Universidade de São Paulo
Escola de Engenharia de São Carlos

Fernando Luis Rossi

Uma contribuição para o problema de programação mixed no-idle flowshop com tempos de preparação dependentes da sequência: análises e métodos de solução

São Carlos
2019

University of São Paulo
São Carlos School of Engineering

Fernando Luis Rossi

A contribution for the mixed no-idle flowshop scheduling problem with sequence-dependent setup times: analysis and solutions procedures

São Carlos
2019

Fernando Luis Rossi

Uma contribuição para o problema de programação mixed no-idle flowshop com tempos e preparação dependentes da sequência: análises e métodos de solução

Tese apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Doutor em Ciências - Programa de Pós-Graduação em Engenharia de Produção.

Área de concentração: Pesquisa Operacional

Supervisor: Prof. Dr. Marcelo Seido Nagano

São Carlos

2019

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Rossi, Fernando Luis
R831u Uma contribuição para o problema de programação mixed no-idle flowshop com tempos de preparação dependentes da sequência: análises e métodos de solução / Fernando Luis Rossi; orientador Marcelo Seido Nagano. São Carlos, 2019.

Tese (Doutorado) - Programa de Pós-Graduação em Engenharia de Produção e Área de Concentração em Processos e Gestão de Operações -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. Flowshop. 2. No-idle. 3. Tempos de Preparação. 4. Heurísticas. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

Fernando Luis Rossi

A contribution for the mixed no-idle flowshop scheduling problem with sequence-dependent setup times: analysis and solutions procedures

Doctoral Dissertation presented to the Graduate Program in Production Engineering of the São Carlos School of Engineering at University of São Paulo to obtain the degree of Doctor of Science.

Concentration Area: Operational Research

Supervisor: Prof. Dr. Marcelo Seido Nagano

São Carlos

2019

Catalog card prepared by Patron Service at "Prof. Dr. Sergio Rodrigues Fontes" Library at EESC/USP

Rossi, Fernando Luis
R831c A Contribution for the mixed no-idle flowshop scheduling problem with sequence-dependent setup times : analysis and solutions procedures / Fernando Luis Rossi ; Thesis directed by Marcelo Seido Nagano. -- São Carlos, 2019.

Doctoral (Dissertation) - Graduate Program in Production Engineering and Research Area in Process and Operations Management – São Carlos School of Engineering, at University of São Paulo, 2019.

1. Flowshop. 2. No-idle. 3. Setup times. 4. Heuristics. I. Title.

Elena Luzia Palloni Gonçalves – CRB 8/4464

FOLHA DE JULGAMENTO

Candidato: Bacharel **FERNANDO LUÍS ROSSI.**

Título da tese: "Uma contribuição para o problema de programação mixed no-idle flowshop com tempos de preparação dependentes da sequência: análises e métodos de solução".

Data da defesa: 03/07/2019.

Comissão Julgadora: **Resultado:**

Prof. Dr. **Marcelo Seido Nagano**
(Orientador)
(Escola de Engenharia de São Carlos/EESC) Aprovado

Prof. Associado **Evandro Eduardo Seron Ruiz** Aprovado
(Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto/FFCLRP-USP)

Prof. Dr. **Roberto Fernandes Tavares Neto** Aprovado
(Universidade Federal de São Carlos/UFSCar)

Prof. Dr. **Paulo Rogério Politano** APROVADO
(Universidade Federal de São Carlos/UFSCar)

Profa. Associada **Débora Pretti Ronconi** Aprovada
(Escola Politécnica/EP-USP)

Coordenador do Programa de Pós-Graduação em Engenharia de Produção:
Prof. Dr. **Marcelo Seido Nagano**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Murilo Araujo Romero**

This work is wholeheartedly dedicated to my wife, daughter and beloved parents, who have been our source of inspiration and gave us strength when we thought of giving up, who continually provide their moral, spiritual, emotional, and financial support.

To our brothers, sisters, relatives, mentor, friends, and classmates who shared their words of advice and encouragement to finish this study.

And lastly, we dedicated this book to the Almighty God, thank you for the guidance, strength, power of mind, protection and skills and for giving us a healthy life. All of these, we offer to you.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Marcelo Seido Nagano for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I also want to be thankful with my colleagues in the Department of Managment of Federal Institute of São Paulo. Although each of them has been really important to carry out this Thesis, I want to be specially grateful to the support of Rodolfo Butcher, Cynthia Regina Fischer, Hânia Cecília Pilan and Francisco Manoel Filho.

"Computers are like Old Testament gods; lots of rules and no mercy."
Joseph Campbell, The Power of Myth.

RESUMO

Rossi, F. **Uma contribuição para o problema de programação mixed no-idle flowshop com tempos de preparação dependentes da sequência: análises e métodos de solução**. 2019. 176p. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Neste trabalho é abordado o problema de programação de produção em ambiente mixed no-idle flowshop com tempos de preparação dependente da sequência. Este ambiente de produção ainda não foi estudado na literatura, apesar de estar presente na dinâmica dos sistemas produtivos. No ambiente no-idle flowshop, as máquinas que compõem o sistema não podem ficar ociosas e todas as tarefas são executadas ininterruptamente. Geralmente os motivos são associados à fatores econômicos ou tecnológicos, onde uma máquina parada influencia diretamente o desempenho do sistema produtivo. O ambiente no-idle flowshop está presente no processamento de fibra de vidro, produção de circuitos integrados, em siderúrgicas, dentre outros. Entanto, assumir que todas as máquinas não fiquem ociosas no ambiente produtivo geralmente não é realístico. Uma situação mais realista é considerar o ambiente misto, onde apenas algumas máquinas que compõem o sistema executam as tarefas ininterruptamente, enquanto as outras permitem a ociosidade normalmente. Neste caso, o ambiente é chamado de mixed no-idle flowshop. Na extensão estudada neste trabalho, tempos de preparação que antecedem o processamento das tarefas são considerados nas máquinas em que é permitida a parada. Este é o primeiro trabalho a abordar o problema mixed no-idle flowshop com tempos de preparação. Nesta Tese, métodos heurísticos eficientes para resolução do problema mixed no-idle flowshop foram propostos. Para demonstrar a performance dos métodos desenvolvidos, foram realizadas extensas comparações com métodos considerados estado-da-arte na literatura. Os resultados mostram que as heurísticas propostas fornecem soluções de qualidade com eficiência computacional, superando os métodos da literatura.

Palavras-chave: Flowshop, No-idle, Tempos de Preparação, Heurísticas

ABSTRACT

Rossi, F. **A contribution for the mixed no-idle flowshop scheduling problem with sequence-dependent setup times: analysis and solutions procedures**. 2019. 176p. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

In this work the mixed no-idle permutation flowshop with sequence-depdent setup times scheduling problem is approached. This production environment has not yet been studied in the literature, despite being present in the dynamics of production systems. In the no-idle flowshop environment, the machines cannot be idle and all jobs are processed uninterruptedly. Generally, the reasons are associated with economic or technological factors, where a stationary machine directly influences the performance of the production system. The no-idle flowshop is present in the manufacturing of fiberglass, production of integrated circuits, in steelworks, among others. However, assuming that all machines cannot be idle is often unrealistic. A more realistic situation would consider a mixed environment, where only a few machines perform the jobs uninterruptedly, while the other allow idleness. In this case, the environment is called mixed no-idle flowshop. In the problem extension studied in this work, setup times are considered on machines where idleness is allowed. This is the first work that addresses the mixed no-idle flow shop scheduling problem with setup times. In this Thesis, efficient heuristic methods for solving the mixed no-idle flowshop with setup times scheduling problem are proposed. To demonstrate the performance of the new methods, extensive comparisons with state-of-the-art methods from literature are performed. The results show that the proposed heuristics provide high quality solutions with computational efficiency, outperforming the methods from the literature.

Keywords: Flowshop, No-idle, Setup Times, Heuristics

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AA	Approximate Algorithms
ACO	Ant Colony Optimization Algorithms
B&B	Branch-and-Bound
B&C	Branch-and-Cut
CH	Constructive Heuristic
DE	Differential Evolution Algorithm
DP	Dynamic Programming Formulation
GA	Genetic Algorithms
IG	Iterated Greedy Algorithm
IH	Improvement Heuristic
ILS	Iterated Local Search
IWO	Invasive Weed Optimization Algorithm
MBO	Migration Bird Optimization Algorithms.
ME	Memetic Algorithm.
MILP	Mixed-integer Linear Programming model
PFSP	Permutation Flowshop Scheduling Problem
PSO	Particle Swarm Optimization Algorithms
SI	Saving Index Algorithm

CONTENTS

1 INTRODUCTION

In the past decades, following the massive globalization of markets, companies worldwide struggle in a more competitive market, where corporations from different regions must battle for common customers. As a consequence, efficiency in the production processes of the companies have become more essential than ever (SLACK et al., 2009). Thus, production management is a important aspect for firms to remain competitive. Production management comprises decision making related to many issues, as example the master scheduling, material requirements planning, capacity planning, production scheduling (VOLLMANN, 2005). Between these, production scheduling performs an crucial role on resource productivity and client service (FERNANDEZ-VIAGAS, 2016). Companies around the world must meet promised delivery times, and failing to meet them can result in significant customer losses (LIU; REEVES, 2001).

Production scheduling is a decision process that deals with the allocation of resources to the jobs in a given period of time in order to optimize one or more objectives (PINEDO, 2016). With the objective of determining the best schedule for the shop floor, constraints and the objective of the shop have to be considered. The complexity of the scheduling problem increases and becomes NP-hard even for small scheduling problems (FRAMINAN; LEISTEN, 2003). Also, scheduling decisions should be made in short time intervals requiring a rapid response time, due to several aspects such as the lifetime of a schedule, the delay in the suppliers, arrivals of new jobs to be processed, rescheduling due to failures while processing a job (FERNANDEZ-VIAGAS, 2016). Therefore, the development of fast and efficient solution procedures for solving manufacturing scheduling problems is decisive for companies' efficiency.

Currently, many processing layouts have been utilized by manufacturing industries. The Permutation Flowshop Scheduling Problem (PFSP) has been extensively studied in the literature with several papers published that deal with flowshops and other related problems (RAD; RUIZ; BOROOJERDIAN, 2009). The main reason for this is that the flowshop layout is the common configuration in many real manufacturing scenarios, as it presents several advantages over more general job shop configuration, and, in addition, many job shops are indeed a flow shop for most of the jobs (FERNANDEZ-VIAGAS, 2016). Another reason is that many models and solutions methods for different constraints and layouts have their origins in the PFSP.

The flowshop environment is present in many relevant industry segments such as metallurgical, chemical, and pharmaceutical industries. In certain situations in which profitability is improved by means of (a) maximizing the use of resources, (b) reducing work-in-process inventory or (c) better meeting deadlines, scheduling methods have the

minimization criteria of (a) maximum completion time of a sequence or makespan, (b) total flowtime or (c) total tardiness (MACCARTHY; LIU, 1993). These three objectives are the most relevant for the flowshop environment (LIU; REEVES, 2001; FRAMINAN; LEISTEN, 2003; FERNANDEZ-VIAGAS; FRAMINAN, 2015).

More specifically, in a PFSP a set jobs is processed on machines with the objective of minimising a certain criterion. In a permutation flowshop, the order in which each machine processes the jobs is identical for all machines. In this Thesis, the mixed no-idle PFSP with sequence dependent setup times is studied, which is a variation of the no-idle PFSP. In the no-idle PFSP, machines are not allowed to be idle after a job sequence begins. This condition occurs when their operating costs are high enough to make idle machines economically prohibitive or technological constraints do not allow machines to stop after the process is started. Briefly, in a no-idle environment, machines must process all the jobs of the sequence without interruption. Due to this, when necessary, some jobs must be delayed to ensure that the no-idle constraint is met.

However, in practice we rarely see an environment where there are only no-idle machines (PAN; RUIZ, 2014). In a real production system, it is more common to have a mixed environment in which no-idle machines and idle machines coexist. (PAN; RUIZ, 2014) were the first to formally define this type of mixed production system in a flowshop environment (mixed no-idle PFSP). The mixed no-idle environment can be found in steel production using the continuous casting process as described by (PAN; RUIZ, 2014). Another example arises from the production of truck engine blocks in a foundry (SAADANI; GUINET; MOALLA, 2003). In more detail, this includes casting sand moulds and sand cores. The moulds are filled up with molten metal and they prevent the metal from filling some spaces. The production system is defined by four main activities: the core production shop; the smelting step; the casting line; and the finishing step. The smelting step, casting line and some core production machines must work continuously due to both technical and economic reasons, while the other operations can work with idle times.

Within the flowshop problem, another important consideration is the existence of setup times that precede the jobs. Some example of setup operations are obtaining tools, cleaning machines, positioning raw material, adjustments, inspections and other activities. There is a distinction between two types of setup: a) sequence independent setup ($s_{i,j}$), when the setup time of machine M_i only depends on job J_j; b) sequence dependent setup ($s^i_{j,k}$), when the setup time of machine M_i depends on the previous job J_j and the current job J_k. The setups times can also be described as anticipatory or non-anticipatory. Anticipatory setups are those that can be performed on the machine as soon as this machine finishes the previous job in the sequence. Otherwise, a setup is non-anticipatory. Literature reviews addressing the scheduling problem with setup can be found in Allahverdi, Gupta e Aldowaisan (1999), Allahverdi e Soroush (2008), Allahverdi

et al. (2008), Allahverdi (2015).

In an environment in which all machines are no-idle, the setup time makes no practical sense, as the machine requires continuous processing. However, in a mixed no-idle environment, the existence of setup times is acceptable, and therefore the setup times on regular machines can be considered, which allows idleness. For example, in the continuous casting process in the steel industry, setup times for the machine to accommodate the iron load to be processed or to separate the specific alloying elements for the steel to be produced can be considered. Therefore, the setup times would occur in the stages in which idle times are allowed. For example, the setup times in the finishing step can result from cleaning and adjustment operations needed in the machines and tools used at this stage.

In this work, this special variant of the PFSP is denoted as mixed no-idle PFSP with sequence-dependent setup times. Succinctly, in this new problem, no-idle machines coexist with stages that allow iddleness, in these stages sequence-dependent setup times between the jobs are considered. To the best of our knowledge, this problem has not yet been studied in the literature, although it can be found in the dynamics of productive systems.

1.1 Objectives and outline of the Thesis

As stated in the previous section, the goal of this Thesis is to study the mixed no-idle PFSP with sequence-dependent setup times, both deeply analysing the problem under different criteria and developing new efficient methods to solve the problem. To carry out this goal, the following general research objectives are identified:

OBJ1. To provide in-depth analysis of the mixed no-idle PFSP with sequence-dependent setup times, presenting mathematical models, formulations and calculations methods for different objective functions;

OBJ2. To review the no-idle PFSP, PFSP with setup times and classical PFSP literature for the most common objectives, i.e. makespan, total flowtime or total tardiness minimisation;

OBJ3. To provide efficient methods to solve the mixed no-idle PFSP with sequence dependent setup times for makespan, total flowtime or total tardiness minimisation;

OBJ4. To demonstrate the efficiency and good performance of the new proposed methods thought extensive computational and statistical experiments.

To achieve these objectives, the Thesis have been structured in five chapters as follows:

In Chapter 2 the problem under consideration is stated, where mathematical formulation and calculation methods for the different objective functions are provided in detail. Chapters 3, 4 and 5, the mixed no-idle is analysed under different minimization criteria along three chapters (Chapter 3 - makespan, Chapter 4- total flowtime, Chapter 5 - total tardiness). In each chapter, the main contributions in the literature are reviewed and the state-of-the-art algorithms are presented. We propose new heuristics algorithms to solve the mixed no-idle PFSP under each objective. We developed each heuristics in order to exploit the specific structure of the problem to both reduce the computational times of them and improve the quality of the solutions. Additionally, they are tested in extensive computational evaluations, comparing them with the state-of-the-art algorithms under the same conditions. Finally, in Chapter 6, the conclusions of this research and future research lines are discussed.

2 PROBLEM STATEMENT

The PFSP in industrial environments are frequent, complex and have characteristics peculiar to each company (NAGANO; MOCCELLIN, 2002). In the scope of Operational Research, such problems are represented by models, in most cases mathematical, and must contain the variables and parameters that explain and represent significantly the behaviour of the problems treated (PINEDO, 2016). The problem consists of the determination of the sequence of n jobs which achieves the minimal objective function value when all jobs are processed, in the order, on the m machines of the flow shop. The following additional hypotheses are usually assumed for the PFSP:

- Processing times are known and deterministic.
- No preemption is allowed.
- Transportation times can be considered either insignificant or constant.
- Each job can be processed by at most one machine at the same time.
- Release times are set to 0.
- Each machine can process only one job at the same time.
- All machines are available on the whole scheduling horizon.
- Unlimited in-process inventory is considered.

To obtain the solution of the problem of production scheduling many methods of solution have been proposed. According to Framinan, Leisten e Ruiz-Usano (2005), these methods are mainly: efficient methods of optimal solutions, enumerative methods of optimal solutions, heuristic and metaheuristics methods. Among the efficient methods of optimal solutions, the best known method is Johnson (1954), which determines the optimal solution for the PFSP with two machines and n jobs in a polynomial time. The enumerative methods determine the optimal solution of the problem, but at a high computational cost. Among these, the most studied are the branch & bound procedures (BAPTISTE; HGUNY, 1997; BAGGA, 2003). The last type of solution method is the heuristic and metaheuristics methods, which provide good quality solutions at a computational time that is not very high compared to the other solution methods.

The heuristic methods can be classified in several ways, the most usual being to classify them in: constructive heuristic methods and improvement methods. In the case of constructive heuristic methods, the solution of the problem is obtained: directly from

the ordering of jobs according to priority indexes calculated according to the processing times of the jobs, such as the methods proposed by Palmer (1965) and Koulamas (1998); choosing the best sequence of jobs from a set of sequences (CAMPBELL; DUDEK; SMITH, 1970; HUNDAL; RAJGOPAL, 1988); or from the construction of partial sequences of a set of jobs (subsequences) until a complete sequence is obtained (for example, Nawaz, Enscore e Ham (1983)). In the case of the improvement methods, an initial solution is obtained and, later, by means of some iterative procedure, a better solution is sought than the current one in relation to the adopted measure of performance. Thus, the improvement methods start from an initial solution and look for a better solution in their neighbourhood. Two widely used neighbourhoods are the pairwise exchange and the insertion of jobs (FRAMINAN; GUPTA; LEISTEN, 2004). In the pairwise exchange, all possible exchanges of pairs in a sequence are performed; therefore, if we have a sequence with n tasks, we will have $n(n-1)/2$ possible pairwise exchanges. In the insertion method, a job of the current sequence is removed and subsequently tested in all possible positions, resulting in $n(n-1)$ sequences generated by the insertion procedure.

Finally, a metaheuristic can be seen as a general-purpose heuristic method designed to guide the solution toward promising regions of the search space containing high quality solutions (DORIGO et al., 2008). In addition, metaheuristics may move to not necessary improving solutions, which constitutes the main mechanism to avoid stopping at local optima Crainic e Toulouse (2003). Some examples are the particle swarm algorithm proposed by Pan e Wang (2008b), the evolutionary algorithms of Tasgetiren et al. (2011) and Deng e Gu (2012) or the bee colony algorithm of Tasgetiren et al. (2013b). Before the problem is formally described, the notations used in the expressions are presented in detail in the next section.

2.1 Notations

The notations used to describe the problem are detailed bellow.

