used to make easier its identification in the original paper.

Problem	Author	Label	Order1	Order2	Bin packing policy
$1\|nr-pm\|C_{\max}$	[2]	LPT_{FF}	ind_j	LPT	FF
	[3]	LPT_{BF}			BF
	[4]	RND_{FF}		Random	FF
		RND_{BF}			BF
		SPT_{FF}		SPT	FF
		SPT_{BF}			BF
		$VSHARP_{FF}$		V-Sharp	FF
		$VSHARP_{BF}$			BF
		$ASHARP_{FF}$		A-Sharp	FF
		$ASHARP_{BF}$			BF
		$HILO_{FF}$		HILO	FF
		$HILO_{BF}$			BF
	[5]	$LOHI_{FF}$		LOHI	FF
		$LOHI_{BF}$			BF
$DP2 \rightarrow 1\|\|C_{\max}$	[6]	$LCL1_N$	$\max_i p_{ij}$	at_j	NONE
		$LCL1_{FF}$			FF
		$LCL1_{BF}$			BF
		$LCL2_N$	p_{i*j}	at_j	NONE
		$LCL2_{FF}$			FF
		$LCL2_{BF}$			BF
		$LCL3_N$	$\sum p_{ij}/at_j$	at_j	NONE
		$LCL3_{FF}$			FF
		$LCL3_{BF}$			BF
$DPm \rightarrow 1\|BS\|C_{\max}$	[7]	LCC_N	$\sum p_{ij}/m$	at_j	NONE
		LCC_{FF}			FF
		LCC_{BF}			BF
$DPm \rightarrow 1\|r_j\|C_{\max}$	[8]	$I4_N$	$\sum p_{ij}/m$	non-decreasing	NONE
		$I4_{FF}$			FF
		$I4_{BF}$			BF
		$I8_N$	p_{i*j}		NONE
		$I8_{FF}$			FF
		$I8_{BF}$			BF

[2] Ji et al. (2007).
[3] Hsu et al. (2010).
[4] Low et al. (2010b).
[5] Perez-Gonzalez and Framinan (2018).
[6] Lee et al. (1993).
[7] Lin et al. (2006).
[8] Komaki and Kayvanfar (2015).

Table 7.2: Classification of the dispatching rules adapted from the related problems (I).

Problem	Author	Label	Order1	Order2	Bin packing policy
$DPm \rightarrow 1\|r_j\|C_{\max}$	[8]	$I10_N$	$\max_i p_{ij} + at_j$	non-decreasing	NONE
		$I10_{FF}$			FF
		$I10_{BF}$			BF
		$I11_N$	$\sum p_{ij}/m + at_j$	non-increasing	NONE
		$I11_{FF}$			FF
		$I11_{BF}$			BF
		$I12_N$	$p_{i*j} + at_j$		NONE
		$I12_{FF}$			FF
		$I12_{BF}$			BF
		$I27_N$	$\max_i p_{ij}$	$\max_i p_{ij} + at_j$	NONE
		$I27_{FF}$			FF
		$I27_{BF}$			BF
		$I28_N$	$\sum p_{ij}/m$	$\max_i p_{ij} + at_j$	NONE
		$I28_{FF}$			FF
		$I28_{BF}$			BF
		$I32_N$	p_{i*j}	$\max_i p_{ij} + at_j$	NONE
		$I32_{FF}$			FF
		$I32_{BF}$			BF
		$I35_N$	$\max_i p_{ij}$	$\sum p_{ij}/m + at_j$	NONE
		$I35_{FF}$			FF
		$I35_{BF}$			BF
		$I36_N$	$\sum p_{ij}/m$	$\sum p_{ij}/m + at_j$	NONE
		$I36_{FF}$			FF
		$I36_{BF}$			BF
		$I40_N$	p_{i*j}	$\sum p_{ij}/m + at_j$	NONE
		$I40_{FF}$			FF
		$I40_{BF}$			BF
		$I43_N$	$\max_i p_{ij}$	$p_{i*j} + at_j$	NONE
		$I43_{FF}$			FF
		$I43_{BF}$			BF
		$I46_N$	$\sum p_{ij}/m$	$p_{i*j} + at_j$	NONE
		$I46_{FF}$			FF
		$I46_{BF}$			BF
		$I48_N$	p_{i*j}	$p_{i*j} + at_j$	NONE
		$I48_{BF}$			BF

[8] Ji et al. (2007).

Table 7.3: Classification of the dispatching rules adapted from the related problems (II).

Problem	Author	Label	Order1	Order2	Bin packing policy
$DPm \rightarrow F2\|\|C_{\max}$	[9]	$KTKB1_N$	$\max_i p_{ij}$	non-decreasing	NONE
		$KTKB1_{FF}$			FF
		$KTKB1_{BF}$			BF
		$KTKB3_N$	at_j		NONE
		$KTKB3_{FF}$			FF
		$KTKB3_{BF}$			BF
		$KTKB4_N$	$\max_i p_{ij} + at_j$		NONE
		$KTKB4_{FF}$			FF
		$KTKB4_{BF}$			BF
$DPm \rightarrow 1\|nr - pm\|C_{\max}$	First time imple-mented*	RND_N		Random	NONE
		SPT_{NONE}		SPT	
		LPT_N		LPT	
		$VSHARP_N$		V-Sharp	
		$ASHARP_N$		A-Sharp	
		$HILO_N$		HILO	
		$LOHI_N$		LOHI	

[9] Hsu et al. (2010).
[*] Only for complementing the existing methods for $1|nr - pm|C_{\max}$.

Table 7.4: Classification of the dispatching rules adapted from the related problems (III).

a NEH-based mechanism, named IT_{PFF1} and IT_{PFF2}. As for the $DP2 \rightarrow 1||C_{\max}$ problem, algorithms from Sun et al. (2003), named $SMN13$ and $SMN14$, are adjusted by considering m machines in the first stage and are combined with NONE, FF and BF policies. Finally, from the $DPm \rightarrow 1||C_{\max}$ problem, the method AA (Allahverdi and Al-Anzi, 2006) can be easily adapted and combined with NONE, FF and BF policies. In total, we obtain 15 heuristics from the state-of-the-art.

Furthermore, as the computational effort required by $CH1_{NLS}$ and $CH2_{NLS}$ can be considered quite higher than that by the other heuristics adapted for the comparison (see Section 7.6.4), we also adapt the imperialist competitive algorithm (ICA) proposed by Seidgar et al. (2014) for the $DPm \rightarrow 1||F_l(C_{\max}, \sum C_j)$ problem. This metaheuristic is a population-based algorithm, and the population consists of some countries (sequences), which are classified in imperialists and colonies. Both types of countries compete to be the most powerful country, i.e., the sequence with the best value of the objective function. The stopping criterion considered is the computational time required by the different versions of $CH1_{NLS}$ and $CH2_{NLS}$.

7.4 Constructive heuristics

In this section, we propose two constructive heuristics and a local search mechanism for the $DPm \rightarrow 1|nr - pm|C_{\max}$ problem. The first proposal, denoted as $CH1$, constructs a sequence by iteratively appending jobs at the end of a sequence. Then, the second constructive heuristic, *CH2* in the following, is obtained by applying a partial local search mechanism enabled at each iteration of *CH1* (Section 7.4.2) and, finally, we propose a novel local search mechanism, denoted as NLS, to be implemented after constructing a complete sequence by *CH1* and *CH2*, hereinafter denoted as $CH1_{NLS}$ and $CH2_{NLS}$, respectively (Section 7.5).

7.4.1 Constructive Heuristic 1

Following the ideas of the methods proposed by Liu and Reeves (2001) and applied to other assembly problems (see e.g., Framinan and Perez-Gonzalez, 2017b), the proposed heuristic *CH1* iteratively constructs a sequence by selecting the most suitable job among the unscheduled jobs and appending it to the end of the partial sequence. The pseudocode of $CH1$ is provided in Algorithm 3. *CH1* starts with a set $\mathscr{W}$ of all (unscheduled) jobs and an empty sequence Π. For each iteration ($\lambda \in \{1, \ldots, n\}$),

Problem	Author	Label	Bin packing policy	Complexity
$1\|nr-pm\|C_{\max}$	[10]	PGF_N	NONE	n^2m
		PGF_{FF}	FF	n^2mZ
		PGF_{BF}	BF	n^2mZ
$Fm\|prmu, nr-pm\|C_{\max}$	[11]	$PFF1$	-	n^3mZ
		$PFF2$	-	n^3mZ
		IT_{PFF1}	-	Kn^3mZ*
		IT_{PFF2}	-	Kn^3mZ*
$DP2 \rightarrow 1\|\|C_{\max}$	[12]	$SMN13_N$	NONE	n^2+nm
		$SMN13_{FF}$	FF	n^2+nmZ
		$SMN13_{BF}$	BF	n^2+nmZ
		$SMN14_N$	NONE	n^2+nm
		$SMN14_{FF}$	FF	n^2+nmZ
		$SMN14_{BF}$	BF	n^2+nmZ
$DPm \rightarrow 1\|\|C_{\max}$	[13]	AA_N	NONE	nm
		AA_{FF}	FF	nmZ
		AA_{BF}	BF	nmZ

[10] Perez-Gonzalez and Framinan (2018).
[11] Perez-Gonzalez et al. (2020).
* K is the number of times that the NEH procedure is executed. It is equal to $\lceil \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} + at_j)/T \rceil$.
[12] Sun et al. (2003).
[13] Allahverdi and Al-Anzi (2006).