J_j : job to be processed;

M_i : machine or stage;

$p_{i,j}$: processing time of job J_j in machine M_i $(i = 1, \ldots, m; j = 1, \ldots, n)$;

d_j : the due date of job J_j;

π : sequence of processing;

π_j or $[j]$: defines a job occupying the jth position of the sequence;

U : set of jobs that have not yet been sequenced;

$C_{i,j}$: the completion time of job J_j in machine M_i;

$S_{i,j}$: the start time of job J_j in machine M_i;

$C'_{i,j}$: the completion time calculated from back to front of J_j in machine M_i;

$C_{max}, C_{m,j}$: the makespan or the completion time of the sequence;

$\sum T_j$, $\sum_{j=1}^{n} \max(C_{m,j} - d_j, 0)$: the total tardiness of the sequence;

$\sum C_j$, $\sum_{j=1}^{n} C_{m,j}$ or $\sum C_j(\pi)$: total flowtime of the sequence π;

$S'_{i,j}$: the start date calculated from back to front of J_j in machine M_i;

M' : set of no-idle machines. Machines that do not belong to M' are defined as regular machines, which allow stopping and also have setup time between jobs.

$s^i_{j,k}$: setup time in machine M_i between jobs J_j and J_k ($k \leq n$, $k \neq j$).

2.2 The no-idle and mixed no-idle PFSP

As mentioned before, a variant of the PFSP is caused when idleness is not allowed on machines (no-idle). Cases like these are common and important in practical situations where, for example, machines with high economic value added are used in the production process. Allowing these machines to be idle many times may not be financially desirable. Clear examples of this arise in the production of integrated circuits through photolithography (RUIZ; VALLADA; FERNÁNDEZ-MARTÍNEZ, 2009). Other examples can be seen in sectors where the machines are of low value, but the machines can not be easily restarted or stand still because the costs would be very high, for example, the ceramics industry uses kilns that use large amounts of gas while operating, in this case, idleness should be avoided, since it would take several days to stop the oven due to a large thermal inertia for heating. In all these cases, idleness should be avoided, either for economic reasons or for technological reasons inherent to the process. In order to understand and better illustrate the no-idle PFSP, an example is presented with three machines and three jobs. The processing times of the jobs are given in Table 1.

Table 1: Small example with three machines and three jobs.

Jobs	J_1	J_2	J_3
M_1	21	10	23
M_2	27	20	15
M_3	7	12	33

A simple way to represent the solution of a production scheduling problem is to use the Gantt Chart. Figure 1 presents a Gantt Chart of an optimal solution for the small

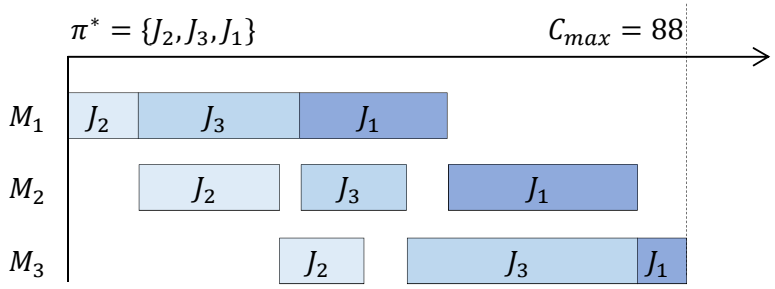

Figure 1: Optimum solution for the PFSP without no-idle.

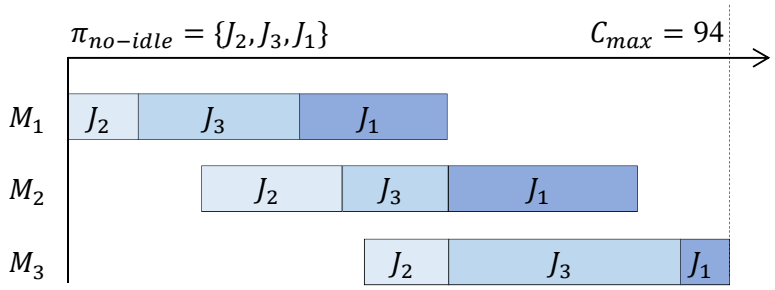

Figure 2: The same solution of Figure 1, now with no-idle machines.

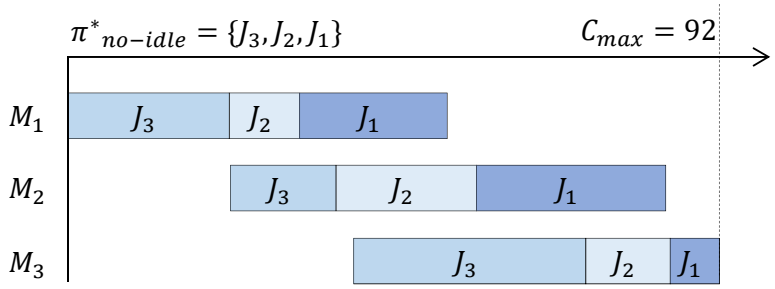

Figure 3: Optimum solution for the no-idle PFSP.

PFSP example presented in Table 1, note that the problem does not yet considers no-idle machines.

Considering a PFSP with makespan criterion, enumerating all possible solutions, the optimal solution is provided by the sequence $\pi^* = \{J_2, J_3, J_1\}$, resulting in a makespan of 88 (Figure 1). On the other hand, if it is considered the no-idle PFSP, the same solution results in a worse non-optimal makespan of 94 (Figure 2). The optimal solution for the same problem when no-idle machines are considered is another one, $\pi^*_{no-idle} = \{J_3, J_2, J_1\}$, with makespan 92 (Figure 3). Thus, the existence of no-idle machines can lead to a different optimal solution.

The small example presented shows that the no-idle machines significantly impact the solutions for the problem. As stated before, the problems studied is an extension of the no-idle PFSP, denoted as mixed-noidle PFSP with sequence-dependent setup times. In the

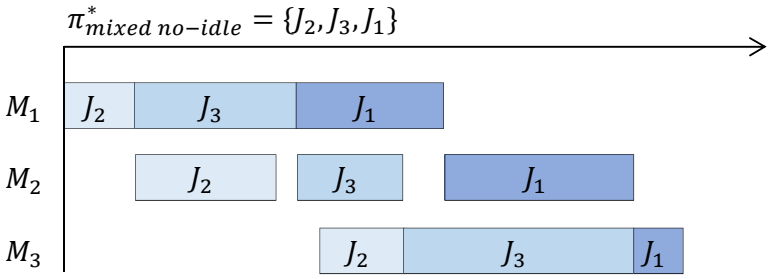

Figure 4: The mixed no-idle PFSP.

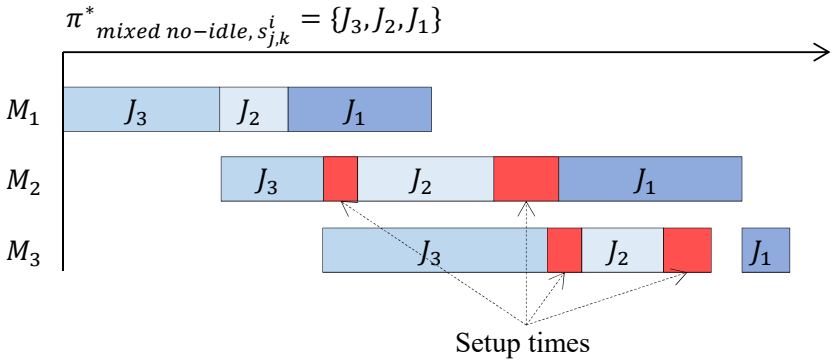

Figure 5: The mixed no-idle PFSP with sequence-dependent setup times.

mixed no-idle PFSP, some machines allow idleness, while the other are no-idle. Figure 4 illustrates the mixed no-idle PFSP under the processing times presented in Table 1, where only machine M_2 allow idleness and M_1 and M_3 are no-idle machines.

As stated before, in the machines that allow idleness, setup times can occur between the jobs. Figure 5 shows a small example where only machine M_1 is no-idle, M_2 and M_3 allow idleness and have setup times between the jobs. In the next section we will present the detailed definition and formalization of the mixed no-idle PFSP with sequence dependent setup times, which is the object of study of this work.

2.3 The mixed no-idle PFSP with sequence dependent setup times

Formally, the problem under consideration can be defined as follows. A set of n jobs $J = \{J_1, \ldots, J_n\}$ must be processed in the same order by a set of m machines $M = \{M_1, \ldots, M_m\}$ with the objective of minimising a certain criterion, namely, in this work, makespan, total flowtime or total tardiness. A processing sequence is defined by $\pi = \{\pi_1, \ldots, \pi_n\}$, π_j or $[j]$ defines a job in the jth position of the sequence. The processing time of job J_j in machine M_i is $p_{i,j}$ ($\forall i = \{1, \ldots, m\}$, $\forall j = \{1, \ldots, n\}$). The completion time of job J_j in machine M_i is called $C_{i,j}$ and the starting date is $S_{i,j}$. The makespan (C_{max} or $C_{max}(\pi)$) is equal to the completion time of the last job in the last machine $C_{m,[n]}$ for the sequence π. The total flowtime is the sum of the completions times of the

jobs in the last machine M_m, $\sum_{j=1}^{n} C_{m,j}$. The total tardiness is the sum of the tardiness of all jobs, $\sum_{j=1}^{n} \max(C_{m,j} - d - j)$, while d_j is the due date of job J_j. Machines that are no-idle belong to a set defined by M'. The machines that do not belong to M' are defined as regular machines, which allow idle and also have a setup time between jobs. The sequence-dependent setup times are considered in the regular machines M_i ($M_i \in M'$) between jobs J_j and J_k ($k \leq n$, $k \neq j$), and are denoted as $s^i_{j,k}$.

According to Graham et al. (1979), a scheduling problem can be defined in a notation comprising three fields $\alpha|\beta|\gamma$, in which α represents the machine environment considered (single machine, parallel machine, flowshop, jobshop or openshop) and the number of machines, β the technological constraints (sequence dependent setup times, unavailability of machines, no-wait, no-idle, mixed no-idle, permutation) and γ is the performance criterion (makespan, tardiness, total flowtime, among others). Thus, the problem considered in this work can be defined as $F_m|prmu, mixed\ no-idle,\ s^i_{j,k}|(C_{max}$, $\sum C_j$, or $\sum T_j)$ where F_m defines that the machine environment is a flowshop with m machines and $prmu$ (permutation) means that the jobs are processed in the same order by all the machines.

2.3.1 Mixed integer linear programming model

We extended the MILP model of Pan e Ruiz (2014) in order to consider the sequence dependent setup times for the regular machines ($M_i \notin M'$). The decision variable is defined by $X_{j,k}$, where $X_{j,k} = 1$ if jobs J_j is the kth job of the sequence π and $X_{j,k} = 0$ otherwise. The relation between the adjacent jobs is controlled by the decision variable $Y_{j,k,l}$, as $Y_{l,k,j} = 1$ if job J_j is in position k in the sequence and is immediately preceded by J_l; otherwise, it equals zero ($\forall l, j \in \{1, \ldots, n\}$, $\forall k \in \{2, \ldots, n\}$). The decision variables are detailed below:

$$X_{j,k} = \begin{cases} 1 & \text{if } J_j \text{ is in position } k \text{ in the sequence} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{l,k,j} = \begin{cases} 1 & \text{if } J_j \text{ is in position } k \text{ in the sequence and} \\ & \text{is preceded by } J_l,\ j \neq l \\ 0 & \text{otherwise} \end{cases}$$

The mixed integer programming model is provided as follows. The objective function, Z, is defined in (2.1) and minimises the makespan, total flowtime or the total tardiness. The set (2.2) and (2.3) state that each job is attributed to one position and vice-versa. Constraints (2.4) and (2.5) ensures that each job precedes only one job and follows exactly one job, and these two jobs are not the same. Constraint (2.6) ensures that there is no precedence for the job in the first position of the sequence, as when $k = 1$ no job precedes J_j, therefore $Y_{l,1,j} = 0$ ($\forall l, j \in \{1, \ldots, n\}$). Constraint (2.7) establish that the completion

time of the job in the first position is greater than or equal to its processing time in the first machine. Constraint (2.8) ensures that a job cannot be processed on a machine before its completion on the preceding machine. In constraints (2.9) and (2.10) the job completion time is equal to the completion time of the previous job plus the processing time if the machine is no-idle. For regular machines, the completion time is greater than or equal to the previous job plus the processing time and setup time between the jobs. Constraints (2.11), (2.12) and (2.13) give the domains of the variables.

$$\text{Minimize } Z = \begin{cases} C_{m,n} & \text{for the makespan criterion} \\ \sum_{j=1}^{n} C_{m,j} & \text{for the total flowtime criterion} \\ \sum_{j=1}^{n} T_j = \max(C_{m,j} - d_j, 0) & \text{for the total tardiness criterion} \end{cases} \tag{2.1}$$

Subject to:

$$\sum_{k=1}^{n} X_{j,k} = 1, \quad \forall j \in \{1, \ldots, n\} \tag{2.2}$$

$$\sum_{j=1}^{n} X_{j,k} = 1, \quad \forall k \in \{1, \ldots, n\} \tag{2.3}$$

$$X_{j,k} = \sum_{l=1}^{n} Y_{l,k,j}, \quad \forall j \in \{1, \ldots, n\}\ \forall k \in \{2, \ldots, n\} \tag{2.4}$$

$$X_{j,k} = \sum_{l=1}^{n} Y_{j,k+1,l}, \quad \forall j \in \{1, \ldots, n\}\ \forall k \in \{1, \ldots, n-1\} \tag{2.5}$$

$$Y_{l,1,j} = 0, \quad \forall j, l \in \{1, \ldots, n\} \tag{2.6}$$

$$C_{1,k} \geq \sum_{j=1}^{n} X_{j,k} \cdot p_{1,j}, \quad \forall k \in \{1, \ldots, n\} \tag{2.7}$$

$$C_{i,k} \geq C_{i-1,k} + \sum_{j=1}^{n} X_{j,k} \cdot p_{i,j} \quad \forall k \in \{1, \ldots, n\},\ \forall i \in \{2, \ldots, m\} \tag{2.8}$$

$$C_{i,k} = C_{i,k-1} + \sum_{j=1}^{n} X_{j,k} \cdot p_{i,j},\ \forall i \in M',\ \forall k \in \{2, \ldots, n\} \tag{2.9}$$

$$C_{i,k} \geq C_{i,k-1} + \sum_{j=1}^{n} X_{j,k} \cdot p_{i,j} + \sum_{j=1}^{n} \sum_{l=1}^{n} Y_{l,k,j} \cdot s^{i}_{l,j},\ \forall i \notin M',\ \forall k \in \{2, \ldots, n\} \tag{2.10}$$

$$C_{i,k} \geq 0 \quad \forall k \in \{1, \ldots, n\},\ \forall i \in \{1, \ldots, m\} \tag{2.11}$$

$$X_{i,k} \in \{0, 1\} \quad \forall k \in \{1, \ldots, n\},\ \forall i \in \{1, \ldots, m\} \tag{2.12}$$

$$Y_{l,k,j} \in \{0, 1\} \quad \forall j, k, l \in \{1, \ldots, n\} \tag{2.13}$$

According to Pan e Ruiz (2014), the mixed no-idle PFSP is $\mathcal{NP}$-Hard in the strong sense. Therefore, as our problem is a special case of the mixed no-idle PFSP where setup times are considered in the regular machines, the new problem studied in this work in also $\mathcal{NP}$-Hard.

2.3.2 Makespan, total flowtime and total tardiness calculation

As the mixed no-idle PFSP with sequence-dependent setup times has not yet been studied in the literature, we present a method to evaluate the makespan of a permutation sequence. Supposing a sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_{l-1}, \pi_l, \ldots, \pi_n\}$ where π_l ($l = \{1, \ldots, n\}$) represents a job in position l of sequence π. To simplify the expressions, the notation $[l]$ denotes the job in the position l of sequence π, i.e. $\pi_l = [l]$. $S_{i,[l]}$ and $C_{i,[l]}$ denote the earliest start and completion time of job $[l]$ in machine M_i, respectively. The $C_{max}(\pi)$ can be obtained adopting the following steps.

Step (1): In order to simplify the calculations, consider that the setup time is equal to zero between the jobs J_j and J_k $\forall j, k \in J, j \neq k$ in a no-idle machine $M_i \in M'$.

$$s^i_{j,k} = 0 \quad \forall j, k \in J, j \neq k \quad \text{if } M_i \in M' \tag{2.14}$$

Step (2): The earliest start and completion times for the machines M_i ($\forall i = \{1, \ldots, m\}$) can be calculated as follows:

Step (2.1): For the jobs ($l = \{1, \ldots, n\}$), calculate the earliest start and completion times in machine M_i without considering if the machine is no-idle.

$$C_{0,[l]} = 0 \tag{2.15}$$

$$S_{i,[1]} = C_{i-1,[1]} \tag{2.16}$$

$$C_{i,[1]} = S_{i,[1]} + p_{i,[1]} \tag{2.17}$$

$$S_{i,[l]} = \max\left(C_{i,[l-1]} + s^i_{[l-1],[l]}, C_{i-1,[l]}\right) \tag{2.18}$$

$$C_{i,[l]} = S_{i,[l]} + p_{i,[l]} \tag{2.19}$$

Step (2.2): The completion time values are recalculated to $l = \{n-1, \ldots, 1\}$, now considering if machine M_i allows or not idleness among the jobs.

$$C_{i,[l]} = \begin{cases} C_{i,[l]} + \max\left(C_{i,[l+1]} - p_{i,[l+1]} - C_{i,[l]}, 0\right) & \text{if machine } M_i \in M' \\ C_{i,[l]} & \text{otherwise} \end{cases} \tag{2.20}$$

Step (2.3): If $i = m$, then calculate the objective function, otherwise return to Step (2.1) and consider the next machine M_{i+1}.

- If the objective is to minimize the makespan, $C_{max}(\pi) = C_{m,[n]}$;
- If the objective is to minimize the total flowtime, $\sum C_j$, $\sum_{j=1}^{n} C_{m,j}$;
- If the objective is to minimize the total tardiness, $\sum T_j$, $\sum_{j=1}^{n} \max(C_{m,j} - d_j, 0)$.

Steps (2.1) and (2.2) are iterated for m machines and have worst case complexity of $O(n)$. As a result, the makespan evaluation procedure has worst case complexity of $O(nm)$.

2.3.3 Acceleration method to calculate the makespan for the insertion neighbourhood

The insertion neighbourhood is extensively used in the PFSP (RUIZ; VALLADA; FERNÁNDEZ-MARTÍNEZ, 2009; RUIZ; STÜTZLE, 2007; RAD; RUIZ; BOROOJERDIAN, 2009). The insertion neighbourhood in a π sequence with n jobs is the result of movements of inserting the jobs into the sequence. An insertion movement consists of removing job π_j from the sequence, and then π_j is reinserted into all positions of π, except for its original position.

The insertion neighbourhood is generated by applying this insertion movement to each job in the sequence. The first to propose acceleration methods for insertion neighbourhood was Taillard (1990) for the PFSP with makespan criterion. Since then, acceleration methods have been extensively used in the literature (RAD; RUIZ; BOROOJERDIAN, 2009; FERNANDEZ-VIAGAS; FRAMINAN, 2014; ROSSI; NAGANO; NETO, 2016), even for other variations of the problem (PAN; WANG, 2008b; PAN; WANG, 2008a; RUIZ; VALLADA; FERNÁNDEZ-MARTÍNEZ, 2009; RIBAS; COMPANYS; TORT-MARTORELL, 2010; TASGETIREN et al., 2013a; INCE et al., 2016; NAGANO; ROSSI; TOMAZELLA, 2017; PAGNOZZI; STÜTZLE, 2017).

Pan e Ruiz (2014) explained and demonstrated an acceleration method to evaluate the makespan of an insertion neighbourhood for the mixed no-idle PFSP. We extended the acceleration procedure for the variant discussed in this study, the mixed no-idle PFSP with a sequence dependent setup.

The speed-up procedure described below can be used to evaluate the makespan of an insertion neighbourhood. $S'_{i,j}$ and $C'_{i,j}$, denote the start and completion time for the job J_j in the machine M_i calculated backwards.

Step (1): Considering a partial sequence $\pi = \{\pi_1, \ldots, \pi_{n-1}\}$. Calculate the earliest start and completion time for sequence π in machines M_i $(i = 1, \ldots, m)$, $S_{i,j}$ and $C_{i,j}$ $(j = \{1, \ldots, n-1\})$ (Section 2.3.2).

Step (2): Calculate the earliest start and completion time backwards in the sequence π in machines M_i $(i = \{m, \ldots, 1\})$, $S'_{i,j}$ and $C'_{i,j}$ $(j = \{n-1, \ldots, 1\})$ (Section 2.3.2).

Step (3): For all the positions in the sequence π $(l = \{1, \ldots, n\})$, go through the following steps: Step (3.1): Insert job J_k in position l of π, resulting in the complete sequence $\pi = \{\pi_1, \ldots, \pi_{k-1}, J_k, \pi_{k+1}, \ldots, \pi_{n-1}\}$.

Step (3.2): Calculate the earliest completion time of job J_k in machine M_i, denoted by $C_{i,[k]}$ using Expressions (2.21)-(2.24). The delay at the beginning of processing the jobs in machine M_i, so that there is no idleness, is defined by a_i.

$$\begin{cases} s^{i}_{j,k} = 0 \quad \forall j,k \in J, j \neq k \quad \text{if } M_i \in M' \end{cases} \tag{2.21}$$

$$\begin{cases} S_{1,[k]} = C_{1,[k-1]} + s^{1}_{[k-1],[k]} \\ C_{1,[k]} = S_{1,[k]} + p_{1,[k]} \end{cases} \tag{2.22}$$

$$\begin{cases} S_{2,[k]} = \max(C_{2,[k-1]} + s^{2}_{[k-1],[k]}, C_{1,[k]}) \\ C_{2,[k]} = S_{2,[k]} + p_{2,[k]} \\ a_2 = \begin{cases} \max(C_{1,[k]} - C_{2,[k-1]}, 0) & \text{if } M_i \in M' \\ 0 & \text{otherwise} \end{cases} \end{cases} \tag{2.23}$$

$$\begin{cases} S_{i,[k]} = \max(C_{i,[k-1]} + a_{i-1} + s^{i}_{[k-1],[k]}, C_{i-1,[k]}) \\ C_{i,[k]} = S_{i,[k]} + p_{i,[k]} \\ a_i = a_{i-1} + \begin{cases} \max(C_{i-1,[k]} - (C_{i,[k-1]} + a_{i-1}), 0) & \text{if } M_i \in M' \\ 0 & \text{otherwise} \end{cases} \\ \forall i = \{3, \ldots, m\} \end{cases} \tag{2.24}$$

Expression (2.21) defines that the setup times are non-existent in the no-idle machines. Expression (2.22) calculates the starting and completion times for the job in position k in the first machine is calculated. Expression (2.23) calculates the start and completion for the second machine, and $\max(C_{1,[k]} - C_{2,[k-1]}, 0)$ defines the right shift delay in the start time for the job in position k if it is a no-idle machine. Expression (2.24) calculates the start and completion time is calculated for the other machines, and $\max(C_{i-1,[k]} - (C_{i,[k-1]} + a_{i-1}), 0)$ is the shift delay.

Step (3.3): The makespan (C_{max}) of the complete sequence $\pi = \{\pi_1, \ldots, \pi_{k-1}, \pi_k, \pi_{k+1}, \ldots, \pi_{n-1}\}$ can be calculated by Expressions (2.25)-(2.29).

$$\begin{cases} s^{i}_{j,k} = 0 \quad \forall j,k \in J, j \neq k \quad \text{if } M_i \in M' \end{cases} \tag{2.25}$$

$$\begin{cases} L_1 = C_{1,[k]} + C'_{1,[k+1]} + s^{1}_{[k],[k+1]} \end{cases} \tag{2.26}$$

$$\begin{cases} L_2 = C_{2,[k]} + C'_{2,[k+1]} + s^{2}_{[k],[k+1]} \\ L = \max\left(L_1, L_2\right) \\ a_2 = \begin{cases} \max\left(L - L_2, 0\right) & \text{if } M_2 \in M' \\ 0 & \text{otherwise} \end{cases} \end{cases} \tag{2.27}$$

$$\begin{cases} L_i = C_{i,[k]} + a_{i-1} + C'_{i,[k+1]} + s^{i}_{[k],[k+1]} \\ L = \max\left(L, L_i\right) \\ a_i = a_{i-1} + \begin{cases} \max\left(L - L_i, 0\right) & \text{if } M_i \in M' \\ 0 & \text{otherwise} \end{cases} \\ \forall i = \{3, \ldots, m\} \end{cases} \tag{2.28}$$

$$C_{max} = L \tag{2.29}$$

Expression (2.26) defines L_1 as the sum of the earliest completion time calculated backwards, Step (2), and forward, Step (3), in the first machine with the setup time between position k and $k+1$ in the first machine. In set (2.27)-(2.28), the value of L_i is calculated for the other machines. Expression (2.29) define the makespan for the sequence. Note that the makespan (C_{max})is the maximum value of L_i ($\forall i = \{1, \ldots, m\}$).

Steps (1) and (2) are followed only once for each insertion of job J_k at the n positions of the π sequence and the complexity is $O\left(nm\right)$. Steps (3.2) and (3.3) have complexity $O(m)$ and are within the iteration of the step (3) which is carried out n times ($l = \{1, \ldots, n\}$). Therefore, evaluating the C_{max} of the insertion of J_k in all the n positions of sequence π results in complexity $O(nm)$. If the evaluation method described in Section 2.3.2 was used, the same insertion procedure would have a complexity $O(n^2m)$.

2.3.4 Acceleration method to calculate the total flowtime or the total tardiness

Pan e Ruiz (2014) proposed an acceleration method to evaluate the makespan in an insertion neighbourhood of the mixed no-idle flowshop problem. In the previous section, this acceleration method was adpated for the problem under consideration with makespan criterion. However, a problem arises when adapting the proposed method directly to the mixed no-idle problem with total flowtime or total tardiness criteria as this method can only determine the completion time of the last job in the machines. Furthermore, the method does not take into consideration the existence of setup times between the jobs.

Considering that in order to evaluate the total flowtime we need to know the completion time of all jobs, and not only the last one, and also considering that there is a setup time between the jobs, it is not possible to directly adapt the acceleration method. However, a partial acceleration method can be developed for the insertion neighbourhood can be proposed considering the following:

The completion time of all jobs in no-idle machine M_i can be determined if the completion time of the last job in machine M_i ($C_{i,[n]}$) and the processing times of the jobs in the same machine ($p_{i,j}$) are known.

Considering $I^i_{k,j}$ as the idle time between jobs J_j and J_k in the regular machine M_i (allows idle times), the completion time of the job in the last position in machine M_i is:

$$C_{i,[n]} = C_{i,[n-1]} + I^i_{[n-1],[n]} + p_{i,[n]} \tag{2.30}$$

On the other hand, if M_i is a no-idle machine, then $I^i_{[n-1],[n]} = 0$, which results in:

$$C_{i,[n]} = C_{i,[n-1]} + p_{i,[n]} \tag{2.31}$$

Otherwise:

$$\begin{aligned} C_{i,[n-1]} &= C_{i,[n]} - p_{i,[n]} \\ C_{i,[n-2]} &= C_{i,[n]} - p_{i,[n]} - p_{i,[n-1]} \\ C_{i,[j]} &= C_{i,[n]} - \sum_{k=j+1}^{n} p_{i,[k]} \end{aligned} \tag{2.32}$$

Supposing that M_h ($1 \leq h \leq m$) is the last no-idle machine of the flowshop, we can apply an acceleration mechanism adapted from Pan e Ruiz (2014) to calculate the completion time of the last job in M_h, which is denoted as $C_{h,[n]}$. Then, calculate recursively the completion times for the remaining jobs. For machines M_{h+1}, M_{h+2}, ..., M_m the procedure of regular calculation (Subsection 2.3.2) can be used to calculate the completion times of the jobs in the last machine M_m.

The procedure below describes the step-by-step procedure of the acceleration method.

Step (1): Considering an incomplete sequence $\pi = \{\pi_1, \ldots, \pi_{n-1}\}$, and that M_h is the last no-idle machine of the flowshop. Calculate the start and completion time from front to back ($S_{i,j}$ and $C_{i,j}$, respectively) in π in machines M_i ($i = 1, ..., h$) using the procedure explained at Subsection 2.3.2 for $j = 1, \ldots, n-1$.

Step (2): Calculate the start and completion times from back to front for sequence π in machines M_i ($i = h, \ldots, 1$), $S'_{i,j}$ and $C'_{i,j}$ ($j = n-1, \ldots, 1$) (Subsection 2.3.2).

Step (3): For all positions of the sequence π ($l = 1, \ldots, n$), go through the following steps:

Step (3.1): Insert job J_k in position l of the sequence π, resulting in $\pi = \{\pi_1, \ldots, \pi_{k-1}, J_k, \pi_{k+1}, \ldots, \pi_{n-1}\}$.

Step (3.2): Calculate the completion time of job J_k in machine M_i, $C_{i,[k]}$. The delay for starting processing the jobs is denoted by a_i.

$$\begin{cases} s^i_{j,k} = 0 \quad \forall j,k \in J, j \neq k \quad \text{if } M_i \in M' \end{cases} \tag{2.33}$$

$$\begin{cases} S_{1,[k]} = C_{1,[k-1]} + s^1_{[k-1],[k]} \\ C_{1,[k]} = S_{1,[k]} + p_{1,[k]} \end{cases} \tag{2.34}$$

$$\begin{cases} S_{2,[k]} = \max(C_{1,[k-1]} + s^1_{[k-1],[k]}, C_{1,[k]}) \\ C_{2,[k]} = S_{2,[k]} + p_{1,[k]} \\ a_2 = \begin{cases} \max(C_{1,[k]} - C_{2,[k-1]}, 0) & \text{if } M_i \in M' \\ 0 & \text{otherwise} \end{cases} \end{cases} \tag{2.35}$$

$$\begin{cases} S_{i,[k]} = \max(C_{i,[k-1]} + a_{i-1} + s^i_{[k-1],[k]}, C_{i-1,[k]}) \\ C_{i,[k]} = S_{i,[k]} + p_{i,[k]} \\ a_i = a_{i-1} + \begin{cases} \max(C_{i-1,[k]} - (C_{i,[k-1]} + a_{i-1}), 0) & \text{if } M_i \in M' \\ 0 & \text{otherwise} \end{cases} \\ i = 3, ..., h \end{cases} \tag{2.36}$$

Step (3.3): Obtain the completion time of the last job in machine M_h, $C_{h,[n]}$, of the complete sequence $\pi = \{\pi_1, \ldots, \pi_{k-1}, \pi_k, \pi_{k+1}, \ldots, \pi_n\}$, through the expressions below.

$$\begin{cases} s^i_{j,k} = 0 \quad \forall j,k \in J, j \neq k \quad \text{if } M_i \in M' \end{cases} \tag{2.37}$$

$$\begin{cases} L_1 = C_{1,[k]} + C'_{1,[k+1]} + s^1_{[k],[k+1]} \end{cases} \tag{2.38}$$

$$\begin{cases} L_2 = C_{2,[k]} + C'_{2,[k+1]} + s^2_{[k],[k+1]} \\ L = \max(L_1, L_2) \\ a_2 = \begin{cases} \max(L - L_2, 0) & \text{if } M_2 \in M' \\ 0 & \text{otherwise} \end{cases} \end{cases} \tag{2.39}$$

$$\begin{cases} L_i = C_{i,[k]} + a_{i-1} + C'_{i,[k+1]} + s^i_{[k],[k+1]} \\ L = \max(L, L_i) \\ a_i = a_{i-1} + \begin{cases} \max(L - L_i, 0) & \text{if } M_i \in M' \\ 0 & \text{otherwise} \end{cases} \\ i = 3, ..., h \end{cases} \tag{2.40}$$

$$C_{h,[n]} = L \tag{2.41}$$

Step (4): Calculate the completion time of jobs ($j = n-1, \ldots, 1$) in the last no-idle machine M_h.

$$C_{h,[j]} = C_{h,[n]} - \sum_{k=j+1}^{n} p_{h,[k]} \tag{2.42}$$

Step (5): Use the regular calculation method (Subsection 2.3.2) to determine the completion time for the remaining machines ($i = h+1, \ldots, m$) and the total flowtime of the sequence, $\sum_{j=1}^{n} C_{m,j}$ or the total tardiness, $\sum_{j=1}^{n} T_j = \max(C_{m,j} - d_j, 0)$.

Steps (1) and (2) have complexity $O(nh)$ and are carried out only once for each insertion of job J_k at the n positions of the sequence. Steps (3.2) and (3.3) have complexity $O(h)$ and are within the iteration of Step (3) which is carried out n times. Step (4) has complexity $O(n)$ and is carried out only once. Step (5) has the complexity of the regular evaluation method of $O(n^2(m-h))$, except for the fact that it is carried out in the last $m-h$ machines. Therefore, to evaluate the total flowtime of the job insertion of J_k in all n positions of the sequence π results in complexity $O(nh + n^2h)$. Thus, the complexity of the method will vary from case to case. If the no-idle machine M_h is the last machine of the M_m system, i.e., $h = m$, the complexity will be the best possible $O(nm)$. In the worst case, if the last no-idle machine M_h is the first machine M_1, $h = 1$, steps (1-4) of the acceleration method will not be used, resulting in the same complexity of the regular evaluation method, i.e. $O(n^2m)$.

In the next chapters, the mixed no-idle PFSP with setup times will be studied under different criteria. The following objective functions will be addressed: makespan, total flowtime and total tardiness.

3 THE MIXED NO-IDLE PFSP WITH SETUP TIMES AND MAKESPAN MIMIZATION

The makespan criterion has attracted the attention in the last decades (ROSSI; NAGANO; NETO, 2016). This is mainly due to the importance that companies give to maximize the utilization of the resources, which is possible when the makespan of a processing sequence is minimized (MACCARTHY; LIU, 1993). To achieve this goal, algorithms are used to generate good solutions in an efficient way.

Recently, constructive heuristics have shown state-of-the-art performance for the PFSP where the aim is to minimize the makespan (RAD; RUIZ; BOROOJERDIAN, 2009; RIBAS; COMPANYS; TORT-MARTORELL, 2010; FERNANDEZ-VIAGAS; FRAMINAN, 2014; ROSSI; NAGANO; NETO, 2016; FERNANDEZ-VIAGAS; RUIZ; FRAMINAN, 2017). Some of these constructive heuristics were successfully modified for the no-idle and setup time problems with great success (RUIZ; VALLADA; FERNÁNDEZ-MARTÍNEZ, 2009; VANCHIPURA; SRIDHARAN; BABU, 2014). Constructive methods are often used as a solution method, as they provide good solutions in a simple and efficient way. Furthermore, since metaheuristics require a procedure that generates initial quality solutions quickly, constructive heuristics are usually used to initiate metaheuristics (FERNANDEZ-VIAGAS; LEISTEN; FRAMINAN, 2016). Most of the constructive heuristics are based on the NEH of Nawaz, Enscore e Ham (1983) and apply the acceleration method from Taillard (1990) in order to efficiently evaluate the makespan and, consequently, reduce the computational complexity. Due to the aforementioned reasons, this work focuses on the proposal of a constructive heuristic for the mixed no-idle PFSP with sequence dependent setup times. In order to increase the efficiency of our proposed method, an acceleration procedure for calculating the makespan of a permutation sequence in the insertion neighbourhood is provided in detail. Moreover, a straightforward and general method to calculate the makespan for the problem is presented. As this problem has not yet been studied in the literature, the main constructive heuristics of the no-idle PFSP and PFSP with sequence dependent setup times were adapted aiming to provide a basis for comparison for the proposed method. Two extensive benchmarks were generated with the objective of comparing the implemented constructive methods using computational and statistical experiments. The results show that the proposed method outperforms the methods adapted from the literature.

This chapter is organised as follows. Section 3.1 presents an extensive literature review of the no-idle PFSP and PFSP with sequence dependent setup times. Section 3.2 proposes the new constructive heuristic. In Section 3.3, the computational and statistical experiments among the constructive heuristics are compared. Finally, Section 3.4 draws the main conclusions of this study.

3.1 Literature review

As the mixed no-idle PFSP with sequence dependent setup times has not yet been studied in the literature, a review was carried out covering papers addressing the no-idle PFSP with makespan (C_{max}) minimization and the PFSP with a sequence dependent flowshop for the makespan criterion.

3.1.1 The no-idle PFSP

Vachajitpan (1982) was the first to study the no-idle PFSP with makespan minimization, also known as $F_m|prmu, no-idle|C_{max}$. In his work, a mixed-integer linear programming (MILP) model and a Branch-and-Bound (B&B) method were proposed. Woollam (1986) was the first to develop heuristics for the problem. Baptiste e Hguny (1997) proposed a B&B method. The authors also proved that the problem is NP-Hard. Bagga (2003), Saadani, Guinet e Moalla (2003) and Kamburowski (2004) studied the three-machine problem, $F_3|prmu, no-idle|C_{max}$. Bagga (2003) presented an MILP model and a B&B procedure for the problem. Saadani, Guinet e Moalla (2003) developed heuristics and a lower bound based on the Johnson (1954) rule. Furthermore, Kamburowski (2004) proposed a representation in the form of networks and presented some paradoxes by which the reduction of processing times can increase the makespan and vice-versa.

Saadani, Guinet e Moalla (2005) proposed a heuristic based on the Traveling Salesman Problem (TSP) for the $F_m|prmu, no-idle|C_{max}$ problem. In another paper Narain e Bagga (2005b) presented four variants of the $F_m|prmu, no-idle|C_{max}$ problem with dominant machines.

Kalczynski e Kamburowski (2005) proposed a heuristic based on the Johnson (1954) rule. Later, the same authors, Kalczynski e Kamburowski (2007a) presented some relations between the $F_m|prmu, no-idle|C_{max}$ e $F_m|prmu, no-wait|C_{max}$ problems.

Baraz e Mosheiov (2008) developed a heuristic comprising two phases, which outperformed the method proposed by Saadani, Guinet e Moalla (2005). In the first stage, all unscheduled jobs are tested in the last position of a partial sequence, and the job resulting in the lowest makespan is attached to the sequence. The first phase is completed when all the n jobs are scheduled. In the second phase, a pairwise job interchange procedure is performed, where all possible pairs of jobs are tested, and only those interchanges that reduce the makespan value are performed.

Pan e Wang (2008b) presented a hybrid discrete particle swarm algorithm (HDPSO). An acceleration mechanism (based on the method proposed by Taillard (1990) was also demonstrated to calculate the makespan for the insertion neighbourhood. The proposed algorithm was compared with the heuristics from Kalczynski e Kamburowski (2005), Baraz e Mosheiov (2008) and Tasgetiren et al. (2007) in the Taillard benchmark Taillard

(1993). The HDPSO method presented significantly better solutions in an acceptable computational time.