Table 7.5: Classification of the heuristics adapted from the related problems.

each unscheduled job $\omega_l \in \mathscr{W}$ $(l = 1, \ldots, n - \lambda + 1)$ is analysed as candidate to be added to the last position of Π. The suitability of each job is measured by computing the index ψ_{ω_l} and the job with the lowest value is selected to be scheduled at the end of Π. There are three main characteristics of the problem to be considered in order to fully define this index:

1. The completion time of job ω_l. It depends on its completion time in the first stage (i.e., $C1_1 = max_i\ p_{i\omega_l}$) and the processing time in the second stage, at_{ω_l}.

2. The idle time between stages before processing job ω_l. We denote with IT_{ω_l} the idle time in machine $m + 1$ caused by scheduling job ω_l at the end of the sequence, i.e., $IT_{\omega_l} = max\{C1^{BF}_{\omega_l} - CT_{z,m+1}, 0\}$, where $C1^{BF}_{\omega_l}$ is the completion time of ω_l in the first stage by applying BF bin-packing policy, and $CT_{z,m+1}$ is the completion time of the last job scheduled before ω_l in the bin z in machine $m + 1$ according to the notation from Section 2.4.

3. The waiting time incurred when a job has to wait for being processed in the second stage, i.e., when $IT_{\omega_l} = 0$. Then, the difference between the completion time of the job in the second stage and in the first stage is computed as $C^{BF}_{\omega_l} - C1^{BF}_{\omega_l}$, where $C^{BF}_{\omega_l}$ is the completion time of job ω_l, using BF as bin-packing policy.

By considering these three aspects, we will ensure that the jobs to be first sequenced are those with lower values of idle/waiting time and completion time. Additionally, we also take into account the availability period T. Therefore, the index ψ_{ω_l}, which estimates the suitability of appending a candidate job ω_l at the end of Π, is computed as follows:

$$\psi_{\omega_l} = \begin{cases} \max_i p_{i\omega_l} + at_{\omega_l} - \frac{c_1 \cdot T}{c_3} \cdot IT_{\omega_l} & \text{if } IT_{\omega_l} > 0, \\ \max_i p_{i\omega_l} + at_{\omega_l} + \frac{c_2 \cdot T}{c_3} \cdot (C^{BF}_{\omega_l} - C1^{BF}_{\omega_l}) & \text{if } IT_{\omega_l} = 0, \end{cases}$$

where c_1, c_2 and c_3 are parameters of the algorithm that would be determined via a proper calibration analysis (see Section 7.6.1).

The complexity of this heuristic is $\mathcal{O}(n \cdot (n - \lambda) \cdot m \cdot Z) \sim \mathcal{O}(n^2 \cdot m \cdot Z)$, where b is the final number of bins used. It is clear that the main loop of the algorithm performs n iterations and, in each one, $n - \lambda$ jobs are considered in the partial sequence. In each evaluation, the maximum processing time in the first stage is computed, so all

machines m are taken into consideration. Besides, for each machine, all bins (Z) are also considered in order to determine the bin where best fits job ω_{ω_l}.

Algorithm 3 Pseudocode of the proposed heuristic $CH1$

procedure CH1
  All jobs are initially unscheduled
  $\Pi := \emptyset$;
  $\mathscr{W} := \{1, \ldots, n\}$;
  for $\lambda = 1$ to n **do**
    for $l = 1$ to $n - \lambda + 1$ **do**
      Compute the completion times, applying the bin packing policy *Best Fit* in both stages with ω_l as candidate.
      Compute the idle time induced if job ω_l is inserted at the end of the partial sequence:
      $IT_{\omega_l} = \max\left\{C1^{BF}_{\omega_l} - CT_{zi}, 0\right\}$
      Compute the index:
      if $IT_{\omega_l} > 0$ **then**
        $\psi_{\omega_l} = \max_i p_{i\omega_l} + at_{\omega_l} - c_1 \cdot T/c_3 \cdot IT_{\omega_l}$
      else
        $\psi_{\omega_l} = \max_i p_{i\omega_l} + at_{\omega_l} + c_2 \cdot T/c_3 \cdot (C^{BF}_{\omega_l} - C1^{BF}_{\omega_l})$
      end if
    end for
    $r := \arg\min_{1 \le l \le n-\lambda+1} \psi_l$;
    Append ω_r at the end of Π, i.e., $\Pi := (\pi_1, \ldots, \pi_{\lambda-1}, \omega_r)$;
    Extract ω_r from U;
  end for
  Evaluate the sequence Π: $C^{BF}_{\max}(\Pi) = max_j C^{BF}_j(\Pi)$
return $C^{BF}_{\max}(\Pi)$
end procedure

7.4.2 Constructive Heuristic 2

Similar to $CH1$, we design an additional constructive heuristic by adding an improvement procedure after appending the most suitable job in each iteration. More

specifically, the proposed method, labelled $CH2$, consists of two phases. First, we select a job to be added to the partial sequence according to the index ψ_l, and then, we apply an improvement procedure based on exploring the space of solutions of the partial sequence. The neighbourhood structure is based on interchanging the last added job with the rest of the jobs in the partial sequence, considering a relative improvement strategy, i.e., if the new solution obtained after an interchange improves the best solution in the current iteration, the latter solution is replaced and the local search keeps evaluating the space of solutions by interchanging the selected job with the remaining jobs. To reduce the computational effort of the improvement procedure, each partial schedule obtained by each interchange is evaluated in this case by the FF policy. Algorithm 4 describes the pseudocode of $CH2$. The complexity of this heuristic is $\mathcal{O}(n^3 \cdot m \cdot Z)$.

Let consider the example from Section 7.2. In Figure 7.5 is shown the methodology followed by applying the improvement procedure. Being the partial sequence $\Pi = (2, 1, 4)$ and job 4 the last job to be added, the first sequence evaluated by the FF bin packing policy is Π' =(2,1,4) with $bestsol := C^{FF}_{\max} = 88$. Then, job 4 is interchanged with job 2 obtaining the sequence Π' =(4,1,2) with $currsol := C^{FF}_{\max} = 86$. As $currsol < bestsol$, then $bestsol = currsol$ and $\Pi = \Pi'$. Job 4 is interchanged with job 1, obtaining the sequence Π' =(1,4,2) with $currsol := C^{FF}_{\max} = 89$. Therefore, Π =(4,1,2) and the next iteration is started.

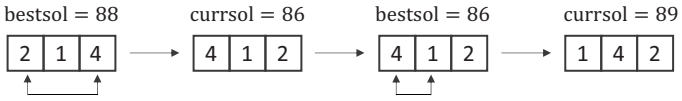

Figure 7.5: Example of the relative improvement strategy.

7.5 Composite heuristics

In this section, an additional local search mechanism is proposed. This mechanism, named *Novel Local Search* (NLS), is designed to be applied once we get a complete sequence by the constructive heuristics $CH1$ and $CH2$, being the resultant heuristics named $CH1_{NLS}$ and $CH2_{NLS}$. NLS consists of interchanging two sets of jobs ($\mathscr{A}$ and $\mathscr{B}$), both with the same length γ in the complete sequence Π provided by the corresponding constructive heuristic. The procedure followed by this mechanism is

Algorithm 4 Pseudo-code of the proposed heuristic $CH2$

procedure CH2
    All jobs are initially unscheduled
    $\Pi := \emptyset$;
    $\mathscr{W} := \{1, \ldots, n\}$;
    for $\lambda = 1$ to n **do**
        for $l = 1$ to $n - \lambda + 1$ **do**
            Compute the completion times, applying the bin packing policy *Best Fit* in both stages with ω_l as candidate.
            Compute the idle time induced if job ω_l is inserted at the end of the partial sequence:
            $IT_{\omega_l} = \max\left\{C1^{BF}_{\omega_l} - CT_{ki}, 0\right\}$
            Compute the index ψ_{ω_l}:
            if $IT_{\omega_l} > 0$ **then**
                $\psi_{\omega_l} = \max_i p_{i\omega_l} + at_{\omega_l} - c_1 \cdot T/c_3 \cdot IT_{\omega_l}$
            else
                $\psi_{\omega_l} = \max_i p_{i\omega_l} + at_{\omega_l} + c_2 \cdot T/c_3 \cdot (C^{BF}_{\omega_l} - C1^{BF}_{\omega_l})$
            end if
        end for
        $r := \arg\min_{1 \le l \le n-\lambda+1} \psi_l$;
        Append ω_r at the end of Π, i.e., $\Pi := (\pi_1, \ldots, \pi_{\lambda-1}, \omega_r)$;
        Extract ω_r from U;
        if $\lambda \ge 2$ **then**
            Evaluate Π: $C^{FF}_{\max}(\Pi) = max_j C^{FF}_j(\Pi)$
            Compute the completion times, applying the bin-packing policy First Fit. Do $bestsol := C^{FF}_{\max}(\Pi)$
            for $s = 1$ to $s = \lambda - 1$ **do**
                Let Π' be the neighbour obtained interchanging ω_r with job in position s.
                Evaluate Π': $C^{FF}_{\max}(\Pi') = max_j C^{FF}_j(\Pi')$. Do $currsol := C^{FF}_{\max}(\Pi')$
                if $currsol < bestsol$ **then**
                    $bestsol = currsol$. $\Pi = \Pi'$
                end if
            end for
        end if
    end for
    return $C^{FF}_{\max}(\Pi)$
end procedure

as follows:

- The selection of the jobs in the first set, $\mathscr{A}$, is based on the variable ind_j, according to Equation 7.1. This index is defined for each job j (with $j \in \{1, \ldots, n\}$) as the sum of the maximum completion time in the first stage and the processing time in the second stage:

$$ind_j = \max_i p_{ij} + at_j \tag{7.1}$$

  Let $\mathscr{O} = (\mathscr{O}_1, \ldots, \mathscr{O}_n)$ be the order obtained after sorting the jobs in non-decreasing order according to such index, then we select γ consecutive jobs of $\mathscr{O}$ to be included in $\mathscr{A}$. Three options for positioning the selected jobs, denoted as $\mathscr{P}_{\mathscr{A}}$, are considered:

  - $\mathscr{P}_{\mathscr{A}}$ = Beginning. The selected jobs are at the beginning of $\mathscr{O}$: $\mathscr{A} = (\mathscr{O}_1, \ldots, \mathscr{O}_\gamma)$.
  - $\mathscr{P}_{\mathscr{A}}$ = Middle. The selected jobs are at the middle of $\mathscr{O}$: $\mathscr{A} = (\mathscr{O}_j, \ldots, \mathscr{O}_{j+\gamma})$ with $j = \lceil \gamma/2 \rceil$.
  - $\mathscr{P}_{\mathscr{A}}$ = End. The selected jobs are at the end of $\mathscr{O}$: $\mathscr{A} = (\mathscr{O}_{n-\gamma}, \ldots, \mathscr{O}_n)$.

- The jobs in the second set, $\mathscr{B}$, are selected directly from Π, being γ consecutive jobs in this sequence. In the same way, the three previous options, $\mathscr{P}_{\mathscr{B}}$, are considered, selecting those jobs at the beginning, the middle, or the end of Π.

Note that to do the interchange process, the jobs indicated by $\mathscr{A}$ are selected one by one in Π, and they are interchanged with all jobs indicated by $\mathscr{B}$. The procedure depends on three parameters: $\gamma \in \{\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}, n\}$ and $\mathscr{P}_{\mathscr{A}}, \mathscr{P}_{\mathscr{B}} \in \{Beginning, Middle, End\}$. Each neighbour is evaluated by the FF policy and the relative improvement strategy used in the Improvement Procedure described in Algorithm 4. Algorithm 5 describes the pseudocode of this mechanism, whose complexity is $\mathscr{O}(n^3 \cdot m \cdot Z)$.

The procedure considering $\gamma = \frac{n}{2}$, $\mathscr{P}_{\mathscr{A}} = Beginning$ and $\mathscr{P}_{\mathscr{B}} = End$, is illustrated by an example (see Figure 7.6), where $\mathscr{O} = [1, 6, 4, 3, 5, 2]$ and $\Pi = [2, 1, 6, 4, 3, 5]$ has been obtained by $CH1$ or $CH2$, with a value of the objective function equals to 93. Then, $\mathscr{A} = [1, 6, 4]$ and $\mathscr{B} = [4, 3, 5]$. The process is as follows:

- Iteratively, each job is selected from Π given by $\mathscr{A}$ and interchanged with each job given by $\mathscr{B}$. Each neighbour is evaluated, and Π is updated in case of improvement. More specifically, the following steps are done:

Algorithm 5 Pseudo-code of the proposed local search mechanism NLS

procedure NLS(Π, γ, $\mathscr{P}_{\mathscr{A}}$, $\mathscr{P}_{\mathscr{B}}$)
  Let Π be a sequence given by heuristic $CH1$ or $CH2$
  $bestsol := C^{FF}_{\max}(\Pi)$
  Compute for each job ind_j according to Equation (7.1).
  Let $\mathscr{O} = (\mathscr{O}_{[1]}, \ldots, \mathscr{O}_{[n]})$ be the order obtained after sorting the jobs in non-decreasing order of ind_j.
  Two sets of jobs, $\mathscr{A}$ and $\mathscr{B}$, initially empty.
  $\{\mathscr{A}, \mathscr{B}\} := \emptyset$;
  Select γ consecutive jobs, depending on $\mathscr{P}_{\mathscr{A}}$, of $\mathscr{O}$ to be included in $\mathscr{A}$.
  Select γ consecutive jobs, depending on $\mathscr{P}_{\mathscr{B}}$, of Π to be included in $\mathscr{B}$.
  for each job $\mathscr{O}_x$ in $\mathscr{A}$ **do**
    Let Π' be the neighbour obtained interchanging $\mathscr{O}_x$ with each job given by $\mathscr{B}$.
    Evaluate Π': $C^{FF}_{\max}(\Pi') = max_j C^{FF}_j(\Pi')$. Do $currsol := C^{FF}_{\max}(\Pi')$
    if $currsol < bestsol$ **then**
      $bestsol = currsol$
      $\Pi = \Pi'$
    end if
  end for
end procedure

- 1^{st} iteration: The first job in $\mathscr{A}$ is 1. This job is located in the second position of Π, and it is interchanged with the first job indicated by $\mathscr{B}$, 4. The neighbour obtained is [2,4,6,1,3,5], with objective function equals to 96. Then, 1 is interchanged with the second job indicated by $\mathscr{B}$, 3. The neighbour obtained is [2,3,6,4,1,5], with objective function equals to 102. Finally, 1 is interchanged with the third job indicated by $\mathscr{B}$, 5. The neighbour obtained is [2,5,6,4,3,1], with the objective function equals to 91. As, in this case there is an improvement, Π is updated: $\Pi = [2, 5, 6, 4, 3, 1]$
- 2^{nd} iteration. The second job in $\mathscr{A}$ is 6, and it is interchanged with the first job indicated by $\mathscr{B}$, 4, then with the second job 3, and finally with the third job 5. In this iteration, there is not improvement.
- 3^{rd} iteration. The third job in $\mathscr{A}$ is 4. As it coincides with the first job in $\mathscr{B}$, in this iteration, there are only two neighbours, those obtained by the interchange of jobs 3 and 5. In this iteration, there is not improvement.

• After that, the procedure returns as the best solution, the new sequence $\Pi = [2, 5, 6, 4, 3, 1]$.

7.6 Computational experience

This section describes the procedure for evaluating the proposed algorithms and the adapted methods, detailed in Sections 7.4 and 7.3. Section 3.2 explains the testbeds and the performance indicators necessary to evaluate the MILP and compare the different algorithms, while Section 7.6.1 presents the design of experiments to properly set the values of the control parameters pertaining to $CH1$ and the NLS procedure as well. Section 7.6.2 analyses the efficiency of the MILP model, Section 7.6.3 outlines the results of a comparison analysis involving the aforementioned dispatching rules, and Section 7.6.4 allows assessing the difference in performance among existing and proposed heuristics in solving the scheduling problem at hand. This comparison has been carried out using the same language (C# using Visual Studio) and on an Intel Core i7-3770 PC with 3.4 GHz and 16 GB RAM, and using the same common functions and libraries. In order to obtain a better estimate of the CPU time required by each algorithm, and based on the computational experiments carried out by Ruiz

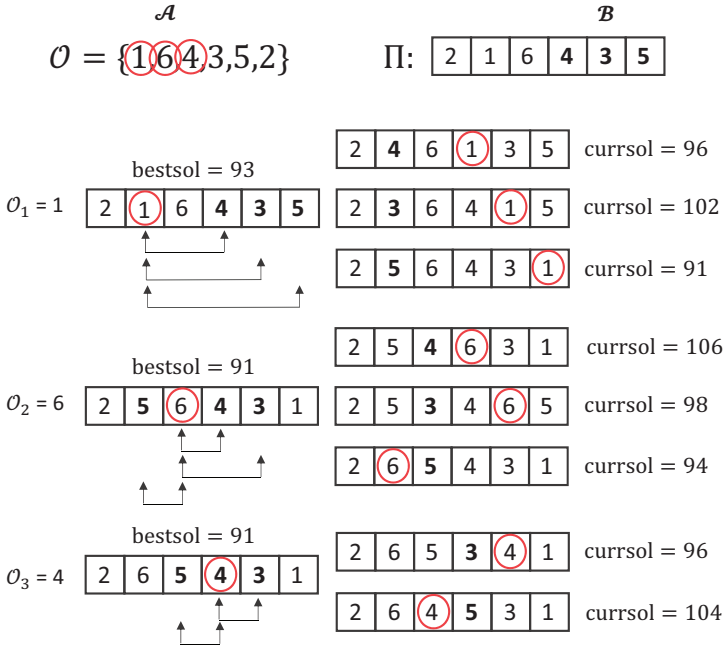

Figure 7.6: Example of the partial interchange local search mechanism.

and Stützle (2007), Pan and Ruiz (2013) and Fernandez-Viagas and Framinan (2014), a total of five replicates for each instance are carried out and the results are averaged.