Pan e Wang (2008a) developed a discrete differential evolution algorithm (DDE) for the $F_m|prmu, no-idle|C_{max}$ problem. The DDELS algorithm was compared to the methods proposed by Kalczynski e Kamburowski (2005) and Baraz e Mosheiov (2008).

Ruiz, Vallada e Fernández-Martínez (2009) carried out an extensive literature review involving the no-idle PFSP. The authors also presented heuristics and an Iterated Greedy (IG) algorithm based on the work of Ruiz e Stützle (2007). Among the proposed heuristics, the authors highlighted the FRB3 heuristic adapted from Rad, Ruiz e Boroojerdian (2009), as well as the GH-BM2. The GH-BM2 is a modification of the heuristic from Baraz e Mosheiov (2008), where the first phase is substituted by an adaptation of the NEH heuristic, and in the second phase, an insertion neighbourhood is performed instead of the pairwise interchange procedure. The proposed heuristics were compared with the methods by Kalczynski e Kamburowski (2005) and Baraz e Mosheiov (2008). The IG algorithm significantly outperformed the DDE and HDPSO algorithms from Pan e Wang (2008a) and Pan e Wang (2008b) and FRB5 from Rad, Ruiz e Boroojerdian (2009).

Goncharov e Sevastyanov (2009) presented an approximation algorithm with complexity $O(n^2m^2)$ ensuring a theoretical performance. The authors proved that given an instance of the flowshop problem with no-idle constraints, m machines and n jobs, a permutation schedule π can be constructed in $O(n^2m^2)$ time with absolute guarantee that:

$$C_{max}(\pi) - C^*_{max} \leq C_{max}(\pi) - B \leq ((m-1)^2 + 1)p_{max}$$
$$B = \sum_{i=1}^{m-1} (l_i - l_{i+1})^+ + l_m$$
$$l_i = \sum_{j=1}^{n} p_{i,j}$$

where $C_{max}(\pi)$ is the makespan of the permutation π, C^*_{max} is the optimum makespan, B is the lower bound on the optimum makespan, l_i is the machine load, $p_{i,j}$ is the processing time of the job J_j on machine M_i, p_{max} denotes the maximum processing time considering all jobs and machines ($p_{max} = \max_{i,j} p_{i,j}$).

Deng e Gu (2012) developed the Hybrid Discrete Differential Evolution (HDDE) algorithm and an acceleration method. Based on an extensive comparison, it was concluded that the HDDE algorithm performs better than the IG method Rad, Ruiz e Boroojerdian (2009), HDPSO Pan e Wang (2008b) and DDELS Pan e Wang (2008a).

Tasgetiren et al. (2013b) presented an IG algorithm variable, called vIG-DE. The Iterated Greedy (IG) algorithms proposed by Ruiz e Stützle (2007) and Framinan e Leisten (2008) were also reimplemented. The vIG-DE algorithm outperformed the DDE and HDPSO metaheuristics from Pan e Wang (2008a) and Pan e Wang (2008b) and the

HDDE from Deng e Gu (2012).

An Invasive Weed Optimisation algorithm (IWO) was proposed by Zhou, Chen e Zhou (2014). The method outperformed the PSOvns metaheuristics from Tasgetiren et al. (2007) and HDPSO from Pan e Wang (2008b), and the constructive methods from Kalczynski e Kamburowski (2005) and Baraz e Mosheiov (2008).

Shao, Pi e Shao (2017) proposed a memetic algorithm with a hybrid node and edge histogram (MANEH) and outperformed the algorithms from Pan e Wang (2008b), Pan e Wang (2008a), Ruiz, Vallada e Fernández-Martínez (2009), Deng e Gu (2012) and Tasgetiren et al. (2013b) in Ruiz, Vallada e Fernández-Martínez (2009) benchmark.

As stated early, another important variant of the no-idle problem recently studied by Pan e Ruiz (2014) is the mixed no-idle flowshop problem with makespan criterion. The authors presented an MILP model and an IG algorithm. The IG algorithm was compared to the HDPSO algorithms from Pan e Wang (2008b), DDE from Pan e Wang (2008a), HDDE from Deng e Gu (2012) and the HGA from Ruiz, Maroto e Alcaraz (2006). Based on extensive statistical and computational comparisons, it was demonstrated that the proposed IG is statistically better than all the other compared methods.

Despite the significant number of papers addressing the no-idle PFSP problem, the consideration of setup times has not yet been studied in the literature. Table 2 presents a summary of the algorithms proposed for the mixed no-idle PFSP with makespan. In fact, the existence of setup times is incompatible with a pure no-idle environment, since the machines require uninterrupted processing from the moment program processing starts. However, in a mixed no-idle environment Pan e Ruiz (2014), it is possible to have setup times in the productive stages in which machine standstill is allowed.

Table 2: Summary of works addressing the no-idle PFSP.

Year	Reference	NM[a]	Algorithm (*notation*)	Outperformed by
1982	Vachajitpan (1982)	m	MILP, B&B	
1986	Woollam (1986)	m	CH (*CAMP2, NAWAZ*)	
1997	Baptiste e Hguny (1997)	m	B&B	
2003	Bagga (2003)	3	MILP, B&B	
2003	Saadani, Guinet e Moalla (2003)	3	CH (*H*)	
2005	Saadani, Guinet e Moalla (2005)	m	CH	Baraz e Mosheiov (2008); Ruiz, Vallada e Fernández-Martínez (2009)
2005	Kalczynski e Kamburowski (2005)	m	CH (*KK*)	Pan e Wang (2008a); Ruiz, Vallada e Fernández-Martínez (2009); Zhou, Chen e Zhou (2014)
2008	Baraz e Mosheiov (2008)	m	CH (*Improved Greedy - IG*[a])	Ruiz, Vallada e Fernández-Martínez (2009); Zhou, Chen e Zhou (2014)
2008	Pan e Wang (2008b)	m	Hybrid PSO (*HDPSO*)	Ruiz, Vallada e Fernández-Martínez (2009); Deng e Gu (2012); Tasgetiren et al. (2013b); Zhou, Chen e Zhou (2014); Shao, Pi e Shao (2017)
2008	Pan e Wang (2008a)	m	DE (*DDE*)	Ruiz, Vallada e Fernández-Martínez (2009); Deng e Gu (2012); Tasgetiren et al. (2013b); Zhou, Chen e Zhou (2014); Shao, Pi e Shao (2017)
2009	Ruiz, Vallada e Fernández-Martínez (2009)	m	CH (*GH-BM2, FRB3, FRB4*) IG (IG_{LS})	Deng e Gu (2012); Tasgetiren et al. (2013b); Shao, Pi e Shao (2017)
2009	Goncharov e Sevastyanov (2009)	m	AA	
2012	Deng e Gu (2012)	m	Hybrid DE (*HDDE*)	Tasgetiren et al. (2013b); Zhou, Chen e Zhou (2014); Shao, Pi e Shao (2017)
2013	Tasgetiren et al. (2013b)	m	IG with DE (*vIG-DE*)	Shao, Pi e Shao (2017)
2014	Zhou, Chen e Zhou (2014)	m	IWO (*IWO*)	
2017	Shao, Pi e Shao (2017)	m	ME (*MANEH*)	

Notation: MILP, Mixed-integer Linear Programming model; B&B, Branch-and-Bound; CH, Constructive Heuristic; PSO, Particle Swarm Optimization Algorithms; DE, Differential Evolution Algorithm; IG, Iterated Greedy Algorithm; AA, Approximate Algorithms; IWO, Invasive Weed Optimization Algorithm; ME, Memetic Algorithm.

[a] Number of machines. [b] Also known as GH-BM in Ruiz, Vallada e Fernández-Martínez (2009).

3.1.2 The PFSP with sequence dependent setup times

Corwin e Esogbue (1974) were one of the first to address the PFSP with sequence dependent setup times. The authors proposed a dynamic programming formulation for the two-machine problem ($F_2|prmu, s^i_{j,k}|C_{max}$) where the setup times are sequence dependent on the first machine and sequence-independent on the second machine. For the same problem, Uskup e Smith (1975) developed a B&B algorithm. Considering the two-machine problem, Gupta e Darrow (1986) presented approximate algorithms. The experimental results concluded that the proposed algorithms presented good solutions when compared to the optimal solution for the problems.

Various exact methods were proposed for the m machine problem, $F_m|prmu, s^i_{j,k}|C_{max}$. Srikar e Ghosh (1986) developed an MILP model. Stafford Jr e Tseng (1990) improved the MILP from Srikar e Ghosh (1986) with three new MILP models. B&B and Branch-and-Cut (B&C) algorithms were proposed by Ríos-Mercado e Bard (1998a) and Ríos-Mercado e Bard (1999). Stafford Jr e Tseng (2002) developed two MILP models, which are based on the work of Tseng e Stafford Jr (2001). Ríos-Mercado e Bard (2003) also formulated two MILP models. The first is related to the Asymmetric Traveling Salesman Problem (ATSP). The second is derived from a model proposed by Srikar e Ghosh (1986). The two models were compared using a branch-and-cut algorithm, which showed that the approach related to the ATSP was outperformed in terms of the computational time.

Simons Jr (1992) were the first to propose heuristics for the flowshop sequence dependent setup times. Two heuristics were presented, TOTAL and SETUP, for the $F_m|prmu, s^i_{j,k}|C_{max}$. In the case of TOTAL, a distance matrix between pairs of jobs is obtained considering the sum of the processing times of the jobs and the setup times in all the machines, whereas in the SETUP only the setup times are considered. The heuristics outperformed the MINCOT and MINIT methods adapted from (GUPTA, 1972).

Das, Gupta e Khumawala (1995) presented a savings index heuristic algorithm, denoted as SI, to find a minimum or approximately minimum makespan of a sequence. Using the same approach, Tseng, Gupta e Stafford Jr (2006) developed a penalty-based heuristic algorithm to find an approximate minimum makespan schedule. The method was compared to the work of Das, Gupta e Khumawala (1995) and showed better results.

Ríos-Mercado e Bard (1998b) proposed a greedy randomized adaptive search procedure (GRASP) based on insertion movements and an adaptation of the NEH constructive heuristic of Nawaz, Enscore e Ham (1983), called NEHT-RB, for the $F_m|prmu, s^i_{j,k}|C_{max}$. The NEHT-RB also used the Taillard (1990) makespan acceleration in the construction phase of the NEH heuristic. The NEHT-RB and GRASP methods presented better results than the SETUP Simons Jr (1992), although the GRASP is considerably slower than both methods. Later on, Ríos-Mercado et al. (1999) proposed an improvement of the GRASP

denoted as HYBRID.

Ruiz, Maroto e Alcaraz (2005) proposed two Genetic Algorithms (GA), and showed that the obtained results outperform the methods from the literature NEHT-RB of Ríos-Mercado e Bard (1998b), GRASP of Ríos-Mercado et al. (1999), SETUP and TOTAL of (Simons Jr, 1992) and SI of Das, Gupta e Khumawala (1995). The methods were compared in an extensive benchmark based on the instances of Taillard (1993).

Gajpal, Rajendran e Ziegler (2006) developed an Ant Colony algorithm, which obtained better results, as compared to those solutions given by the ant colony algorithm of Stützle (1997), called the MMAS (Max-Min Ant System), and the GRASP heuristic from Ríos-Mercado e Bard (1998b).

Ruiz e Stützle (2008) presented two Iterated Greedy algorithms (IG) for the $F_m|prmu, s^i_{j,k}|C_{max}$. The methods performed better than those of Ruiz, Maroto e Alcaraz (2005) in the benchmark proposed by Ruiz, Maroto e Alcaraz (2005).

Vanchipura e Sridharan (2013) proposed a constructive heuristic, denoted as FJSRA (Fictitious Job Setup Ranking Algorithm), for the $F_m|prmu, s^i_{j,k}|C_{max}$. The heuristic is based on the formation of fictional jobs and the construction of the final sequence using these fictitious jobs applying the NEHT-RB construction procedure. Initially the heuristic calculates for each pair of jobs j and k the sum of the setup times between the jobs ($sst_{j,k} = \sum_{i=1}^{m} s^i_{j,k}$). Then, the pairs of jobs are ordered in non-descending order of $sst_{j,k}$. The $n/2$ pairs of jobs that have the smallest value of $sst_{j,k}$ are considered as fictitious jobs. For each fictitious job the total processing time is calculated. Then, the heuristic uses the NEHT-RB heuristic to construct the final sequence, inserting at each iteration the fictitious jobs is in all the possible positions of the partial sequence and choosing the position that minimizes the makespan. The procedure continues until $n/2$ fictitious jobs have been sequenced.

In Vanchipura, Sridharan e Babu (2014), a neighbourhood search known as variable neighbourhood descent (VND) was used to improve the FJSRA constructive heuristic. The new method was called FJSRA-VND. The VND method uses insertion movements to improve the solutions obtained from the FJSRA.

More recently, several metaheuristics were proposed for the $F_m\ |prmu,\ s^i_{j,k}|\ C_{max}$. Mirabi (2011) proposes an Ant Colony Optimization (ACO) algorithm and showed better results than the algorithm from Ruiz, Maroto e Alcaraz (2005). Mirabi (2014) developed a Hybrid Genetic Algorithm (HGA) and compared it to the fuzzy algorithm adapted from Sheibani (2010), which was originally proposed for the $F_m|prmu|C_{max}$. Wang et al. (2014) presented Iterated Local Search (ILS) algorithms for the $F_m|prmu, s^i_{j,k}|C_{max}$, which resulted in better solutions when compared to the method from Ruiz e Stützle (2008). Benkalai et al. (2017) proposed an enhanced migrating birds optimization (EMBO) metaheuristic

and the results were compared to their MBO from their previous work (BENKALAI et al., 2016). Sioud e Gagné (2018) also presented a migrating birds optimization (MBO) metaheuristic and compared the results with the FJSRA-VND method proposed by Vanchipura, Sridharan e Babu (2014), HGA by Mirabi (2014) and the adapted algorithms from Pan e Ruiz (2012), which was originally proposed for the $F_m|prmu|\sum C_j$. The works from Sioud e Gagné (2018) and Benkalai et al. (2017) still need to be compared.

As shown in the literature review, at the moment the no-idle and mixed no-idle conditions have not yet been considered for the sequence dependent flowshop problem in the literature. Although several metaheuristics were proposed for the problem, only a few constructive heuristics were presented. The main constructive methods are the NEHT-RB, FJSRA and FJSRA-VND. Moreover, the FJSRA and FJSRA-VND heuristics perform better than the NEHT-RB heuristic. Table 3 shows a summary of algorithms proposed for the problem.

Table 3: Summary of works addressing the PFSP with sequence dependent setup times.

Year	Reference	NM[a]	Algorithm (*notation*)	Outperformed by
1974	Corwin e Esogbue (1974)	2	DP	
1975	Uskup e Smith (1975)	2	B&B	
1986	Gupta e Darrow (1986)	2	AA	
1986	Srikar e Ghosh (1986)	m	MILP	Stafford Jr e Tseng (1990)
1990	Stafford Jr e Tseng (1990)	m	MILP	
1992	Simons Jr (1992)	m	CH (*SETUP, TOTAL*)	Ruiz, Maroto e Alcaraz (2005)
1995	Das, Gupta e Khumawala (1995)	m	SI	Ruiz, Maroto e Alcaraz (2005); Gajpal, Rajendran e Ziegler (2006)
1998	Ríos-Mercado e Bard (1998a)	m	B&B, B&C	
1998	Ríos-Mercado e Bard (1998b)	m	CH (*NEHT-RB*)	Ríos-Mercado et al. (1999); Ruiz, Maroto e Alcaraz (2005);
			IH (*GRASP*	Gajpal, Rajendran e Ziegler (2006); Vanchipura e Sridharan (2013);
				Vanchipura, Sridharan e Babu (2014)
1999	Ríos-Mercado et al. (1999)	m	IH (*HYBRID*)	
1999	Ríos-Mercado e Bard (1999)	m	B&B, B&C	
2002	Stafford Jr e Tseng (2002)	m	MILP	
2003	Ríos-Mercado e Bard (2003)	m	MILP	
2005	Ruiz, Maroto e Alcaraz (2005)	m	Hybrid GA (*HGA*)	Ruiz e Stützle (2008); Mirabi (2011)
2006	Gajpal, Rajendran e Ziegler (2006)	m	ACO (*PACA*)	
2008	Ruiz e Stützle (2008)	m	IG (*IG_RS*, IG_RS_{LS})	Wang et al. (2014)
2011	Mirabi (2011)	m	ACO (*ACO*)	
2013	Vanchipura e Sridharan (2013)	m	CH (*FJSRA*)	Vanchipura, Sridharan e Babu (2014)
2014	Vanchipura, Sridharan e Babu (2014)	m	CH (*FJSRA-VND*)	Sioud e Gagné (2018)
2014	Mirabi (2014)	m	Hybrid GA (*HGA*)	Sioud e Gagné (2018)
2014	Wang et al. (2014)	m	ILS (*MRSILS*)	
2017	Benkalai et al. (2017)	m	MBO (*MBO*)	
2018	Sioud e Gagné (2018)	m	MBO (*EMBO*)	

Notation: DP, Dynamic Programming Formulation; MILP, Mixed-integer Linear Programming model; B&B, Branch-and-Bound; B&C, Branch-and-Cut; AA, Approximate Algorithms; SI, Saving Index Algorithm; GA, Genetic Algorithms; CH, Constructive Heuristic; IH, Improvement Heuristic; ACO, Ant Colony Optimization Algorithms; IG, Iterated Greedy Algorithm; ILS, Iterated Local Search; MBO, Migration Bird Optimization Algorithms.

[a] Number of machines.

3.2 A new constructive heuristic

Constructive heuristics are broadly used in scheduling problems to efficiently obtain good quality solutions (LIU; REEVES, 2001; NAGANO; MOCCELLIN, 2002; FRAMINAN; LEISTEN; RUIZ-USANO, 2002; FRAMINAN; LEISTEN, 2003; DONG; HUANG; CHEN, 2008; RAD; RUIZ; BOROOJERDIAN, 2009; KALCZYNSKI; KAMBUROWSKI, 2009; LAHA; SARIN, 2009; RIBAS; COMPANYS; TORT-MARTORELL, 2010; PAN; RUIZ, 2013; FERNANDEZ-VIAGAS; FRAMINAN, 2014; FERNANDEZ-VIAGAS; FRAMINAN, 2015; BENAVIDES; RITT, 2016; FERNANDEZ-VIAGAS; LEISTEN; FRAMINAN, 2016; RIBAS; COMPANYS; TORT-MARTORELL, 2017; LIU; JIN; PRICE, 2017; HUANG et al., 2017).

The constructive heuristic proposed in this work was called RN. The heuristic inserts d jobs using a greedy heuristic, and then the rest of the $n - d$ are inserted into the sequence using an NEH heuristic variant.

3.2.1 Greedy heuristic

Recently, greedy heuristics have been extensively used in the literature (LIU; REEVES, 2001; PAN; WANG, 2012; PAN; RUIZ, 2013; FERNANDEZ-VIAGAS; FRAMINAN, 2015; ROSSI; NAGANO; SAGAWA, 2017; PESSOA; ANDRADE, 2017; NAGANO; ROSSI; TOMAZELLA, 2017; NAGANO; ROSSI; MARTARELLI, 2018). This type of heuristic appends the jobs iteratively to the last position of the sequence according to an index based on the properties of the problem. At each iteration, the job that obtained the lowest or highest index value is chosen to be inserted in the last position of the sequence. This procedure is repeated until n jobs have been sequenced.

As the object of study of this work is the mixed no-idle flowshop problem with sequence dependent setup, the index used in our proposal takes into account: (a) idle time between the jobs, which in this case is allowed in regular machines, (b) the setup times between jobs in regular machines. Thus, the index was developed in order to consider these two properties of the problem.

Suppose a partial sequence with k jobs $\pi = \{\pi_1, \ldots, \pi_k\}$. J_j is the candidate job to be tested in position $k+1$ in the sequence π and U is the set of unscheduled jobs ($U \notin \pi$). The index of the candidate job J_j in a sequence with k jobs is denoted by $\delta_{k,j}$. The index has two components in its formulation.

The first component $I_{k,j}$ evaluates the impact on idle times of the insertion of job J_j in the last position of sequence π, i.e. $\pi_{k+1} = J_j$, resulting in the partial sequence $\pi' = \{\pi_1, \ldots, \pi_k, \pi_{k+1}\}$. In order to improve the speed of the method and also make it possible to be used in environments where all machines are no-idle, the index considers that the idle times between jobs π_k and π_{k+1} are allowed in all machines. However,

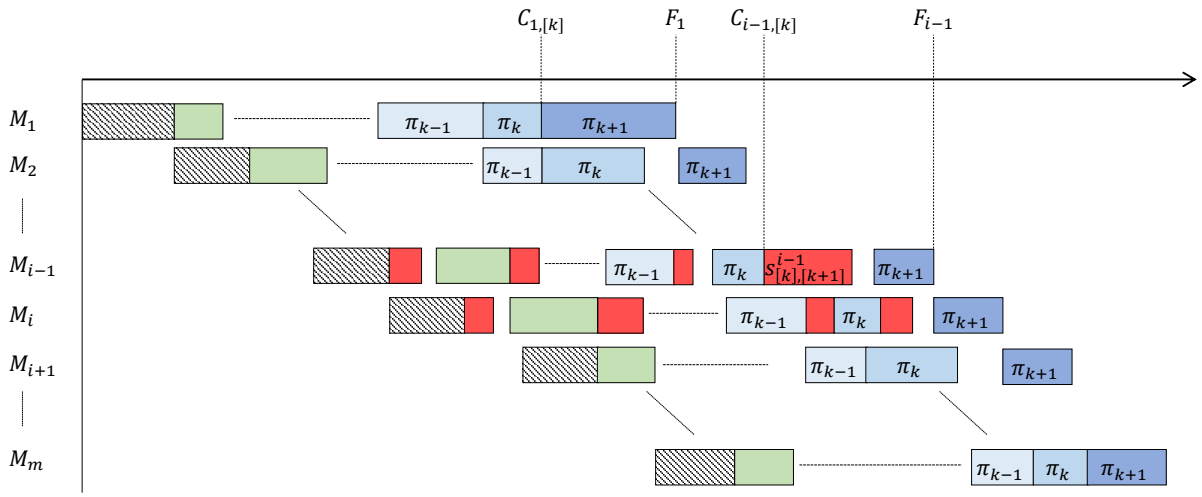

Figure 6: Earliest completion time for the candidate job $\pi_{[k+1]}$ in machine M_i.

among the jobs already sequenced $(\pi_1, \pi_2, \ldots, \pi_{[k]})$ the normal condition of the problem is considered where there is a mix between regular and no-idle machines. Therefore, the earliest completion times in machine M_i $(i = \{1, \ldots, m\})$ for job π_{k+1}, called F_i (Figure 6), can be calculated using Expressions (3.1) and (3.2). In order to simplify the expressions, consider $s^i_{j,k} = 0\ \forall j, k \in J, j \neq k$ in the no-idle machines $(M_i \in M')$.

$$F_1 = C_{1,[k]} + p_{1,[k+1]} + s^1_{[k],[k+1]} \tag{3.1}$$

$$F_i = \max\left(C_{i,[k]} + s^i_{[k],[k+1]}, F_{i-1}\right) + p_{i,[k+1]} \quad \forall i = \{2, \ldots, m\} \tag{3.2}$$

Completion time $C_{i,[k]}$ can be calculated using the method described in Section 2.3.2. Based on the calculated earliest completion times, F_i, it is possible to obtain the value of $I_{k,j}$ using Expression (3.3) can be obtained.

$$I_{k,j} = \sum_{i=2}^{m} \max\left(F_i - p_{i,[k+1]} - C_{i,[k]} - s^i_{[k],[k+1]}, 0\right) \quad \forall j \in U \tag{3.3}$$

The second component is denoted by $SSD_{k,j}$ and is the sum of the setup times between jobs π_k and π_{k+1} in machines m, Expression (3.4).

$$SSD_{k,j} = \sum_{i=1}^{m} s^i_{[k],[k+1]} \quad \forall j \in U \tag{3.4}$$

To select the jobs to be inserted in the first position of the sequence $(k = 1)$, index $\delta_{k,j}$ takes into account only the earliest start time of the candidate jobs in the machines, $S_{i,[k]}$. Finally, the index value $\delta_{k,j}$ can be calculated by Expression (3.5).

$$\delta_{k,j} = \begin{cases} \sum_{i=2}^{m} s_{i,[k]} & \forall j \in U \quad \text{if } k = 1 \\ I_{k,j} + SSD_{k,j} & \forall j \in U \quad \text{if } k \neq 1 \end{cases} \tag{3.5}$$

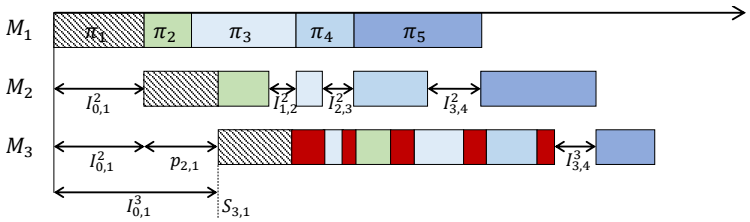

Figure 7: Idles times and starting times for the jobs in a permutation flowshop without no-idle machines.

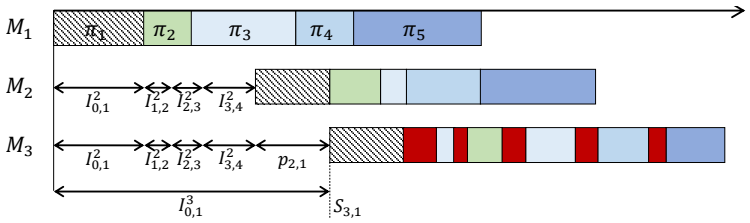

Figure 8: Idles times and starting times for the jobs in a permutation flowshop with no-idle machines.

At each iteration, the candidate job chosen to be appended is the one that presents the lowest index value $\delta_{k,j}$. Thus, index $\delta_{k,j}$ favours the job that results in the lowest values of $I_{k,j}$ and $SSD_{k,j}$. The principle behind this method is that if the idle time is minimised between jobs π_k and π_{k+1}, the makespan will also be reduced due to better utilisation of the machines.

For example, in Figure 7 we have a small example with three machines and five jobs, where all machines allow idle and only M_3 have setup times between the jobs. Considering $I^i_{k,j}$ as the idle time between jobs J_j and J_k in the machine M_i $i = \{1, \ldots, m\}$, the makespan can be obtained as follows:

$$C_{max} = \sum_{j=1}^{n} p_{3,j} + \sum_{j=0}^{n} I^3_{j,j+1} \tag{3.6}$$

If we consider machines M_1 and M_2 as no-idle, we have the situation demonstrated in Figure 8. Note that the idle times in machine M_2 results in the delay for the starting time of machines M_2 and M_3. This time, the makespan can be calculated as follows:

$$C_{max} = \sum_{j=1}^{n} p_{3,j} + \sum_{j=0}^{n} I^3_{j,j+1} \tag{3.7}$$

$$\sum_{j=0}^{n} I^3_{j,j+1} = \sum_{j=0}^{n} I^2_{j,j+1} + p_{2,1} \tag{3.8}$$

From the expressions above, we can see that minimizing the idle times between the jobs, the makespan of the sequence can be improved.

For the same reason, it is also interesting that the setup times between jobs π_k and π_{k+1} in the machines are minimised. Our index is partially based on the Shortest Setup Time (SST) first rule, which is often used when setup times are involved. This rule implies that whenever a job is completed, the job with the smallest setup time is selected to go next. This SST rule is equivalent to the Nearest Neighbour rule for the Travelling Salesman Problem (TSP). The SST rule is known to lead to reasonable schedules (PINEDO, 2016). After inserting d jobs using the greedy method, the heuristic RN uses an NEH heuristic variant to construct the rest of the sequence.

3.2.2 NEH heuristic variant

The NEH heuristic from Nawaz, Enscore e Ham (1983) was originally proposed for the PFSP with the makespan criterion. The heuristic has two phases. In the first phase, the NEH orders the jobs in non-ascending order of the sum of their processing times, also known as the LPT (Longest Processing Time) rule. In the second phase, a sequence is constructed by evaluating partial sequences using the jobs from the initial order provided by the first phase. Suppose a sequence already determined for the $k-1$ first jobs, k partial sequences are obtained by inserting job k into the k possible positions of the current sequence. From these k generated partial sequences, the one with the lowest makespan is maintained as the current sequence for the k first jobs of the first phase ordering. Afterwards, the job in position $k+1$ in the first phase is considered analogously, and so on until the n jobs have been sequenced.

The NEH variant used in the RN heuristic has two main modifications. The first is to replace the LPT rule by the prioritisation of the jobs that have a greater standard deviation (STD) of the processing times in the machines. As demonstrated in Dong, Huang e Chen (2008), for problems with makespan criterion, better solutions can be obtained when this initial ordering is used instead of the LPT.

The second important modification is to carry out a small local search using the insertion neighbourhood in the partial sequence generated at the end of each NEH insertion procedure. Therefore, after a job from the initial order is inserted into the sequence by the NEH insertion, reinsertion movements of jobs in the partial sequence are performed. In this movement, an adjacent job pair is removed, π_k and π_{k+1} ($k = \{1, \ldots, n-1\}$), from the current sequence, and then this pair is reinserted into the partial sequence. The first job of the pair π_k is evaluated in all positions, choosing the position that provides the least C_{max}, and afterwards the second job π_{k+1} is considered analogously. This type of neighbourhood search when performed within the NEH heuristic iterations allows a better optimisation of the partial sequences, which results in better final solutions (LAHA; SARIN, 2009; RAD; RUIZ; BOROOJERDIAN, 2009; ROSSI; NAGANO; NETO, 2016).

However, if we are to consider all pairs of adjacent jobs possible for reinsertion,

the method becomes computationally costly. Considering the acceleration mechanism proposed in Section 2.3.3, the computational complexity of the NEH heuristic is $O(n^2m)$. In the worst case scenario, we will have $n-1$ pairs of possible adjacent jobs. Therefore, performing the reinsertion procedure for each of the pairs after each NEH iteration results in a computational complexity of $O(n^3m)$, one more order of complexity. To solve this problem, we limit the number of jobs to be inserted at each iteration. In order to do this, we resorted to the proposal of the FRB4$_x$ heuristic from Rad, Ruiz e Boroojerdian (2009) to insert only the x jobs that are positioned around the job just inserted by the NEH. This intelligent limitation makes the reinsertion mechanism more efficient while maintaining partial sequence optimisation. For this reason, our proposed method can be denoted as RN$_x$, where x limits the number of jobs selected for reinsertion.

To the best of our knowledge, this is the first time that a limited local search based on reinserting pairs of jobs has been proposed as an extension of the NEH heuristic. The FRB3 and FRB4$_x$ (RAD; RUIZ; BOROOJERDIAN, 2009) applied a local search based on insertion movements where only one job is removed from the sequence and tested in all positions. As mentioned before, FRB4$_x$ limits the number of reinserted jobs, while FRB3 reinserts all jobs from the partial sequence. More recently, the heuristic from Rossi, Nagano e Neto (2016) used a local search based on reinserting pairs of jobs. However, in this heuristic all adjacent pairs of jobs are reinserted in the sequence, which results in a costly computational procedure. To counter this inefficiency, the RN$_\text{x}$ heuristic limits the number of pairs of jobs selected for reinsertion, while keeping the optimization of partial sequences generated by the NEH.

3.2.3 The RN$_x$ heuristic

Combining the greedy method with the NEH heuristic variant, we have the proposed heuristic called RN$_\text{x}$. Algorithm 2 presents the pseudocode of the RN$_\text{x}$ heuristic. Note that the RN$_\text{x}$ has two parameters. The first, called d, controls how many jobs will be inserted using the greedy heuristic (Section 3.2.1). The second, denoted by x, controls how many jobs will be reinserted after each iteration of the NEH heuristic variant (Section 3.2.2).

Algorithm 1 shows that initially d jobs are inserted from the unscheduled set of jobs $U = \{J_1, \ldots, J_n\}$, choosing the jobs based on the lowest value of the index $\delta_{k,j}$ (Section 3.2.1). When the chosen job is inserted in the last position of the sequence π, the same job is removed from the set U ($U = U - J_j$). After d jobs have been scheduled, the heuristic inserts the $n-d$ remaining jobs with the NEH variant. Firstly, the jobs from U are ordered by the non-ascending order of the standard deviation of the processing times (STD_j), generating a list of jobs $\alpha = \{\alpha_1, \ldots, \alpha_{n-d}\}$. The first job from list α (α_1) is tested in all positions of π and is inserted in the best position, denoted as b. The jobs in the x positions around b are selected to be reinserted using a local search based on the reinsertion of pairs

Algorithm 1 RN_x heuristic

$\pi = \emptyset$
$U = \{J_1, \ldots, J_n\}$
for $k = 1$ to d **do**
  Select the job $J_j \in U$ with the lowest $\delta_{k,j}$ (Equation 3.5)
  Place it at the end of π.
  $U = U - J_j$
end for
Order the jobs in U according to the non-ascending order of STD_j, generating $\alpha = \{\alpha_1, \ldots, \alpha_{n-d}\}$.
for $l = 1$ to $n - d$ **do**
  Insert job α_l in π in the position b that results in the lowest C_{max}
  for $k = \max(1, b - x)$ to $\min(l, b + x)$, step $k = k + 2$ **do**
    $\pi' = \pi$
    Remove the jobs π'_k and π'_{k+1} from π'.
    Insert the job π'_k in the position of that results in the lowest C_{max}.
    Insert the job π'_{k+1} in the position of that results in the lowest C_{max}.
    if $C_{max}(\pi') < C_{max}(\pi)$ **then**
      $\pi = \pi'$
    end if
  end for
end for

of jobs. For example, when the pair of jobs π_k and π_{k+1} $(x - b \leq k < x + b)$ is chosen for reinsertion, both are removed from the sequence, then the first job π_k is reinserted in the best position, and the next job of the pair π_{k+1} is also reinserted. The same procedure is applied for the next pair π_{k+2} and π_{k+3}, and so on until all pairs around the position b are reinserted. The next job from the list α (α_2) is considered analogously. The algorithm continues until the sequence is completed with n scheduled jobs.

3.3 Computational and statistical experiments

3.3.1 Instances generation

For a more careful analysis of our method, we generated four different computational experiments: the parameter tuning for the RN_x, the computational evaluation with a benchmark adapted from Pan e Ruiz (2014), the comparison of the same heuristics in a set of problems from the setup times literature from Ruiz, Maroto e Alcaraz (2005), and finally the evaluation of the MILP model. For each computational experiment we generated a different benchmark. The instances and results used in our computational experiments are available as supplementary material.

3.3.1.1 Benchmark for the parameter tuning for the RN_x

In order to avoid an overfitting of parameter d of our proposed RN_x heuristic we calibrated it in a different set of problems to those used for comparisons with the heuristics from the literature. The benchmark is generated with combinations between a number of jobs $n = \{60, 120, 240, 360\}$ and a number of machines $m = \{20, 40\}$, totalling $4 \times 2 = 8$ possible combinations. Five replications were generated for each combination, resulting in $8 \times 5 = 40$ problems per combination. We considered the mixed-idle scenarios used in a benchmark proposed by Pan e Ruiz (2014), except for the scenario where all machines are no-idle and no setup time is allowed:

- Group 1: The first 50% of the machines are no-idle, the rest are regular machines.
- Group 2: The first 50% of the machines are regular, the rest are no-idle machines.
- Group 3: The machines alternate between regular and no-idle.
- Group 4: 25% of the machines are randomly no-idle.
- Group 5: 50% of the machines are randomly no-idle.
- Group 6: 75% of the machines are randomly no-idle.

The processing times were generated randomly with a uniform distribution [1,99]. As we are addressing the mixed no-idle PFSP with dependent-sequence setup times, we consider the setup times in the machines that allow idleness. The setup times were generated with three different distributions:

- SSD-40: the distribution of setup times is limited to 40% of the limit for the processing time interval. That is, the setup times were generated according to a uniform distribution in the interval [1, 39].
- SSD-80: the distribution of setup times is limited to 80% of the limit for the processing time interval. That is, the setup times were generated according to a uniform distribution in the interval [1, 79].
- SSD-120: the distribution of setup times is limited to 120% of the limit for the processing time interval. That is, the setup times were generated according to a uniform distribution in the interval [1, 119].

For each of the three setup time distribution, the different mixed no-idle scenarios from Pan e Ruiz (2014) were considered. With 40 problem instances per combination, the benchmark results in a total of $40 \times 3 \times 6 = 720$ instances plus 40 problems for the parameter tuning benchmark.

3.3.1.2 Benchmark adapted from Pan e Ruiz (2014)

As stated before, Pan e Ruiz (2014) proposed a benchmark for the mixed no-idle PFSP. In more detail, seven groups of instances were generated with different mixed no-idle scenarios, varying the number of no-idle machines in the environment. Each one of the groups contains problems with combinations between a number of jobs $n = \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$ and a number of machines $m = \{10, 20, 30, 40, 50\}$, totalling $10 \times 5 = 50$ possible combinations. Five replications were generated for each combination, resulting in a total of $50 \times 5 = 250$ problems per group. As there are seven groups, in total the set has $7 \times 250 = 1750$ problems. Pan e Ruiz (2014) used a uniform distribution [1,99] to generate the processing times. The benchmark can be found at http://soa.iti.es/problem-instances.

In this work, we address the mixed no-idle flowshop problem with the additional condition of sequence dependent setup times on regular machines. Therefore, the benchmark problems proposed by Pan e Ruiz (2014) were extended to our problem. We considered the same scenarios considered in Pan e Ruiz (2014), with exception of the scenario where all machines are no-idle and no setup time is allowed. The pure no-idle scenario was not considered as FJSRA and FJSRA-VND need setup times to generate the fictitious jobs using the setup ranking algorithm (SRA). Thus, a total of six scenarios were used, the same described in the previous section (Group 1-6). We used the same processing times generated by Pan e Ruiz (2014) and added sequence dependent setup times to regular machines with three different distributions:

- SSD-50: setup times with uniform distribution in the interval [1, 49] (limited to 50% of the limit for the processing time interval).
- SSD-100: setup times with uniform distribution in the interval [1, 99] (limited to 100% of the limit for the processing time interval).
- SSD-125: setup times with uniform distribution in the interval [1, 124] (limited to 125% of the limit for the processing time interval).