7.6.1 Tuning control parameters

In this section, a factorial design of experiments is performed to find the best values of the parameters of the constructive heuristic $CH1$ and the novel local search mechanism based on interchange NLS presented in Section 7.4. After some preliminary tests, it has been identified that the best values for the different parameters are in the ranges detailed below. More specifically, the following values are tested:

- $CH1$ parameters: $c_1 \in \{1, 2, 5, 10\}$, $c_2 \in \{0, 0.5, 1\}$ and $c_3 \in \{1, 500\}$, since 500 is the highest level of parameter T (see Section 3.3). In total, there are 24 combinations.

- NLS parameters: $\gamma \in \{\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}, n\}$, and $\mathscr{P}_{\mathscr{A}}, \mathscr{P}_{\mathscr{B}} \in \{Beginning, Middle, End\}$. In this case, we are interested in determining the most favourable cases for the

parameters $\mathscr{P}_{\mathscr{A}}$ and $\mathscr{P}_{\mathscr{B}}$, in order to test $CH1_{NLS}$ and $CH2_{NLS}$ with all values of $\gamma \in \{\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}, n\}$. Therefore, in the tuning process, γ has been set to $\frac{n}{2}$.

To determine the best combination of parameters, the calibration benchmark $\mathscr{B}_{c4}$ presented in Section 3.3 is used. As noted before, these instances are different from those used in Section 7.6.3 and Section 7.6.4 to avoid overfitting these parameters. After proving that the normality and homoscedasticity assumptions are not fulfilled, two Kruskal-Wallis tests are performed to determine the best combinations of parameters:

- In order to determine the best combination for the parameters c_1, c_2 and c_3, the generated instances have been solved through the 24 versions of $CH1$ obtained by combining all the levels of the parameters. The response variable in the statistical analysis is $ARPD$. A non-parametric Kruskal-Wallis test has been carried out, providing significant differences between the levels of the parameters. See Appendix for the different values of $ARPD$. The best combination is obtained for $c_1 = 2$, $c_2 = 1$ and $c_3 = 500$.

- To determine the best combination of parameters $\mathscr{P}_{\mathscr{A}}$ and $\mathscr{P}_{\mathscr{B}}$, the generated instances have been solved by the nine versions of $CH1_{NLS}$ obtained combining the levels of both parameters. The parameters used for $CH1$ have been fixed to those obtained in the previous test. The response variable is again the $ARPD$. Again, significant differences between the levels of the parameters have been obtained, being the best combination $\mathscr{P}_{\mathscr{A}} = Beginning$, and $\mathscr{P}_{\mathscr{B}} = End$. The values of $ARPD$ for the different combinations are shown in Appendix.

7.6.2 Evaluation of the MILP model

In this section, we run the MILP model for the small size instances of benchmark $\mathscr{B}_3$. We set the time limit for the MILP model to 1800 seconds and report the optimal/best feasible solutions found. In Table 7.6, we summarize the computational results grouped by instance size (n and m) and level of parameter T, showing the average value of $ARPD$ and $ACPU$, and the percentage of optimal solutions obtained by the model for 10 instances. From the table, it is clear that as the level of T increases, the model finds more optimal solutions, since as the constraint is relaxed, it is easier to find optimal solutions by the model. It is remarkable the big difference in $ARPD$ between instances of size $n = 40$ and $m = 8$ for $T = 200$ ($ARPD = 138.36$) and

$T = 500$ ($ARPD = 0.36$). With respect to the behaviour of the model, it requires more computational time to obtain an optimal solution as the number of jobs and machines increases. Regarding the total average of percentage of optimal solutions, it is equal to 65%, so we can conclude that the model is effective to solve instances with the size considered in this testbed. However, the MILP is not efficient to solve bigger instances in reasonable computational time, and it becomes necessary to apply approximate methods.

7.6.3 Comparison of the dispatching rules

This section describes the numerical results achieved by the dispatching rules on the benchmark $\mathscr{B}_4$ generated as in Section 3.2. The $ARPD$ of the different methods (84 in total) are summarised in Table 7.7. Note that the $ACPU$ values are not analysed since all the methods are fast and their $ACPU$ are not significant (lower than 0.0001). The $ARPD$ values range from 30.63 ($LOHI_N$) to 2.86 (LCC_{FF}).

From these results, the following conclusions can be obtained:

- The worst performance is found by applying methods with the NONE bin packing policy, since there is a big difference among their $ARPD$ values and those from FF and BF policies. In the case of methods $KTKB3$, $KTKB4$, SPT, $VSHARP$ and $ASHARP$, the bin packing policies do not influence the results since all of them appear quite weak in solving the problem under investigation and a small difference emerges from a numerical viewpoint.

- The FF and BF policies yield a similar performance according to the $ARPD$, being the average $ARPD$ for all the dispatching rules, 8.04 for BF and 8.20 for FF.

- The minimum value of $ARPD = 2.86$ is achieved by LCC with the FF bin packing policy (LCC_{FF}). However, there are other dispatching rules whose $ARPD$ values make them quite comparable with each other, such as $LCL1_{BF}$ and $I27_{BF}$. To establish the statistical significance of these results, a non-parametric Mann-Whitney test is performed, assuming a 0.95 confidence level, i.e., $\alpha = 0.05$. The first hypothesis compares LCC_{FF} and $LCL1_{BF}$, obtaining a p-value $= 0.233$. Hence, there are not statistical evidence to reject it. The second hypothesis compares LCC_{FF} and $I27_{BF}$ and it is rejected since its p-value is equal to 0.014.

T	n	m	$ARPD$	$ACPU$	%OPT	T	n	m	$ARPD$	$ACPU$	%OPT
200	10	2	0.00	0.57	100	400	10	2	0.00	0.14	100
		4	0.00	0.68	100			4	0.00	0.16	100
		6	0.00	1.51	100			6	0.00	0.40	100
		8	0.00	1.93	100			8	0.00	0.71	100
	20	2	0.00	1307.62	30		20	2	0.00	4.00	100
		4	0.00	1642.86	10			4	0.00	11.50	100
		6	0.10	1800.12	0			6	0.00	10.52	100
		8	1.03	1653.79	10			8	0.00	34.08	100
	30	2	0.09	1599.00	20		30	2	0.00	27.37	100
		4	0.28	1713.78	10			4	0.07	592.11	80
		6	1.49	1800.05	0			6	0.00	936.66	60
		8	1.32	1800.09	0			8	0.18	875.50	60
	40	2	0.06	1800.03	0		40	2	0.00	155.80	100
		4	1.40	1800.07	0			4	0.30	1531.11	30
		6	3.88	1800.13	0			6	0.41	1798.52	10
		8	138.36	1800.15	0			8	0.93	1629.23	20
300	10	2	0.00	0.26	100	500	10	2	0.00	0.08	100
		4	0.00	0.27	100			4	0.00	0.20	100
		6	0.00	0.85	100			6	0.00	0.21	100
		8	0.00	0.91	100			8	0.00	0.43	100
	20	2	0.00	185.76	90		20	2	0.00	1.57	100
		4	0.00	366.63	90			4	0.00	2.50	100
		6	0.00	182.76	100			6	0.00	3.71	100
		8	0.00	443.66	90			8	0.00	8.89	100
	30	2	0.00	430.81	90		30	2	0.00	6.17	100
		4	0.14	1191.00	40			4	0.00	243.60	100
		6	0.35	1369.80	30			6	0.00	464.78	90
		8	0.41	1603.95	20			8	0.00	564.89	80
	40	2	0.03	900.96	70		40	2	0.00	27.67	100
		4	0.98	1800.07	0			4	0.06	922.54	60
		6	1.29	1800.09	0			6	0.15	1205.41	50
		8	2.09	1800.12	0			8	0.36	1342.03	30

Table 7.6: Values of $ARPD$, $ACPU$ and %OPT of the MILP.

DR	$ARPD$	DR	$ARPD$	DR	$ARPD$	DR	$ARPD$
$KTKB1_N$	22.81	$ASHARP_N$	27.85	$I4_N$	22.81	$I32_N$	23.38
$KTKB1_{FF}$	4.34	$ASHARP_{FF}$	24.01	$I4_{FF}$	4.34	$I32_{FF}$	4.37
$KTKB1_{BF}$	3.77	$ASHARP_{BF}$	23.91	$I4_{BF}$	3.84	$I32_{BF}$	3.94
$KTKB3_N$	30.05	$LOHI_N$	30.63	$I8_N$	23.32	$I35_N$	22.90
$KTKB3_{FF}$	29.33	$LOHI_{FF}$	9.95	$I8_{FF}$	4.96	$I35_{FF}$	3.28
$KTKB3_{BF}$	29.32	$LOHI_{BF}$	10.58	$I8_{BF}$	4.44	$I35_{BF}$	3.05
$KTKB4_N$	27.85	$HILO_N$	30.23	$I10_N$	23.61	$I36_N$	22.91
$KTKB4_{FF}$	24.01	$HILO_{FF}$	9.57	$I10_{FF}$	4.64	$I36_{FF}$	3.66
$KTKB4_{BF}$	23.91	$HILO_{BF}$	10.22	$I10_{BF}$	4.67	$I36_{BF}$	3.34
RND_N	23.25	$LCL1_N$	22.86	$I11_N$	23.43	$I40_N$	23.38
RND_{FF}	5.01	$LCL1_{FF}$	3.12	$I11_{FF}$	4.54	$I40_{FF}$	4.33
RND_{BF}	4.56	$LCL1_{BF}$	2.95	$I11_{BF}$	4.57	$I40_{BF}$	3.92
SPT_N	27.85	$LCL2_N$	23.37	$I12_N$	23.48	$I43_N$	22.89
SPT_{FF}	24.01	$LCL2_{FF}$	4.26	$I12_{FF}$	4.50	$I43_{FF}$	3.33
SPT_{BF}	23.91	$LCL2_{BF}$	3.89	$I12_{BF}$	4.58	$I43_{BF}$	3.06
LPT_N	23.61	$LCL3_N$	22.98	$I27_N$	22.87	$I46_N$	22.91
LPT_{FF}	4.64	$LCL3_{FF}$	3.63	$I27_{FF}$	3.25	$I46_{FF}$	3.66
LPT_{BF}	4.67	$LCL3_{BF}$	3.32	$I27_{BF}$	3.02	$I46_{BF}$	3.34
$VSHARP_N$	27.85	LCC_N	23.01	$I28_N$	22.90	$I48_N$	23.37
$VSHARP_{FF}$	24.01	LCC_{FF}	2.86	$I28_{FF}$	3.70	$I48_{FF}$	4.33
$VSHARP_{BF}$	23.91	LCC_{BF}	3.11	$I28_{BF}$	3.36	$I48_{BF}$	3.92

Table 7.7: Values of $ARPD$ of the dispatching rules.

In summary, it can be concluded that LCC_{FF} is the most effective dispatching rule for solving the problem under study. It reaches the lowest $ARPD$ and, with the exception of $LCL1_{BF}$, significantly improves the results. As can be seen, there is not a significant difference between the FF and BF policies from an $ARPD$ point of view. In turn, the experimental results reveal that applying different indicators at each stage and adopting the SPT rule to arrange related jobs would improve the quality of solutions.

7.6.4 Comparison of the different heuristics

In order to determine the effectiveness of the proposed heuristics $CH1, CH2, CH1_{NLS}$ and $CH2_{NLS}$, a series of alternative heuristics from the relevant literature and properly adapted to the problem at hand have been involved in a comprehensive experimental analysis. Regarding $CH1_{NLS}$ and $CH2_{NLS}$, they are implemented considering $\gamma = \{\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}, n\}$ resulting in $CH1_{NLS(n/4)}$, $CH1_{NLS(n/2)}$, $CH1_{NLS(3n/4)}$, $CH1_{NLS(n)}$,

$CH2_{NLS(n/4)}$, $CH2_{NLS(n/2)}$, $CH2_{NLS(3n/4)}$ and $CH2_{NLS(n)}$. Hence, we propose $CH1$, $CH2$ and these eight variants of $CH1_{NLS}$ and $CH2_{NLS}$. All the heuristics have been employed to solve the instances from the benchmark $\mathscr{B}_4$ in Section 3.2. The average results in terms of $ARPD$, $ACPU$, and $ARPT$ are shown in Table 7.8 and graphically in Fig. 7.7, where the y-axis represents the $ARPD$ for each heuristic and the x-axis represents the $ARPT$. Observing Table 7.8, where the efficient frontier is marked in bold, the next conclusions can be obtained:

- Regarding to the methods applying bin-packing policies (AA, $SMN13$, $SMN14$ and PGF), they provide the best computational times, being the AA methods the fastest in terms of $ACPU$ and $ARPT$. Comparing the policies, it can be observed that the case NONE yields poor $ARPD$ results, being the cases with BF and FF more efficient with similar results. Among them, $SMN13_{BF}$ provides the best $ARPD$ value (1.88) with competitive computational time (around 0.01 seconds on average).

- As far as the alternative methods are concerned, except $CH1$, the computational times are considerably higher. On the one hand, among the adapted methods, $PFF1$, IT_{PFF1}, $PFF2$ and IT_{PFF2}, $PFF2$ improves the $ARPD$ with respect to $SMN13_{BF}$ (1.06), increasing the computational time to 5.55 seconds on average. On the other hand, for the proposed methods, the results provided by $CH1$ are remarkable, with a very good performance and competitive computational time. For the rest of the proposed methods, the computational effort pays off due to the good $ARPD$ results (less than 0.03 for $CH2$ and its variants).

- The effect of the NLS method on the proposed heuristics is worthy to be investigated as well. In the case of $CH1$ there is an interesting improvement by applying the NLS method, but the length of γ increases considerably the computational times, so the case $\gamma = n/4$ seems to be the most suitable option. In the case of $CH2$, it improves considerably the $ARPD$, but also consumes more computational time. This case should be analysed depending on the size of the instance, since the biggest instances increase the computational time, but small instances provide good $ARPD$ values in reasonable computational time.

- Regarding the ICA algorithm, the results reached with the stopping criteria given by the composite heuristics are shown in Table 7.9. It can be observed

Heuristic	$ARPD$	$ACPU$	$ARPT$	Heuristic	$ARPD$	$ACPU$	$ARPT$
$SMN13_N$	19.32	0.0088	0.0086	IT_{PFF1}	2.30	18.4105	6.5694
$SMN13_{FF}$	2.06	0.0101	0.0097	$PFF2$	1.06	5.5431	2.1214
$\boldsymbol{SMN13_{BF}}$	**1.88**	**0.0102**	**0.0099**	IT_{PFF2}	1.47	10.7562	4.1292
$SMN14_N$	22.92	0.0092	0.0088	$\boldsymbol{CH1}$	**0.29**	**0.0978**	**0.0564**
$SMN14_{FF}$	3.25	0.0109	0.0106	$\boldsymbol{CH1_{NLS(n/4)}}$	**0.19**	**1.3746**	**0.5365**
$SMN14_{BF}$	2.95	0.0111	0.0108	$\boldsymbol{CH1_{NLS(n/2)}}$	**0.16**	**5.2115**	**1.9824**
AA_N	22.86	0.0006	0.0006	$CH1_{NLS(3n/4)}$	0.15	11.6022	4.3910
AA_{FF}	3.12	0.0006	0.0006	$CH1_{NLS(n)}$	0.15	20.5503	7.7663
$\boldsymbol{AA_{BF}}$	**2.99**	**0.0034**	**0.0006**	$\boldsymbol{CH2}$	**0.03**	**5.7693**	**2.2595**
PGF_N	21.85	0.0038	0.0034	$\boldsymbol{CH2_{NLS(n/4)}}$	**0.02**	**7.0397**	**2.7359**
PGF_{FF}	3.62	0.0684	0.0368	$\boldsymbol{CH2_{NLS(n/2)}}$	**0.01**	**10.8705**	**4.1792**
PGF_{BF}	3.61	0.0796	0.0434	$CH2_{NLS(3n/4)}$	0.01	17.2492	6.5836
$PFF1$	3.39	4.7372	1.8045	$CH2_{NLS(n)}$	0.01	26.1883	9.9540

Table 7.8: Summary of results of the different heuristics. The efficient frontier is marked in bold.

that, fixing as stopping criteria the time required by the composite heuristics, the results are worse[14] than those provided by the proposed heuristics. The $ARPD$ values range from 2.32 to 2.19, far for the results given by $CH1_{NLS}$ and $CH2_{NLS}$.

The results from Table 7.8 are graphically shown in Figure 7.7, where the y-axis holds the $ARPD$ for each heuristic and the x-axis represents the $ARPT$, i.e., the $ACPU$ in logarithmic scale. In order to determine the efficient frontier and establish the statistical significance of these results, a Holm's procedure (Holm, 1979) is performed evaluating each hypothesis by using a non-parametric Mann-Whitney test, assuming a 0.95 confidence level, i.e., $\alpha = 0.05$, to establish the p-value of each hypothesis is performed considering the $ARPD$ as response variable. In Holm's test, the hypotheses are sorted in non-descending order of the p-values obtained in the Mann-Whitney test. There are not significant evidences to accept each hypothesis if $p \leq \alpha/(N-i+1)$ where N is the total number of hypotheses and i the hypothesis evaluated. The results can be seen in Table 7.10, where R means that the hypothesis is rejected by Holm's procedure.

[14]The results obtained by this metaheuristic are not good since the method is not able to converge with the time given as stopping criteria.

Heuristic	$ARPD$	Metaheuristic	$ARPD$
$CH1_{NLS(n/4)}$	0.19	$ICA_{CH1_{NLS(n/4)}}$	2.32
$CH1_{NLS(n/2)}$	0.16	$ICA_{CH1_{NLS(n/2)}}$	2.30
$CH1_{NLS(3n/4)}$	0.15	$ICA_{CH1_{NLS(3n/4)}}$	2.26
$CH1_{NLS(n)}$	0.15	$ICA_{CH1_{NLS(n)}}$	2.21
$CH2_{NLS(n/4)}$	0.02	$ICA_{CH2_{NLS(n/4)}}$	2.29
$CH2_{NLS(n/2)}$	0.01	$ICA_{CH2_{NLS(n/2)}}$	2.27
$CH2_{NLS(3n/4)}$	0.01	$ICA_{CH2_{NLS(3n/4)}}$	2.23
$CH2_{NLS(n)}$	0.01	$ICA_{CH2_{NLS(n)}}$	2.19

Table 7.9: Summary of results of the proposed composite heuristics and the metaheuristic ICA.

i	Hi	p-value	Mann-Whitney	$\alpha/(k-i-1)$	Holm's Procedure
1	$SMN13_{BF}$=$SMN13_{FF}$	0.000	R	0.0083	R
2	$CH1_{NLS(n/4)}$=$PFF1$	0.000	R	0.0100	R
3	$CH1_{NLS(n/2)}$=$PFF2$	0.000	R	0.0125	R
4	$CH2_{NLS(n/4)}$=IT_{PFF2}	0.000	R	0.0167	R
5	$CH2_{NLS(n/2)}$=$CH1_{NLS(3n/4)}$	0.000	R	0.0250	R
6	AA_{BF}=AA_{FF}	0.006	R	0.0500	R

Table 7.10: Mann-Whitney's procedure with response variable $ARPD$.