Therefore, the new benchmark adapted from Pan e Ruiz (2014) for the mixed no-idle PFSP consists of three sequence dependent setup times of distribution intervals (SSD-50, SSD-100, SSD125) and six mixed no-idle scenarios. Thus, the total number of tests for the benchmark is $6 \times 3 \times 250 = 4500$ instances. In order to evaluate the performance of heuristics without the setup times, we also compared the heuristics in the original benchmark from Pan e Ruiz (2014), which does not consider the dependent setup times in the machines. In this case, only the mixed no-idle condition was considered.

3.3.1.3 Benchmark adapted from Ruiz, Maroto e Alcaraz (2005)

As our also problem deals with setup times, to generate a more comprehensive comparison we also compared the heuristics in a benchmark the literature on sequence dependent setup times literature. For this purpose, we used the set of problems proposed by Ruiz, Maroto e Alcaraz (2005), which was also used in Ruiz e Stützle (2008), and is publicly available at http://soa.iti.es/problem-instances. The tests contain four different sequence dependent setup time ratios (SSD-10, SSD-50, SSD-100 and SSD-125). For example, the instance set SSD-10 consists of 120 instances where the processing times are those of Taillard (1993) benchmark and where the sequence dependent setup times are 10% of the processing times. In the instance set SSD-50, the setup times are 50% of the processing times and the instance sets SSD-100 and SSD-125 have setup times that are 100% and 125% of the processing times respectively. Therefore, the setup times are uniformly distributed in the range [1, 9], [1, 49], [1, 99] and [1, 124] for the instance sets SSD-10, SSD-50, SSD-100 and SSD-125, respectively. This results in four problem sets and a total of $120 \times 4 = 480$ different instances. As we are addressing the mixed no-idle flowshop problem, we considered the same six mixed described in the previous section for each one of the four groups. The total number of problems in the benchmark is $6 \times 480 = 2880$.

Again, to evaluate the performance of heuristics without the no-idle machines, we also compared the heuristics in the original benchmark from Ruiz, Maroto e Alcaraz (2005), which does not consider no-idle machines. In this benchmark, all machines allow idle and dependent setup times are considered in all machines.

3.3.1.4 Benchmark for the MILP model evaluation

For the MILP formulated in Section 2.3.1 we set a maximum elapsed CPU time limit of three hours to optimally solve the problems. With this time termination criterion, the MILP can optimally solve problems with up to 20 jobs and 5 machines. Thus, for the MILP evaluation we considered the following combination between number of jobs and machines $\{n, m\} = \{10, 5\}$, $\{10, 10\}$, $\{15, 5\}$, $\{15, 10\}$, $\{20, 5\}$. Five replications were generated for each combination with processing times generated using the uniform distribution [1, 99] . The same six mixed no-idle groups used in the previous benchmarks were used. For the setup time generation we used the distributions SSD-50, SSD-100 and SSD-125. With these settings, $5 \times 5 \times 3 \times 7 = 525$ instances were created.

3.3.2 Compared Heuristics

As mentioned previously, the $F|prmu,\ mixed\ no-idle, s^i_{j,k}|C_{max}$ problem has not yet been studied in the literature. For this reason, there are no proposed heuristics or metaheuristics for the problem. In order to create a basis for comparison for our RN_x

heuristic (Section 3.2), we adapted constructive heuristics from $F_m|prmu, no-idle|C_{max}$ and $F_m|prmu, s^i_{j,k}|C_{max}$.

According to the review carried out in Section 3.1, many heuristics have already been proposed for the $F_m|prmu, no-idle|C_{max}$ and $F_m|prmu, s^i_{j,k}|C_{max}$ problems. We selected the main methods to be adapted and compared in the problem addressed in this work. The adapted heuristics were modified only in the makespan evaluation of the sequences considering that the mixed no-idle flowshop with the sequence dependent setup times is considered. Thus, the structure of the constructive heuristics remain unchanged, where the makespan evaluation the only modification made. To evaluate the makespan of a sequence, we used the formulas presented in Section 2.3.2. We also implemented the acceleration method described in Section 2.3.3 to calculate the makespan in the insertion neighbourhood when presented, which allowed a large increase in the heuristics' speed. It is important to notice that the constructive phase of the FJSRA heuristic uses an insertion neighbourhood based on inserting of fictitious jobs, which is incompatible with the acceleration procedure. Therefore, it was not possible to apply the acceleration method to the constructive phase of FSJRA and FSJRA-VND. The heuristics selected for comparison are as follows:

- NEH: well-known heuristic from Nawaz, Enscore e Ham (1983).
- GH-BM: heuristic from Baraz e Mosheiov (2008).
- GH-BM2: modification of the GH-BM heuristic by Ruiz, Vallada e Fernández-Martínez (2009).
- FRB4$_x$: heuristic proposed by Rad, Ruiz e Boroojerdian (2009). It was adapted to the mixed no-idle PFSP by Pan e Ruiz (2014). As explained in Section 3.2, x jobs are reinserted around the newly inserted job by the NEH heuristic. In our comparison, we used the values of $x = \{10, 30, 50, 70\}$.
- FRB3: heuristic proposed by Rad, Ruiz e Boroojerdian (2009). In this method, all jobs are reinserted after each NEH iteration. It was implemented for the $F_m|prmu, no-idle|C_{max}$ problem in Ruiz, Vallada e Fernández-Martínez (2009).
- FJSRA: heuristic proposed by Vanchipura e Sridharan (2013) for the $F_m|prmu, s^i_{j,k}|C_{max}$ problem.
- FJSRA-VND: heuristic proposed by Vanchipura, Sridharan e Babu (2014) for the $F_m|\ prmu, s^i_{j,k}|C_{max}$ problem.
- RN$_x$: proposed heuristic in this work (Section 3.2). It is a result of the combination between an greedy method and a NEH variant, which comprises job pair

reinsertion movements. The parameter x limits the number of pairs of jobs selected for reinsertions in the NEH variant. In our comparison we used the values of $x = \{10, 30, 50, 70\}$.

3.3.3 Performance measures

The performance measure used was the Relative Percentage Deviation (RPD) calculated according to Expression (3.9):

$$RPD\Big(C_{max}\left(\pi_h\right)\Big) = 100 \cdot \Big(C_{max}\left(\pi_h\right) - C^*_{max}\Big)/C^*_{max} \tag{3.9}$$

The value of $C_{max}\left(\pi_h\right)$ is the makespan provided by the sequence π_h generated by heuristic h. C^*_{max} is the best solution found among all the compared heuristics. The Average Relative Percentage Deviation is denoted as ARPD. It can be observed that the lower the RPD value, the better the heuristic performance, since the closer its solutions will be to the best result found among all the methods compared.

The Average CPU Time (ACT) in seconds and the Average Relative Percentage computational Time (ARPT) were used to evaluate the computational efficiency of the heuristics.

Fernandez-Viagas e Framinan (2015) detected that the ACT presents several problems when used to evaluate heuristics with different stopping criteria and proposed the ARPT indicator. We use the improved ARPT version of Fernandez-Viagas, Ruiz e Framinan (2017) and Fernandez-Viagas e Framinan (2017):

$$ACT_t = \frac{\sum_{h=1}^{H} T_{th}}{H} \quad \forall t = \{1, \ldots, T\} \tag{3.10}$$

$$RPT_{th} = \frac{T_{th} - ACT_t}{ACT_t} + 1 \quad \forall t = \{1, \ldots, T\}, \quad \forall h = \{1, \ldots, H\} \tag{3.11}$$

$$ARPT_h = \frac{\sum_{t=1}^{T} RPT_{th}}{T} \quad \forall h = \{1, \ldots, H\} \tag{3.12}$$

where T_{th} is the CPU time of heuristic h in the instance t, ACT_t is the Average CPU time considering all the heuristics in the instance t, H is the number of heuristics considered in the evaluation, T is the number of instances in the test bed.

All the compared heuristics were implemented in C ++, compiled with Intel C ++ 16.0, and run on an Intel Xeon E5-2680 processor running at 2.7 GHz with 16 GB of RAM. To solve the MILP model, we used the IBM CPLEX Optimization Studio (version 12.8) with Python Application Programming Interface (API).

3.3.4 Parameter settings of RN_x

As described in Section 3.2, the proposed heuristic RN_x has two parameters. The first parameter, called d, defines the extent to which the solution is constructed by the greedy heuristic (Section 3.2.1). The second parameter, called x, limits the number of jobs to be reinserted in sequence after each insertion of the NEH heuristic variant (Section 3.2.2). This parameter tuning aims to determine which value of d results in better solutions considering the set of values of parameter x. For the first parameter, the values of $d = \{0.1n, 0.2n, 0.3n, 0.4n, 0.5n, 0.6n, 0.7n, 0.8n, 0.9n\}$ were tested, i.e. when $d = 0.1n$, 10% of the sequence is constructed using the greedy method, and the remaining 90% is constructed by the NEH heuristic variant. For the second parameter, the values of $x = \{10, 30, 50, 70\}$ were used. A factorial experiment was carried out between the parameters, generating $9 \times 4 = 36$ different combinations. The set of test problems defined in Section 3.3.1 was used. The ARPDs resulting from the set of problems grouped by number of jobs are presented in Table 4. The results show that the best performance, considering all the problems and all values of x, is obtained from parameter $d = 0.4n$ with an ARPD of 1.50. Therefore, the parameter values $d = 0.4n$ and $x = \{10, 30, 50, 70\}$ were used in the comparisons made in this section.

3.3.5 Comparison between heuristics in the benchmark adapted from Pan e Ruiz (2014)

The results obtained from the evaluated heuristics are summarized in 5 and were grouped by sub-sets of problems with the same number of jobs in Tables 6, 8, 9 and 10. Each one of the tables presents the result of a group of problems related to a setup time distribution, SSD50, SSD100 and SSD125. Tables 6 and 7 present the ARPDs considering all the setup time distributions, grouped by the number of jobs and number of machines, respectively. The mean computational time in seconds of each heuristic considering all distributions is shown in Tables 11 and 12.

Table 4: ARPD values for the parameter tuning of RN_x heuristic. The best results are highlighted in bold.

Heuristic	n	Parameter d values								
		$0.1n$	$0.2n$	$0.3n$	$0.4n$	$0.5n$	$0.6n$	$0.7n$	$0.8n$	$0.9n$
RN_{10}	60	2.44	**2.39**	2.54	2.41	2.49	2.71	2.79	3.39	4.70
	120	2.58	2.66	2.59	**2.58**	2.62	2.62	2.85	3.33	4.32
	240	2.73	2.61	2.55	2.48	2.45	**2.42**	2.49	2.81	3.61
	360	2.73	2.57	2.45	2.42	2.28	2.26	**2.25**	2.52	3.08
	Average	2.62	2.56	2.53	2.47	**2.46**	2.50	2.60	3.01	3.93
RN_{30}	60	1.56	1.60	1.59	**1.43**	1.75	1.70	1.81	2.20	3.02
	120	1.68	**1.51**	1.65	1.54	1.61	1.67	1.87	2.07	3.04
	240	1.54	1.51	1.43	**1.37**	1.46	1.40	1.54	1.74	2.48
	360	1.60	1.55	1.42	1.40	1.34	**1.31**	1.38	1.59	2.17
	Average	1.59	1.54	1.52	**1.44**	1.54	1.52	1.65	1.90	2.68
RN_{50}	60	1.45	**1.28**	1.31	1.33	1.50	1.30	1.55	1.88	2.40
	120	1.21	1.24	1.30	**1.20**	1.29	1.37	1.50	1.84	2.47
	240	1.12	1.07	1.02	**0.99**	1.00	1.01	1.12	1.39	2.07
	360	1.15	1.05	1.03	0.95	0.90	**0.88**	1.02	1.18	1.70
	Average	1.23	1.16	1.17	**1.12**	1.17	1.14	1.30	1.57	2.16
RN_{70}	60	1.41	**1.29**	1.32	1.35	1.44	1.29	1.53	1.82	2.49
	120	1.05	1.03	**0.97**	0.97	1.08	1.03	1.32	1.46	2.26
	240	0.92	0.86	0.78	**0.75**	0.80	0.76	1.01	1.17	1.74
	360	0.90	0.84	0.77	0.75	**0.64**	0.69	0.73	0.98	1.47
	Average	1.07	1.00	0.96	0.96	0.99	**0.94**	1.15	1.36	1.99
$\overline{\text{Average}}$		1.63	1.56	1.54	**1.50**	1.54	1.53	1.67	1.96	2.69

Table 5: Summary of the results in the benchmark adapted from Pan e Ruiz (2014).

Heuristic	ARPD				Average	ARPT
	SSD50	SSD100	SSD125	Average	CPU Time	
RN_{70}	0.42	0.37	0.35	0.38	11.37	1.29
RN_{50}	0.64	0.60	0.57	0.61	8.52	1.04
RN_{30}	1.00	0.96	0.94	0.96	5.38	0.72
FRB3	0.75	1.15	1.36	1.09	39.24	3.04
$FRB4_{70}$	1.45	2.05	2.33	1.94	11.69	1.35
RN_{10}	1.97	1.93	1.93	1.95	1.92	0.29
$FRB4_{50}$	1.63	2.26	2.52	2.14	8.74	1.09
FJSRA-VND	2.47	2.16	2.19	2.27	18.78	1.55
$FRB4_{30}$	2.04	2.70	2.99	2.57	5.72	0.76
FJSRA	4.11	3.39	3.37	3.62	17.07	1.41
$FRB4_{10}$	3.09	3.97	4.27	3.78	2.00	0.30
GH-BM2	4.60	5.46	5.77	5.28	0.49	0.08
NEH	6.15	6.98	7.28	6.80	0.16	0.03
GH-BM	11.15	11.38	10.88	11.14	11.21	1.05

Table 6: ARPD values for the compared heuristics in the all distributions grouped by number of jobs in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best ARPD, in each line.

n	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
50	6.08	9.70	4.07	2.64	1.70	1.53	1.53	1.39	6.18	3.49	1.71	0.87	0.66	**0.66**
100	6.66	10.78	4.82	3.32	2.26	1.91	1.71	1.36	5.29	3.21	1.95	1.00	0.72	**0.42**
150	6.70	10.90	5.08	3.47	2.37	1.83	1.74	1.17	4.24	2.66	1.94	1.00	0.59	**0.41**
200	6.79	11.15	5.24	3.67	2.50	1.98	1.81	1.02	3.91	2.50	2.01	0.97	0.61	**0.34**
250	6.84	11.26	5.33	3.81	2.52	2.13	1.86	1.00	3.45	2.22	1.92	0.90	0.54	**0.27**
300	6.85	11.22	5.44	3.93	2.68	2.22	2.00	1.05	2.98	1.91	1.94	0.91	0.55	**0.32**
350	6.90	11.40	5.52	4.08	2.84	2.36	2.12	1.01	2.74	1.78	2.00	0.99	0.60	**0.33**
400	7.21	11.90	5.85	4.31	2.97	2.43	2.25	0.99	2.66	1.74	1.97	1.00	0.56	**0.33**
450	6.97	11.47	5.68	4.21	2.93	2.45	2.18	0.94	2.43	1.62	1.98	0.99	0.60	**0.36**
500	7.03	11.59	5.77	4.33	2.99	2.52	2.23	0.91	2.35	1.60	2.02	1.01	0.63	**0.37**
Average	6.80	11.14	5.28	3.78	2.57	2.14	1.94	1.09	3.62	2.27	1.95	0.96	0.61	**0.38**

Table 7: ARPD values for the compared heuristics in the all distributions grouped by number of machines in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best ARPD, in each line.

m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
10	8.70	14.64	6.75	4.95	3.49	2.99	2.80	1.79	3.17	2.00	2.19	1.02	0.64	**0.38**
20	7.46	12.18	5.86	4.22	2.84	2.36	2.17	1.21	3.49	2.20	2.11	1.03	0.64	**0.41**
30	6.48	10.59	5.07	3.56	2.38	1.98	1.77	0.96	3.61	2.28	1.91	0.98	0.61	**0.37**
40	5.87	9.54	4.52	3.18	2.16	1.74	1.56	0.76	3.91	2.45	1.79	0.90	0.58	**0.36**
50	5.50	8.73	4.20	2.97	2.00	1.62	1.41	0.71	3.93	2.43	1.73	0.89	0.57	**0.38**
Average	6.80	11.14	5.28	3.78	2.57	2.14	1.94	1.09	3.62	2.27	1.95	0.96	0.61	**0.38**

Table 8: ARPD values for the compared heuristics in the SSD50 distribution in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best ARPD, in each line.

n	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
50	5.96	9.77	3.94	2.24	1.39	1.28	1.28	1.24	7.09	3.91	1.74	0.96	0.65	**0.65**
100	6.23	10.60	4.32	2.81	1.86	1.55	1.34	1.01	5.99	3.41	1.84	0.96	0.68	**0.39**
150	6.12	10.85	4.46	2.97	2.00	1.41	1.32	0.88	4.73	2.84	1.94	1.02	0.59	**0.36**
200	6.18	11.02	4.53	3.00	2.01	1.49	1.38	0.70	4.36	2.60	1.93	1.02	0.59	**0.36**
250	6.18	11.29	4.65	3.09	1.97	1.62	1.36	0.66	3.92	2.41	1.91	0.90	0.55	**0.24**
300	6.10	11.35	4.70	3.15	2.07	1.63	1.44	0.67	3.36	2.07	1.96	0.91	0.61	**0.35**
350	6.21	11.59	4.81	3.33	2.21	1.75	1.53	0.61	3.13	1.95	2.06	1.03	0.67	**0.40**
400	6.15	11.75	4.80	3.41	2.20	1.75	1.60	0.60	3.03	1.92	2.08	1.08	0.63	**0.45**
450	6.19	11.66	4.87	3.36	2.28	1.92	1.60	0.56	2.80	1.80	2.10	1.07	0.69	**0.51**
500	6.21	11.62	4.95	3.49	2.38	1.90	1.66	0.55	2.67	1.76	2.13	1.09	0.77	**0.51**
Average	6.15	11.15	4.60	3.09	2.04	1.63	1.45	0.75	4.11	2.47	1.97	1.00	0.64	**0.42**

Table 9: ARPD values for the compared heuristics in the SSD100 distribution in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best ARPD, in each line.

n	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
50	6.06	9.62	4.01	2.80	1.69	1.44	1.44	1.26	5.73	3.22	1.60	0.85	0.71	**0.71**
100	6.89	11.02	5.05	3.47	2.40	2.06	1.86	1.50	4.97	3.08	2.05	0.99	0.72	**0.42**
150	6.82	11.13	5.24	3.51	2.42	1.87	1.77	1.20	3.99	2.56	1.93	0.97	0.66	**0.43**
200	6.92	11.45	5.44	3.87	2.52	2.08	1.86	1.14	3.74	2.47	2.04	0.92	0.66	**0.31**
250	6.91	11.45	5.40	3.92	2.62	2.25	1.99	1.08	3.26	2.14	1.93	0.85	0.52	**0.27**
300	6.93	11.40	5.50	4.10	2.82	2.34	2.10	1.05	2.75	1.80	1.97	0.94	0.53	**0.28**
350	7.00	11.58	5.64	4.26	2.97	2.49	2.20	1.09	2.56	1.64	1.95	1.00	0.51	**0.34**
400	7.91	12.61	6.50	4.81	3.36	2.78	2.51	1.22	2.43	1.62	1.97	1.06	0.54	**0.26**
450	7.12	11.70	5.84	4.45	3.07	2.55	2.34	1.01	2.28	1.55	1.94	1.00	0.60	**0.31**
500	7.24	11.87	6.00	4.52	3.13	2.72	2.38	0.99	2.23	1.52	1.97	0.98	0.58	**0.35**
Average	6.98	11.38	5.46	3.97	2.70	2.26	2.05	1.15	3.39	2.16	1.93	0.96	0.60	**0.37**