In this test, each heuristic candidate to be part of the efficient frontier is compared to the closest heuristic, in terms of $ARPD$. As can be seen, all the hypotheses are rejected. Therefore, the efficient frontier is formed by heuristics AA_{BF}, $SMN13_{BF}$, $CH1$, $CH1_{NLS(n/4)}$, $CH1_{NLS(n/2)}$, $CH2$, $CH2_{NLS(n/4)}$ and $CH2_{NLS(n/2)}$. In conclusion, regarding methods AA_{BF} and $SMN13_{BF}$, they are capable of assuring a significant quality of solution with a remarkably small computational effort with values of $ARPD$ of 1.88 and 2.99, respectively. The proposed heuristic $CH1$ and its variants reaches a very good performance, with a considerably decrease in the value of $ARPD$, and competitive computational time. Finally, the heuristic $CH2$ and its variants consume more computational time, but is a time-consuming heuristic that in turn is capable of reaching the best performance in terms of $ARPD$.

To see the differences $ARPD$ clearly, the 95% confidence intervals for the algorithms whose $ARPD$ value is lower than 2.5 are shown in Figure 7.8 and for the

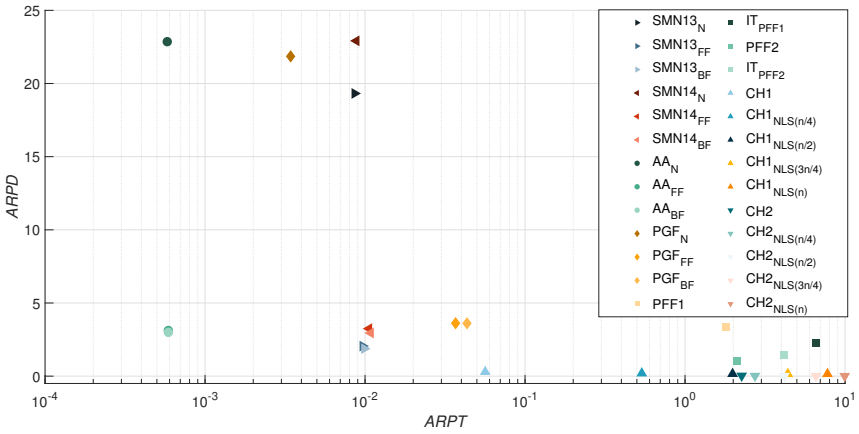

Figure 7.7: $ARPD$ versus $ARPT$ of the different heuristics.

proposed algorithms in Figure 7.9. Based on these results, it can be concluded that the proposed heuristics are the most efficient, since they are statistically different from the rest.

To analyse the impact of the periodic maintenance constraint, the results of $ARPD$ in terms of T are shown in Table 7.11. For all the cases, the values of $ARPD$ increase from $T = 200$ to $T = 300$, and then they get lower as the size of the bins increases. Therefore, the performance of all the heuristics compared in this evaluation gets better as the constraint of periodic maintenance is relaxed.

7.7 Conclusions of the chapter

In this chapter, we have considered the 2-stage assembly scheduling problem with periodic maintenance and the objective of minimising the makespan (see GO5). To the best of our knowledge, such unavailability constraint has never been studied for the 2-stage assembly scheduling problem. In this scenario, with the periodic maintenance constraint, all the machines stop periodically after a given time interval and jobs cannot be processed during these periods in which the machines are not available. We have also considered that jobs are non-resumable, which means that preemption is not allowed, so if a job cannot be finished within an availability period, then it should be scheduled in the next one. The periodic maintenance constraint makes the problem under study highly connected to the bin-packing problem, and, therefore, different

Table 7.11: $ARPD$ of heuristics by levels of T.

T	$SMN13_N$	$SMN13_{FF}$	$SMN13_{BF}$	$SMN14_N$	$SMN14_{FF}$	$SMN14_{BF}$	AA_N
100	17.83	1.54	1.28	22.71	3.50	2.98	21.18
200	29.80	3.09	2.76	33.02	4.76	4.23	34.44
300	15.80	2.08	2.00	19.60	2.75	2.64	19.89
400	13.85	1.52	1.47	16.34	1.97	1.95	15.92
Average	19.32	2.06	1.88	22.92	3.25	2.95	22.86
T	AA_{FF}	AA_{BF}	PGF_N	PGF_{FF}	PGF_{BF}	$PFF1$	IT_{PFF1}
100	2.84	2.73	17.37	1.49	1.34	1.82	0.89
200	4.07	3.70	37.28	6.87	7.00	4.54	3.04
300	3.16	3.13	15.34	3.01	2.91	4.06	3.10
400	2.42	2.41	17.43	3.11	3.19	3.14	2.15
Average	3.12	2.99	21.85	3.62	3.61	3.39	2.30
T	$PFF2$	IT_{PFF2}	$CH1$	$CH1_{NLS(n/4)}$	$CH1_{NLS(n/2)}$	$CH1_{NLS(3n/4)}$	$CH1_{NLS(n)}$
100	1.18	1.34	0.26	0.18	0.10	0.10	0.10
200	1.30	2.26	0.37	0.26	0.24	0.23	0.23
300	1.01	1.32	0.29	0.17	0.15	0.14	0.14
400	0.76	0.95	0.25	0.17	0.15	0.14	0.14
Average	1.06	1.47	0.29	0.19	0.16	0.15	0.15
T	$CH2$	$CH2_{NLS(n/4)}$	$CH2_{NLS(n/2)}$	$CH2_{NLS(3n/4)}$	$CH2_{NLS(n)}$		
100	0.01	0.00	0.00	0.00	0.00		
200	0.03	0.01	0.01	0.01	0.01		
300	0.04	0.03	0.01	0.01	0.01		
400	0.03	0.02	0.01	0.00	0.00		
Average	0.03	0.02	0.01	0.01	0.01		

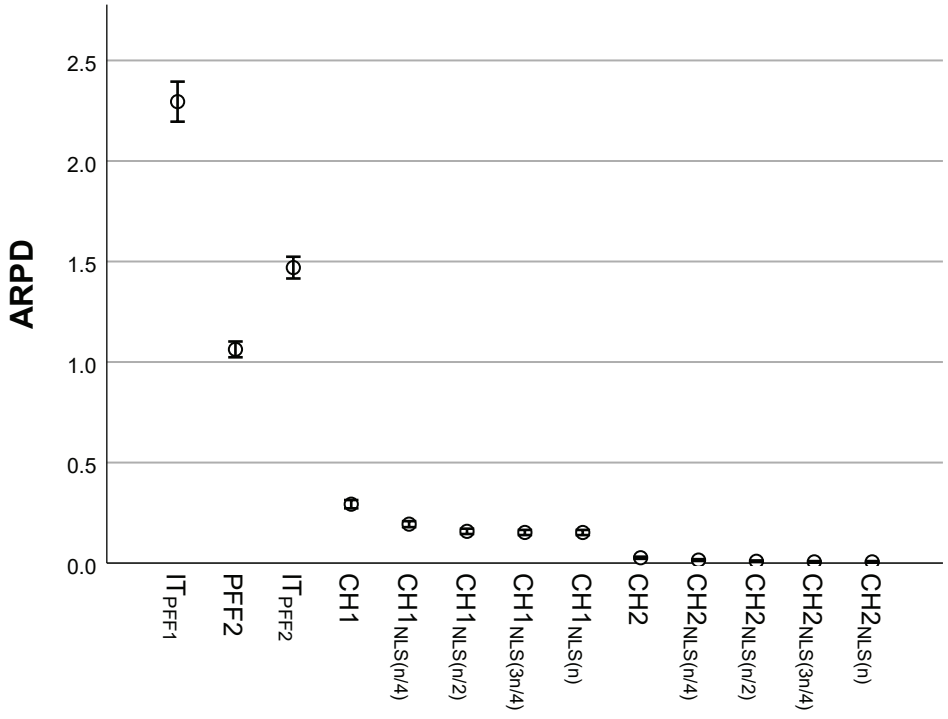

Figure 7.8: 95% confidence interval for $ARPD$ of the algorithms with $ARPD \leq 2.5$.

bin packing policies are applied to construct a schedule and evaluate the objective function.