Table 10: ARPD values for the compared heuristics in the SSD125 distribution in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best ARPD, in each line.

n	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
50	6.21	9.70	4.24	2.89	2.00	1.86	1.86	1.66	5.71	3.34	1.81	0.81	0.63	**0.63**
100	6.85	10.73	5.10	3.68	2.53	2.12	1.92	1.58	4.93	3.15	1.97	1.05	0.76	**0.43**
150	7.17	10.73	5.54	3.93	2.71	2.20	2.12	1.44	3.99	2.57	1.97	1.01	0.51	**0.42**
200	7.29	10.99	5.76	4.15	2.97	2.39	2.18	1.23	3.64	2.43	2.06	0.98	0.58	**0.37**
250	7.42	11.04	5.94	4.40	2.96	2.52	2.22	1.27	3.18	2.12	1.93	0.95	0.54	**0.31**
300	7.51	10.90	6.13	4.53	3.17	2.70	2.46	1.43	2.82	1.87	1.88	0.89	0.51	**0.33**
350	7.48	11.04	6.10	4.64	3.33	2.85	2.63	1.33	2.52	1.73	2.00	0.93	0.63	**0.26**
400	7.57	11.36	6.24	4.72	3.34	2.77	2.63	1.16	2.52	1.69	1.88	0.88	0.50	**0.26**
450	7.59	11.04	6.34	4.82	3.44	2.88	2.58	1.25	2.22	1.52	1.89	0.89	0.51	**0.27**
500	7.65	11.30	6.35	4.98	3.45	2.95	2.66	1.20	2.15	1.51	1.95	0.96	0.54	**0.27**
Average	7.28	10.88	5.77	4.27	2.99	2.52	2.33	1.36	3.37	2.19	1.93	0.94	0.57	**0.35**

Table 11: Average CPU values and ARPT for the compared heuristics grouped by number of jobs in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best results, in each line.

n	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
50	**0.00**	0.03	0.01	0.03	0.06	0.06	0.06	0.07	0.01	0.01	0.03	0.05	0.06	0.06
100	**0.01**	0.28	0.04	0.13	0.31	0.45	0.52	0.61	0.10	0.12	0.13	0.31	0.43	0.50
150	**0.03**	0.99	0.09	0.32	0.87	1.30	1.58	2.40	0.54	0.60	0.32	0.80	1.19	1.49
200	**0.05**	2.37	0.16	0.63	1.81	2.69	3.40	6.47	1.72	1.89	0.60	1.56	2.38	3.10
250	**0.09**	4.63	0.27	1.09	3.16	4.66	6.12	14.20	4.21	4.64	1.02	2.73	4.25	5.55
300	**0.13**	8.00	0.42	1.72	4.93	7.36	9.82	25.92	8.74	9.63	1.58	4.39	6.87	9.10
350	**0.19**	12.71	0.60	2.48	7.10	10.79	14.48	43.69	16.08	17.69	2.34	6.54	10.37	13.84
400	**0.26**	19.03	0.82	3.44	9.75	14.97	20.17	66.25	27.98	30.83	3.29	9.25	14.72	19.73
450	**0.34**	27.08	1.08	4.50	12.84	19.87	26.69	96.75	44.22	48.67	4.34	12.31	19.59	26.38
500	**0.44**	36.96	1.38	5.67	16.38	25.27	34.05	136.05	67.08	73.75	5.59	15.85	25.31	33.99
Average	**0.16**	11.21	0.49	2.00	5.72	8.74	11.69	39.24	17.07	18.78	1.92	5.38	8.52	11.37
ARPT	**0.03**	1.05	0.08	0.30	0.76	1.09	1.35	3.04	1.41	1.55	0.29	0.72	1.04	1.29

Table 12: Average CPU values for the compared heuristics grouped by number of machines in the benchmark adapted from Pan e Ruiz (2014). The values in bold are the best results, in each line.

m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
10	**0.04**	3.48	0.12	0.48	1.43	2.20	2.91	9.86	17.10	18.79	0.46	1.29	2.03	2.75
20	**0.09**	7.34	0.27	1.13	3.33	5.11	6.89	22.59	17.05	18.76	1.07	3.06	4.85	6.56
30	**0.15**	11.21	0.47	1.92	5.53	8.46	11.33	37.11	17.04	18.72	1.85	5.19	8.23	11.04
40	**0.22**	15.04	0.68	2.78	7.87	12.02	16.16	54.11	17.06	18.81	2.68	7.46	11.81	15.71
50	**0.29**	18.97	0.90	3.68	10.43	15.93	21.16	72.53	17.09	18.83	3.56	9.90	15.68	20.80
Average	**0.16**	11.21	0.49	2.00	5.72	8.74	11.69	39.24	17.07	18.78	1.92	5.38	8.52	11.37

Among the heuristics in the literature, the one that presented the best results was the FRB3 that surpassed the NEH, GH-BM, GH-BM2 and FRB4$_x$ methods by a significant difference. Note that the FRB3 obtained an ARPD of 0.75 compared to 6.15 of the NEH heuristic in the SSD50 distribution (Table 8). However, FRB3 is considerably more computational complex than the rest of the heuristic methods, with an average computation time of approximately 39 seconds, compared to 0.16 seconds of the NEH heuristic and around 2 seconds of our proposed RN$_{10}$ heuristic (Tables 11 and 12). Moreover, it can be observed that FRB3 has the highest ARPT with 3.04 compared to 0.16 from NEH. Among the heuristics in the literature of less complexity which obtained the best results was FRB4$_{70}$, with an ARPD of 1.94 and an average CPU time of around 12 seconds considering all the setup time distributions. Moreover, the results grouped by the number of machines provided in Tables 7 and 12 show a strong influence of the number of machines in the ARPD and average CPU time values. The general trend is that better results are obtained as the number of machines increase (FRB3 obtains an ARPD of 1.79 when $m = 10$ and 0.71 for $m = 50$). The only exception is the RN$_{70}$ heuristics where the ARPD remains stable as the number of machines increases.

Considering all the heuristics, the proposed RN$_{70}$ method obtained the best results considering all distributions, with an ARPD of 0.38, an average CPU time of about 11 seconds and an ARPT of 1.29, compared to an ARPD of 1.09, an average CPU time of approximately 39 seconds and an ARPT of 3.04 of the FRB3 heuristic. That is, the RN$_{70}$ heuristic is clearly better as it produces better solutions in less computational time. The highest average CPU time by the FRB3 was already expected as the FRB3 method reinserts all the jobs of the sequence at each iteration, resulting in a complex neighbourhood search procedure. Therefore, the strategy used in the FRB4$_x$ and RN$_x$ to limit the number of jobs to be reinserted by means of the parameter x has demonstrated an efficient way to improve the quality of the final solution without the method becoming very computationally intensive. The results also show that the quality of the solution is improved in the FRB4$_x$ and RN$_x$ heuristics as the value of x increases. However, the average computational time and ARPT values are also increased as more reinsertions are performed. The heuristics from the setup times literature, FJSRA and FJSRA-VND, present good APRD results, especially when the distribution intervals increase (SSD100 and SSD125), however the average CPU times and ARPT values are worse when compared to the RN$_x$ heuristic ($x = \{10, 30, 50, 70\}$). These results are due to the SRA (setup ranking algorithm), which is used to create the fictitious job for the FJSRA and FJSRA-VND, and is being computationally intensive. To make matters worse, the acceleration procedure from Section 2.3.3 cannot be applied to the construction phase of both heuristics. Figure 9 shows the ARPD values and ARPT of the heuristics. The dominant heuristics are highlighted in green, while the remaining are in red.

Although Tables 6, 7, 11 and 12 show that the proposed method is clearly better

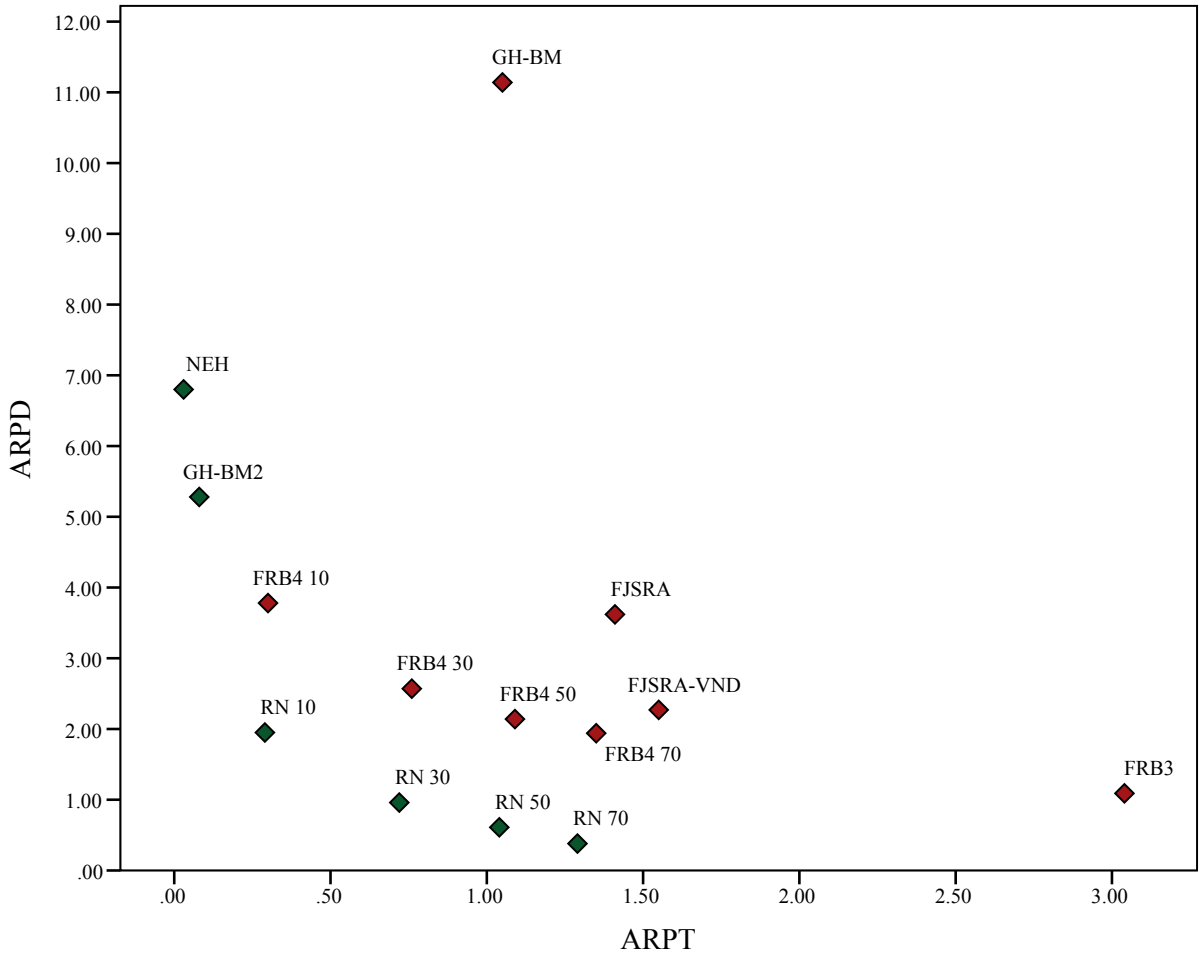

Figure 9: ARPD and ARPT values for compared heuristics. The Pareto dominating heuristic is depicted in green.

than the best heuristics in the literature, a statistical analysis should be performed to prove that the differences in the ARPD values are in fact statistically significant. Figure 10 presents the measurements with 95% confidence intervals in the different setup time distributions. It can be stated that heuristics have statistically different averages. The best heuristics in the literature (FRB4$_x$ and FRB3) obtain consistently worse results as the intervals of distributions increase. In contrast, the RN$_x$ heuristic has a stable performance, with slightly better results in the SSD100 and SSD125 distributions. This shows that the proposed heuristic is favoured when setup times are longer. Table 13 presents the Tukey test, grouped by number of jobs, that shows that the main compared heuristics have statistically different means. Therefore, the presented results and the statistical analyses carried out show that the proposed RN$_x$ method is better than the heuristics adapted from the literature, both in solution quality and in computational efficiency.

Although the experiments show that the proposed heuristics work well for the

Table 13: Tukey test results of the best heuristics, with significance level of 95% in the benchmark from Pan e Ruiz (2014). The values in bold mean that there is a significant statistical difference between the algorithms in the first and second column.

n	Heuristic (I)	Heuristic (J)	Mean Difference (I-J)	Standard Error	Significance
50 - 100[a]	$FRB4_{70}$	FRB3	0.242	0.050	**0.000**
		RN_{70}	1.078	0.050	**0.000**
	FRB3	$FRB4_{70}$	-0.242	0.050	**0.000**
		RN_{70}	0.836	0.050	**0.000**
	RN_{70}	$FRB4_{70}$	-1.078	0.050	**0.000**
		FRB3	-0.836	0.050	**0.000**
150 - 200	$FRB4_{70}$	FRB3	0.674	0.043	**0.000**
		RN_{70}	1.398	0.043	**0.000**
	FRB3	$FRB4_{70}$	-0.674	0.043	**0.000**
		RN_{70}	0.724	0.043	**0.000**
	RN_{70}	$FRB4_{70}$	-1.398	0.043	**0.000**
		FRB3	-0.724	0.043	**0.000**
250 - 300	$FRB4_{70}$	FRB3	0.902	0.047	**0.000**
		RN_{70}	1.631	0.047	**0.000**
	FRB3	$FRB4_{70}$	-0.902	0.047	**0.000**
		RN_{70}	0.729	0.047	**0.000**
	RN_{70}	$FRB4_{70}$	-1.631	0.047	**0.000**
		FRB3	-0.729	0.047	**0.000**
350 - 400	$FRB4_{70}$	FRB3	1.184	0.049	**0.000**
		RN_{70}	1.857	0.049	**0.000**
	FRB3	$FRB4_{70}$	-1.184	0.049	**0.000**
		RN_{70}	0.673	0.049	**0.000**
	RN_{70}	$FRB4_{70}$	-1.857	0.049	**0.000**
		FRB3	-0.673	0.049	**0.000**
450 - 500	$FRB4_{70}$	FRB3	1.277	0.049	**0.000**
		RN_{70}	1.835	0.049	**0.000**
	FRB3	$FRB4_{70}$	-1.277	0.049	**0.000**
		RN_{70}	0.557	0.049	**0.000**
	RN_{70}	$FRB4_{70}$	-1.835	0.049	**0.000**
		FRB3	-0.557	0.049	**0.000**

[a] Instances with 50 and 100 jobs.

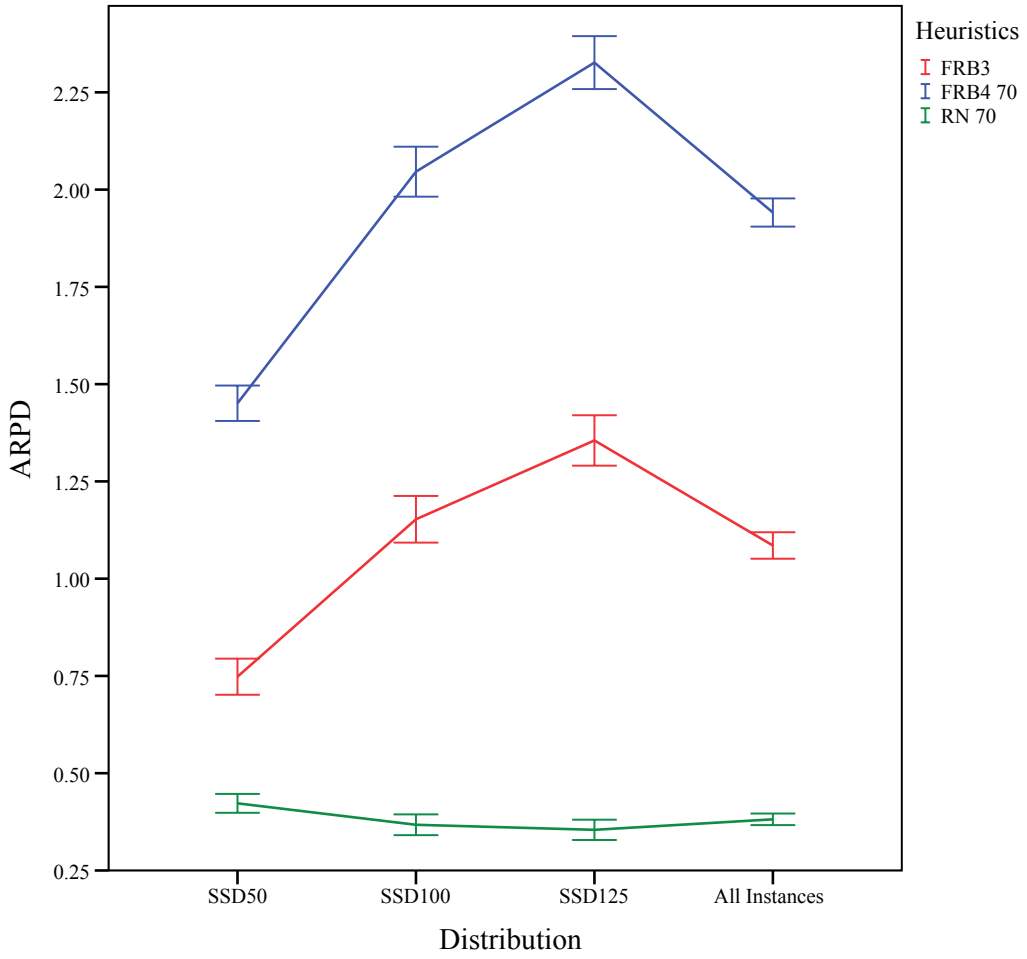

Figure 10: Means plot for the heuristics in all distributions for the benchmark from Pan e Ruiz (2014). All means have 95% confidence intervals

mixed no-idle instances with sequence dependent setup times, it is important to know the performance of our heuristic on (mixed) no-idle instances only (without sequence-dependent setup times). Tables 14 and 15 show that RN_{70} (APRD value of 0.44) presents similar results to FRB3 (ARPD value of 0.23), which obtained the best performance between the compared heuristics. Thus, we conclude that the proposed strategy can also obtain a good performance also for the mixed no-idle PFSP without setup times.