The first objective of this chapter is to implement the MILP model, presented in Section 2.4, to optimally solve the considered problem and to analyse the results obtained in small size instances. The second objective is to provide efficient heuristics to solve the considered problem. Therefore, two specific constructive heuristics have been designed taking into account the characteristics of the problem. Besides, we have also proposed a local search mechanism based on interchanging a given number of jobs in the sequence provided by the constructive heuristics. The third objective is to check the efficiency of the proposal with the state-of-the-art algorithms (see GO5). To do so, heuristics proposed to solve related problems have been adapted to the problem under study. The adaptation of each one of the methods has been carried out in two steps. From the adaptation, we have obtained two groups of methods depending on

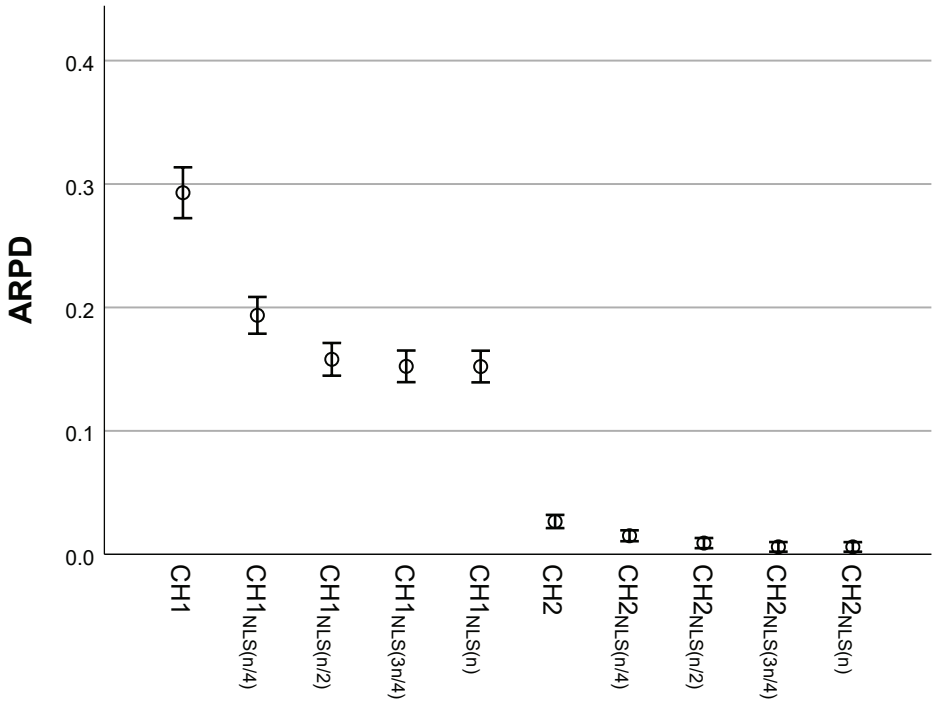

Figure 7.9: 95% confidence interval for $ARPD$ of the proposed algorithms.

the complexity and computational effort: dispatching rules and heuristics. Then, two computational evaluations have been carried out. One experimentation to establish the most efficient dispatching rules, and another one to determine the best heuristics to solve the 2-stage assembly scheduling problem. Regarding the dispatching rules, the results show that the adaptation LCC_{FF} provides the best results. This method makes use of an index considering all the processing times and arrange the jobs according to the SPT rule. From the second computational evaluation, the proposed heuristics $CH1$, $CH1_{NLS(n/4)}$ and $CH1_{NLS(n/2)}$ provide better results, in terms of $ARPD$, than the best adapted heuristics (AA_{BF}, $SMN13_{BF}$) with competitive CPU times. Moreover, the proposed heuristics $CH2$, $CH2_{NLS(n/4)}$ and $CH2_{NLS(n/2)}$ provide the best results, in terms of $ARPD$, with higher computational times.

Part IV

Conclusions

Chapter 8

Final remarks

8.1 Main findings

In this thesis, we have addressed the 2-stage assembly scheduling layout (2-ASP), which has received increasing attention from researchers because of its applications in industry. This problem is a common layout in manufacturing scheduling research, as many products are made up of different components that need to be manufactured in the earlier stages and then assembled into final products in a later stage. The goal of this thesis is to deeply study this important scheduling problem, by proposing new hard instances and by designing new efficient algorithms to solve it. To achieve this goal, the general research objectives identified in Section 1.2 have been addressed in the four parts of this thesis as follows:

[GO1] To review the 2-ASP literature in order to identify the problems addressed in this thesis, the most relevant objectives, and the most interesting constraints.

In Chapter 4, the literature related to the problem under study was reviewed. As the 2-ASP with identical parallel machines in the second stage has been scarcely studied, we also reviewed the state-of-the-art of related problems. In Section 4.2, we review the literature related to the sets of instances used proposed to solve the 2-ASP (variants SA and MA) and the related scheduling problems (CO and PM) with total completion time criterion.

Regarding the 2-ASP to minimise the total completion time, it was addressed in Section 4.3, where 19 heuristics proposed to solve the $DPm \rightarrow Pm||\sum C_j$, $DPm \rightarrow 1||\sum C_j$ and $DPm \rightarrow 0||\sum C_j$ problems were reviewed.

Regarding the 2-ASP-pm to minimise the makespan, it was handled in Section 4.4, where the most relevant approximate methods proposed to solve the $1|nr-pm|C_{\max}$, $DPm \rightarrow 1||C_{\max}$ and $F2|nr-pm|C_{\max}$ were reviewed.

[GO2] To design a new benchmark of hard and exhaustive instances for testing the efficient approximate algorithms in the literature.

After reviewing the literature and ensuring the absence of a commonly accepted set of representative instances of 2-ASP, in Chapter 5, we proposed two comprehensive benchmarks for the problem under study with total completion time criterion to achieve this goal, one for each variant considered, SA and MA. To deal with that, we follow the methodology from the literature to generate the new benchmarks and satisfy the characteristics found in the state-of-the-art: adequacy, hardness, exhaustiveness, and amenability for statistical analysis. Then, a computational analysis was performed to determine the relationship among the variants SA and MA and the related scheduling problems. In this analysis, we generated different scenarios and preliminary testbeds, which were solved using exact and approximate methods.

[GO3] To propose faster and more efficient approximate algorithms to solve the 2-ASP with total completion time objective, based on the conclusions obtained from GO1.

To cover this goal, we developed several heuristic algorithms to solve the 2-stage assembly scheduling problem with total completion time minimisation ($DPm \rightarrow Pm||\sum C_j$). More specifically, in Chapter 6, we proposed a fast constructive heuristic, CH_{MA}, which iteratively constructs a sequence. Then, CH_{MA} was embedded into a beam-search-based heuristic, $BSCH_{MA}$.

[GO4] To demonstrate the efficiency and good performance of the solution procedures developed in GO3.

In this thesis, each proposed approximate algorithm was always compared with the algorithms adapted and implemented from the literature. In Chapter 3, the indicators used to compare the different methods in terms of solution quality and computational effort were presented, and the conditions to carry out a fair comparison were also detailed (such as the use of the same computer, the same programming language (C#), among others).

Regarding the approximate algorithms for the 2-ASP with total completion time minimisation, a computational evaluation comparing the different versions of the $BSCH_{MA}$ heuristic was carried out in Section 6.5.2 and the best were selected. After

that, in Sections 6.5.3 and 6.5.4 the proposals were compared with 19 heuristics from the literature under the same conditions.

Regarding the approximate algorithms for the 2-ASP-pm with makespan minimisation, the proposed heuristics for that problem were compared with the heuristics adapted and implemented from the related scheduling problems. In Sections 7.6.3 and 7.6.4, a total of 84 dispatching rules and seven heuristics, respectively, were compared under the same conditions.

[GO5] To extend the goals GO3 and GO4 to some constrained 2-ASP based on real manufacturing environments.

This objective was addressed in Chapter 7, where we considered the 2-stage assembly scheduling problem with periodic maintenance constraint and makespan minimisation ($DPm \rightarrow 1|nrpm|C_{\max}$). We designed two constructive heuristics and a local search mechanism to solve this problem. The first proposal, $CH1$, constructs a sequence by iteratively appending jobs at the end of a sequence. Then, the second constructive heuristic is then obtained by applying a partial local search mechanism enabled at each iteration of $CH1$. Finally, we developed a novel local search mechanism NLS, which will be implemented after constructing a complete sequence. After proposing the heuristics, a computational evaluation is carried out comparing the proposals with the heuristics adapted from the literature.

8.2 Results

SCI indexed journals

Parts of this thesis have been published in SCI indexed journals. The following publications are directly derived from this thesis:

- Talens, C., Fernandez-Viagas, V., Perez-Gonzalez, P., Framinan, J.M. New efficient constructive heuristics for the two-stage multi-machine assembly scheduling problem (2020) Computers and Industrial Engineering, 140, art. no. 106223 (2021 Impact Factor: 7.180).

- Talens, C., Perez-Gonzalez, P., Fernandez-Viagas, V., Framinan, J.M. New hard benchmark for the 2-stage multi-machine assembly scheduling problem: Design and computational evaluation (2021) Computers and Industrial Engineering, 158, art. no. 107364 (2021 Impact Factor: 7.180).

- Talens, C., Fernandez-Viagas, V., Perez-Gonzalez, P., Costa A. Constructive and composite heuristics for the 2-stage assembly scheduling problem with periodic maintenance and makespan objective (2022) Expert Systems with Applications, 206, art. no. 117824 (2021 Impact Factor: 8.665).

Additionally, during the development of this thesis, the following papers on related scheduling problems have been published in SCI indexed journals:

- Fernandez-Viagas, V., Talens, C., Framinan, J.M. Assembly flowshop scheduling problem: Speed-up procedure and computational evaluation (2021) European Journal of Operational Research (2021 Impact Factor: 6.363).

Papers in conference proceedings

We briefly mention the contributions in national and international conferences:

- Talens, C., Navarro-Garcia, B., Fernandez-Viagas, V., Perez-Gonzalez, P. The 2-stage Assembly Scheduling Problem with periodic maintenance. The second EUROYoung Workshop. Porto (Portugal), June 21-22, 2022.

- Talens, C., Fernandez-Viagas, V., Perez-Gonzalez, P. Heurísticas para el problema de ensamblado en dos etapas con mantenimiento periódico. XXXIX Congreso Nacional de Estadística e Investigación Operativa (SEIO 2022). Granada (Spain), June 7-10, 2022.

- Fernandez-Viagas, V., Talens, C., Framinan, J.M. A novel acceleration procedure for the assembly flow shop scheduling problem. 31st European Conference on Operational Research (EURO 2021). Athens (Greece), July 11-14, 2021.

- Talens, C., Fernandez-Viagas, V., Perez-Gonzalez, P. The two-stage assembly scheduling problem with periodic maintenance: New approximate methods. 31st European Conference on Operational Research (EURO 2021). Athens (Greece), July 11-14, 2021.

- Talens, C., Fernandez-Viagas, V., Perez-Gonzalez, P. Generating instances for the two-stage multi-machine assembly scheduling problem. 17th International Conference on Project Management and Scheduling (PMS 2020/2021). Online, April 21-23, 2021.

- Fernandez-Viagas, V., Molina-Pariente, J.M., Talens, C., Framinan, J.M. An acceleration procedure for several objective functions in the permutation flow shop scheduling problem. 17th International Conference on Project Management and Scheduling (PMS 2020/2021). Online, April 21-23, 2021.

- Talens, C., Perez-Gonzalez, P., Fernandez-Viagas, V., Framinan, J.M. New Approximate Methods for the Two-Stage Multi-Machine Assembly Scheduling Problem. Operation Research 2019 Conference. Dresden (Germany), September 3-6, 2019.

- Talens, C., Perez-Gonzalez, P., Fernandez-Viagas, V., Framinan, J.M. New efficient constructive heuristics for the two-stage multi-machine assembly scheduling problem. 1st EUROYoung Workshop. Sevilla (Spain), May 2-3, 2019.

- Talens, C., Perez-Gonzalez, P., Fernandez-Viagas, V., Framinan, J.M. New approximate methods for the two-stage multi-machine assembly scheduling problem. 19th European Conference on Operational Research (EURO 2018). Valencia (Spain), July 8-11, 2018.

Research projects

This thesis was carried out under the grant "Predoctoral Research Fellow (FPI BES-2017-081028)" funded by the Spanish Ministry of Economy and Competitiveness and has been developed within the framework of the following research projects on manufacturing scheduling.

- PROMISE - "Production Management under Imperfect and Scarce Data" funded by the Spanish Ministry of Economy and Competitiveness (reference DPI2016-80750-P).

- EFECTOS - "Escenarios de Fabricación Emergentes: CaracTerización Optimización y Simulación" funded by Regional Government of Andalusia (reference US-1264511).

- DEMAND - "Decision-making for Emerging MANufacturing and Distribution" funded by Regional Government of Andalusia (reference P18-FR-1149).

- ASSORT - "Advanced Support for Smart Operations & Remanufacture" funded by the Spanish Science, Innovation and Universities (reference PID2019-108756 RB-I00).

8.3 Future research lines

In this section, we present the main future research lines of this thesis.

Regarding the design of the new benchmarks, the total completion time has been adopted as an objective since it was by far the most studied criterion for the 2-ASP. 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 orientated 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.

Regarding the 2-ASP and 2-ASP-pm, it could be interesting to analyse new objective functions, such as those related to just-in-time objectives, or different layouts in the second stage, such as unrelated parallel machines. In addition, further analysis could be conducted by comparing the efficiency of the new state-of-the-art heuristics when these are embedded in metaheuristics.

Bibliography

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. *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.

Alemão, D., Rocha, A. D., and Barata, J. (2021). Smart manufacturing scheduling approaches—systematic review and future directions. *Applied Sciences*, 11(5).

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.

Ángel-Bello, F., Álvarez, A., Pacheco, J., and Martínez, I. (2011). A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. *Computers and Mathematics with Applications*, 61(4):797–808.

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.

Della Croce, F. and T'kindt, V. (2003). Improving the preemptive bound for the one-machine dynamic total completion time scheduling problem. *Operations Research Letters*, 31(2):142–148.

Erenay, F., Sabuncuoglu, I., Toptal, A., and Tiwari, M. (2010). New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs. *European Journal of Operational Research*, 201(1):89–98.

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

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.

Fernandez-Viagas, V. and Framinan, J. M. (2017). 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. M. (2020). Design of a testbed for hybrid flow shop scheduling with identical machines. *Computers and Industrial Engineering*, 141.

Fernandez-Viagas, V., Leisten, R., and Framinan, J. M. (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., Valente, J. M. S., and Framinan, J. M. (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. M., Leisten, R., and García, R. R. (2014). *Manufacturing scheduling systems: An integrated view on models, methods and tools*, volume 9781447162728.

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.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and approximation in deterministic machine scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

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). A simple rejective test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Hsu, C. J., Low, C., and Su, C. T. (2010). A single-machine scheduling problem with maintenance activities to minimize makespan. *Applied Mathematics and Computation*, 215(11):3929–3935.

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.

Ji, M., He, Y., and Cheng, T. C. (2007). Single-machine scheduling with periodic maintenance to minimize makespan. *Computers and Operations Research*, 34(6 SPEC. ISS.):1764–1770.

Kaabi, J. and Harrath, Y. (2019). Scheduling on uniform parallel machines with periodic unavailability constraints. *International Journal of Production Research*, 57(1):216–227.

Komaki, G. M. and Kayvanfar, V. (2015). Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. *Journal of Computational Science*, 8:109–120.

Komaki, G. M., Teymourian, E., Kayvanfar, V., and Booyavi, Z. (2017). Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. *Computers and Industrial Engineering*, 105:158–173.

Koulamas, C. and Kyparisis, G. (2001). The three-stage assembly flowshop scheduling problem. *Computers and Operations Research*, 28(7):689–704.

Lee, C. Y. (1996). Machine scheduling with availability constraints. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pages 22–1–22–14.

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., Cheng, T. C., and Chou, A. S. (2006). Scheduling in an assembly-type production chain with batch transfer. *Omega*, 35(2):143–151.

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*.

Liu, J. and Reeves, C. R. (2001). Constructive and composite heuristic solutions to the $P//\sum C_i$ scheduling problem. *European Journal of Operational Research*, 132(2):439–452.

Low, C., Hsu, C. J., and Su, C. T. (2010a). A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance. *Expert Systems with Applications*, 37(9):6429–6434.

Low, C., Ji, M., Hsu, C. J., and Su, C. T. (2010b). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, 34(2):334–342.

Manzini, R., Gamberi, M., Regattieri, A., and Persona, A. (2004). Framework for designing a flexible cellular assembly system. *International Journal of Production Research*, 42(17):3505–3528.

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.

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.

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

Perez-Gonzalez, P., Fernandez-Viagas, V., and Framinan, J. M. (2020). Permutation flowshop scheduling with periodic maintenance and makespan objective. *Computers and Industrial Engineering*, 143:106369.

Perez-Gonzalez, P. and Framinan, J. M. (2018). Single machine scheduling with periodic machine availability. *Computers and Industrial Engineering*, 123(December 2017):180–188.

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.

Rad, S., Ruiz, R., and Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *Omega*, 37(2):331–345.

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.

Sanchez, R. (1995). Strategic flexibility in product competition. *Strategic Management Journal*, 16:135–159.

Seidgar, H., Kiani, M., Abedi, M., and Fazlollahtabar, H. (2014). An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem. *International Journal of Production Research*, 52(4):1240–1256.

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.

Shoaardebili, N. and Fattahi, P. (2015). Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints. *International Journal of Production Research*, 53(3):944–968.

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

Sotskov, Y., Tautenhahn, T., and Werner, F. (1996). Heuristics for permutation flow shop scheduling with batch setup times. *OR Spectrum*, 18(2):67–80.

Sun, X., Morizawa, K., and Nagasawa, H. (2003). Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. *European Journal of Operational Research*, 146(3):498–516.

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.

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.

Valente, J. M. (2010). Beam search heuristics for quadratic earliness and tardiness scheduling. *Journal of the Operational Research Society*, 61(4):620–631.

Valente, J. M. and Alves, R. A. (2008). Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. *Computers and Operations Research*, 35(7):2388–2405.

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.

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*, 41:97 – 110.

Yazdani, M., Jolai, F., Taleghani, M., and Yazdani, R. (2018). A modified imperialist competitive algorithm for a two-agent single-machine scheduling under periodic maintenance consideration. *International Journal of Operational Research*, 32(2):127 – 155.

Yu, X., Zhang, Y., and Steiner, G. (2014). Single-machine scheduling with periodic maintenance to minimize makespan revisited. *Journal of Scheduling*, 17(3):263–270.

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.

Zhao, F., Zhao, L., Wang, L., and Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Expert Systems with Applications*, 160:113678.

List of Tables

List of Figures