3.3.6 Comparison between heuristics in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005)

We evaluated the RN_x heuristic on the instances from the literature with setup times proposed by Ruiz, Maroto e Alcaraz (2005). The results are summarized in Table 16. The ARPD values grouped by number of jobs are shown in Table 17, and presented in more detail in Tables 18, 19, 20, 21 and 22. The results are similar to those from the

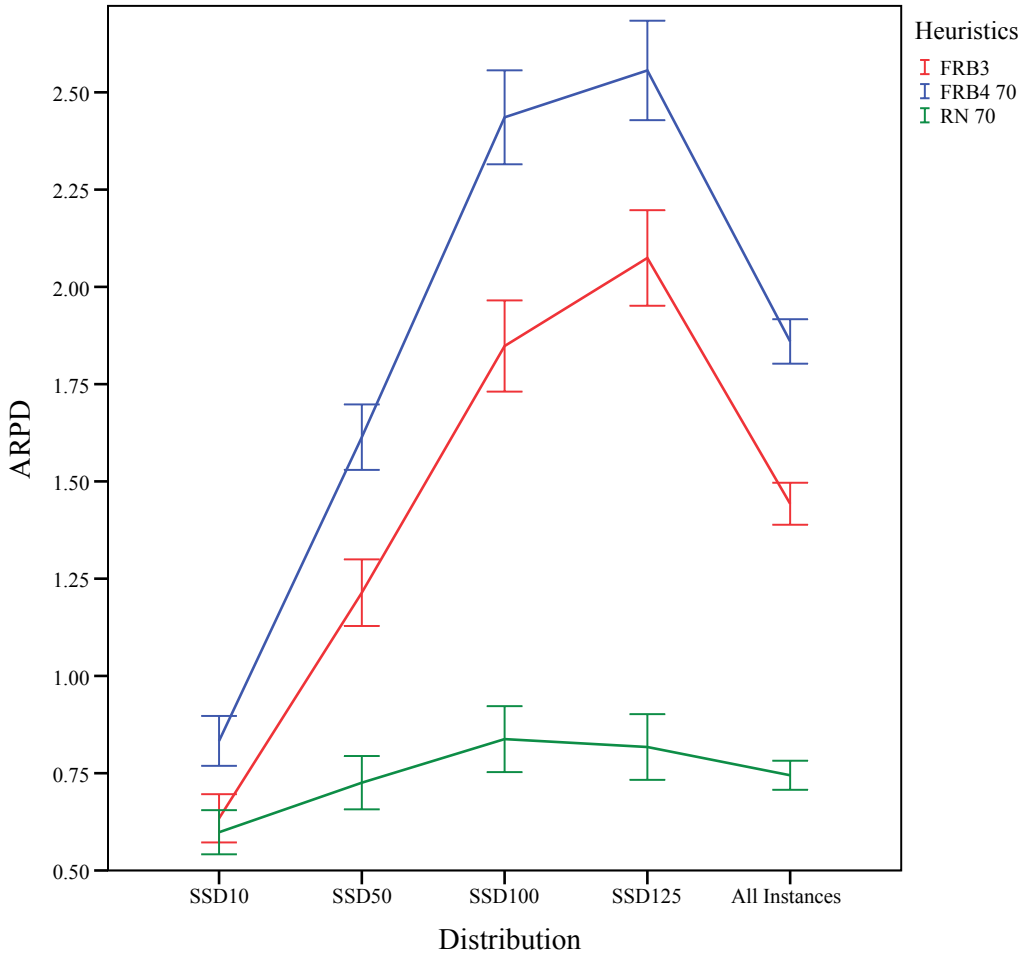

Figure 11: Means plot for the heuristics in all distributions for the benchmark from Ruiz, Maroto e Alcaraz (2005). All means have 95% confidence intervals

benchmark of Pan e Ruiz (2014). It can be observed that the RN_x obtained better results in all distributions. For example, RN_{70} obtained an ARPD of 0.60, 0.73, 0.84 and 0.82 for distribution intervals SSD10, SSD50, SSD100 and SSD125, respectively, compared to 0.63, 1.21, 1.85 and 2.07 from FRB3 and 0.83, 1.61, 2.44 and 2.56 from $FRB4_{70}$. In terms of computational efficiency, the RN_x ($x = \{10, 30, 50, 70\}$) presented lower average CPU times and ARPT when compared to FRB3. On average, the FRB3 consumes in average around 7 seconds compared to RN_{70} with approximately 2 seconds and has an ARPT of 2.32, compared to 1.33 from RN_{70}. The RN_{10} heuristic shows exceptional efficiency with an ARPT of 0.58, compared to 2.32 and 1.37 of FRB3 and $FRB4_{70}$, respectively. Figure 11 shows the ARPD results with 95% confidence intervals.

Table 14: ARPD results in the benchmark from Pan e Ruiz (2014) without sequence dependent setup times. The values in bold are the best results, in each line.

n	NEH	GH-BM2	$FRB4_{70}$	FRB3	FJSRA-VND	RN_{70}
50	5.22	2.91	0.75	0.90	5.11	**0.68**
100	5.04	2.88	0.74	0.55	4.69	**0.49**
150	4.39	2.52	0.74	**0.25**	4.17	0.53
200	3.86	2.23	0.66	**0.16**	3.51	0.47
250	3.44	2.00	0.60	**0.13**	3.27	0.44
300	3.26	1.90	0.60	**0.10**	3.04	0.42
350	3.03	1.77	0.50	**0.08**	2.83	0.38
400	2.66	1.52	0.48	**0.07**	2.49	0.35
450	2.56	1.50	0.47	**0.04**	2.37	0.35
500	2.39	1.40	0.43	**0.04**	2.19	0.35
Average	3.59	2.06	0.60	**0.23**	3.37	0.44

Table 15: Average CPU times in the benchmark from Pan e Ruiz (2014) without sequence dependent setup times.

n	NEH	GH-BM2	$FRB4_{70}$	FRB3	FJSRA-VND	RN_{70}
50	0.00	0.01	0.06	0.07	0.07	0.07
100	0.02	0.04	0.50	0.60	0.55	0.55
150	0.05	0.10	1.41	2.23	1.60	1.62
200	0.08	0.22	3.13	5.98	3.43	3.45
250	0.12	0.30	5.51	11.64	5.92	5.96
300	0.20	0.47	8.21	19.84	9.13	8.93
350	0.29	0.70	12.10	32.17	13.14	13.00
400	0.38	0.96	16.05	49.95	17.57	17.60
450	0.46	1.22	21.05	73.95	22.94	22.73
500	0.64	1.53	27.26	106.40	28.63	28.54
Average	0.22	0.56	9.53	30.28	10.30	10.24

Table 16: Summary of the results in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005).

Heuristic	ARPD					Average	ARPT
	SSD10	SSD50	SSD100	SSD125	Average	CPU Time	
RN_{70}	0.60	0.73	0.84	0.82	0.74	2.01	1.33
RN_{50}	0.66	0.84	0.95	0.93	0.85	1.52	1.22
RN_{30}	0.78	1.05	1.22	1.22	1.07	0.96	1.01
FRB3	0.63	1.21	1.85	2.07	1.44	6.70	2.32
RN_{10}	1.35	1.80	2.11	2.12	1.85	0.34	0.58
$FRB4_{70}$	0.83	1.61	2.44	2.56	1.86	1.93	1.37
$FRB4_{50}$	0.91	1.70	2.60	2.71	1.98	1.44	1.25
$FRB4_{30}$	1.10	1.93	2.85	3.04	2.23	0.92	1.01
FJSRA-VND	3.22	2.93	3.03	2.91	3.03	6.39	1.29
$FRB4_{10}$	1.60	2.75	3.84	4.16	3.09	0.35	0.53
GH-BM2	2.48	4.48	5.68	6.07	4.68	0.08	0.16
FJSRA	6.80	5.39	5.12	5.03	5.58	5.80	1.19
NEH	4.26	6.45	7.78	8.37	6.72	0.02	0.03
GH-BM	7.18	11.44	13.01	12.83	11.11	2.22	0.72

Table 17: ARPD values for the compared heuristics in the all distributions intervals in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005). The values in bold are the best ARPD, in each line.

n	m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
20	5	6.07	10.89	3.77	2.54	2.10	2.10	2.10	1.74	7.20	3.45	1.57	1.41	1.41	**1.41**
20	10	6.38	10.49	3.89	2.04	1.91	1.91	1.91	**1.49**	8.45	4.10	1.96	1.51	1.51	1.51
20	20	5.48	8.20	3.29	1.82	1.47	1.47	1.47	1.56	7.89	3.64	1.59	1.19	1.19	**1.19**
50	5	6.81	11.89	4.65	3.05	2.27	2.14	2.14	1.74	5.01	2.65	1.56	0.89	0.74	**0.74**
50	10	7.28	11.99	4.85	3.19	1.99	1.78	1.78	1.77	6.53	3.74	1.87	0.93	0.82	**0.82**
50	20	6.36	10.18	4.30	2.57	1.88	1.63	1.63	1.46	7.59	4.07	1.83	0.88	0.71	**0.71**
100	5	7.22	12.03	5.21	3.55	2.53	2.27	2.08	1.82	3.79	2.31	1.82	0.90	0.62	**0.39**
100	10	7.07	12.12	5.17	3.37	2.48	1.95	1.74	1.41	4.78	2.85	2.02	1.04	0.76	**0.52**
100	20	6.64	10.85	4.75	3.16	2.11	1.74	1.63	1.25	5.92	3.44	2.03	1.14	0.68	**0.57**
200	10	7.43	12.55	5.61	3.94	2.70	2.33	2.05	1.33	3.12	1.90	1.97	0.97	0.55	**0.39**
200	20	6.94	11.02	5.12	3.67	2.40	2.00	1.67	0.94	4.17	2.51	1.97	0.92	0.57	**0.30**
500	20	6.91	11.14	5.53	4.15	2.89	2.45	2.12	0.81	2.57	1.63	1.98	1.03	0.60	**0.39**
Average		6.72	11.11	4.68	3.09	2.23	1.98	1.86	1.44	5.58	3.03	1.85	1.07	0.85	**0.74**

Table 18: ARPD values for the compared heuristics in the SSD10 distribution interval in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005). The values in bold are the best ARPD, in each line.

n	m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
20	5	3.42	7.39	1.88	1.11	1.03	1.03	1.03	0.98	8.20	3.58	0.75	0.91	0.91	**0.91**
20	10	5.21	8.56	3.08	1.68	1.54	1.54	1.54	**1.05**	10.68	4.53	1.73	1.08	1.08	1.08
20	20	5.48	7.29	2.97	1.45	1.09	1.09	1.09	**1.32**	10.91	4.87	1.45	1.38	1.38	1.38
50	5	2.83	5.65	1.55	0.96	0.69	0.55	0.55	0.44	4.49	1.86	0.79	0.42	0.34	**0.34**
50	10	4.91	8.28	2.92	1.78	1.15	0.93	0.93	0.93	8.23	4.00	1.43	0.57	0.64	**0.64**
50	20	5.04	8.70	3.09	2.30	1.57	1.22	1.22	1.06	10.47	5.03	2.00	1.10	0.73	**0.73**
100	5	2.69	5.05	1.57	0.91	0.57	0.47	0.41	0.37	2.89	1.39	0.67	0.30	0.27	**0.13**
100	10	3.74	6.69	2.11	1.43	0.93	0.62	0.56	0.41	5.27	2.63	1.27	0.60	0.60	**0.34**
100	20	5.22	8.20	2.98	2.30	1.27	1.03	0.77	**0.56**	8.13	4.19	1.89	0.96	0.66	0.65
200	10	3.44	6.32	2.06	1.42	0.83	0.67	0.48	**0.18**	3.42	1.70	1.18	0.55	0.34	0.26
200	20	5.00	7.66	2.92	1.98	1.29	0.84	0.66	**0.24**	5.65	3.01	1.62	0.70	0.53	0.30
500	20	4.12	6.42	2.65	1.83	1.20	0.97	0.77	**0.08**	3.32	1.85	1.47	0.83	0.50	0.42
Average		4.26	7.18	2.48	1.60	1.10	0.91	0.83	0.63	6.80	3.22	1.35	0.78	0.66	**0.60**

Table 19: ARPD values for the compared heuristics in the SSD50 distribution interval in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005). The values in bold are the best ARPD, in each line.

n	m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
20	5	5.97	10.37	3.99	2.10	1.79	1.79	1.79	1.47	6.46	3.05	1.50	1.44	1.44	**1.44**
20	10	6.22	10.73	3.76	1.73	1.66	1.66	1.66	**1.33**	8.49	3.95	1.79	1.41	1.41	1.41
20	20	5.17	8.45	3.32	1.93	1.44	1.44	1.44	1.58	7.41	3.28	1.52	1.03	1.03	**1.03**
50	5	6.34	11.86	4.18	2.65	1.82	1.72	1.72	1.31	5.07	2.79	1.57	0.68	0.65	**0.65**
50	10	7.11	12.15	4.56	3.03	1.85	1.54	1.54	1.45	6.26	3.70	1.95	0.98	0.88	**0.88**
50	20	6.69	10.42	4.35	2.40	1.82	1.50	1.50	1.32	7.61	4.17	1.52	0.85	0.65	**0.65**
100	5	6.43	11.79	4.70	2.86	2.00	1.82	1.62	1.50	3.60	2.23	1.72	0.87	0.51	**0.30**
100	10	6.66	12.28	4.80	2.95	2.01	1.61	1.55	1.03	4.60	2.77	1.96	1.02	0.75	**0.50**
100	20	6.65	11.59	4.76	2.87	1.81	1.59	1.56	1.20	5.63	3.28	1.94	1.07	0.75	**0.63**
200	10	6.77	12.75	5.01	3.18	2.12	1.86	1.56	1.03	3.00	1.82	1.93	0.91	0.61	**0.30**
200	20	6.64	12.06	4.91	3.45	2.13	1.73	1.55	0.68	3.97	2.47	1.97	1.02	0.62	**0.38**
500	20	6.76	12.81	5.47	3.85	2.66	2.10	1.87	0.67	2.58	1.70	2.24	1.27	0.75	**0.53**
Average		6.45	11.44	4.48	2.75	1.93	1.70	1.61	1.21	5.39	2.93	1.80	1.05	0.84	**0.73**

Table 20: ARPD values for the compared heuristics in the SSD100 distribution interval in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005). The values in bold are the best ARPD, in each line.

n	m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
20	5	7.19	13.00	4.93	3.42	3.01	3.01	3.01	2.46	7.03	3.71	2.08	1.89	1.89	**1.89**
20	10	6.63	11.10	4.54	2.40	2.33	2.33	2.33	1.73	7.31	4.15	2.17	1.57	1.57	**1.57**
20	20	5.83	8.46	3.62	1.92	1.83	1.83	1.83	1.55	6.95	3.26	1.52	1.17	1.17	**1.17**
50	5	8.33	14.91	5.62	3.83	2.75	2.73	2.73	1.99	5.07	2.81	2.00	1.20	0.91	**0.91**
50	10	8.27	13.92	5.73	3.85	2.47	2.33	2.33	2.34	5.87	3.90	2.18	1.14	0.79	**0.79**
50	20	6.80	10.87	4.75	2.95	2.00	1.89	1.89	1.76	6.19	3.60	1.81	0.84	0.80	**0.80**
100	5	9.38	15.47	6.82	4.72	3.58	3.13	2.97	2.58	4.21	2.67	2.40	1.27	0.89	**0.60**
100	10	8.55	14.82	6.57	4.31	3.15	2.58	2.17	1.96	4.88	3.18	2.31	1.14	0.93	**0.58**
100	20	7.03	12.06	5.35	3.52	2.59	2.16	2.07	1.48	5.08	3.29	2.24	1.23	0.60	**0.57**
200	10	9.49	16.05	7.41	5.37	3.91	3.30	3.04	1.94	3.18	2.15	2.47	1.19	0.64	**0.56**
200	20	7.82	12.45	6.05	4.44	2.98	2.66	2.09	1.25	3.57	2.30	2.03	0.94	0.60	**0.27**
500	20	8.09	12.95	6.74	5.30	3.63	3.19	2.77	1.13	2.05	1.38	2.16	1.07	0.64	**0.33**
Average		7.78	13.01	5.68	3.84	2.85	2.60	2.44	1.85	5.12	3.03	2.11	1.22	0.95	**0.84**

Table 21: ARPD values for the compared heuristics in the SSD125 distribution interval in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005). The values in bold are the best ARPD, in each line.

n	m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
20	5	7.69	12.81	4.29	3.51	2.56	2.56	2.56	2.03	7.12	3.48	1.96	1.41	1.41	**1.41**
20	10	7.47	11.58	4.20	2.35	2.11	2.11	2.11	**1.86**	7.30	3.76	2.15	1.96	1.96	1.96
20	20	5.45	8.61	3.26	1.99	1.52	1.52	1.52	1.80	6.28	3.17	1.85	1.17	1.17	**1.17**
50	5	9.75	15.12	7.24	4.76	3.82	3.54	3.54	3.22	5.42	3.15	1.88	1.27	1.06	**1.06**
50	10	8.81	13.63	6.20	4.10	2.51	2.32	2.32	2.38	5.76	3.36	1.93	1.04	0.96	**0.96**
50	20	6.91	10.72	5.02	2.61	2.14	1.92	1.92	1.70	6.07	3.49	1.99	0.71	0.68	**0.68**
100	5	10.38	15.81	7.73	5.69	3.99	3.65	3.33	2.82	4.45	2.94	2.50	1.16	0.81	**0.52**
100	10	9.32	14.69	7.20	4.78	3.82	2.98	2.69	2.24	4.36	2.81	2.54	1.41	0.77	**0.65**
100	20	7.68	11.57	5.90	3.93	2.77	2.17	2.11	1.75	4.82	3.01	2.04	1.29	0.71	**0.42**
200	10	10.02	15.09	7.98	5.79	3.94	3.50	3.14	2.18	2.89	1.94	2.32	1.23	0.63	**0.45**
200	20	8.30	11.92	6.59	4.83	3.21	2.76	2.37	1.58	3.48	2.25	2.24	1.02	0.52	**0.25**
500	20	8.68	12.37	7.27	5.63	4.08	3.54	3.08	1.34	2.34	1.61	2.04	0.97	0.52	**0.27**
Average		8.37	12.83	6.07	4.16	3.04	2.71	2.56	2.07	5.03	2.91	2.12	1.22	0.93	**0.82**

Table 22: Average CPU times and ARPT for the compared heuristics in the all distributions intervals in the benchmark adapted from Ruiz, Maroto e Alcaraz (2005). The values in bold are the best CPU times, in each line.

n	m	NEH	GH-BM	GH-BM2	$FRB4_{10}$	$FRB4_{30}$	$FRB4_{50}$	$FRB4_{70}$	FRB3	FJSRA	FJSRA-VND	RN_{10}	RN_{30}	RN_{50}	RN_{70}
20	5	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	10	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	20	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	5	**0.00**	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.01
50	10	**0.00**	0.01	0.00	0.01	0.02	0.02	0.02	0.03	0.00	0.00	0.01	0.02	0.02	0.02
50	20	**0.00**	0.02	0.00	0.02	0.04	0.04	0.04	0.06	0.00	0.00	0.02	0.04	0.04	0.04
100	5	**0.00**	0.03	0.00	0.01	0.04	0.05	0.06	0.09	0.08	0.09	0.02	0.04	0.06	0.06
100	10	**0.00**	0.07	0.01	0.04	0.08	0.12	0.14	0.21	0.09	0.10	0.04	0.09	0.13	0.14
100	20	**0.01**	0.17	0.02	0.08	0.19	0.28	0.33	0.46	0.09	0.10	0.09	0.20	0.28	0.32
200	10	**0.01**	0.64	0.04	0.17	0.40	0.61	0.80	1.75	1.64	1.80	0.17	0.45	0.68	0.88
200	20	**0.03**	1.46	0.09	0.38	0.96	1.46	1.89	3.92	1.65	1.83	0.39	1.00	1.52	1.94
500	20	**0.23**	24.21	0.75	3.47	9.29	14.66	19.90	73.91	65.99	72.73	3.35	9.62	15.46	20.74
Average		**0.02**	2.22	0.08	0.35	0.92	1.44	1.93	6.70	5.80	6.39	0.34	0.96	1.52	2.01
ARPT		**0.03**	0.72	0.16	0.53	1.01	1.25	1.37	2.32	1.19	1.29	0.58	1.01	1.22	1.33

Furthermore, a statistical test with $FRB4_{70}$, FRB3 and RN_{70} was generated to show that the averages are statistically different. The results from the statistical tests are presented in Table 23. From the tests we can see that the RN_{70} is statically better than the best heuristics from the literature in the benchmark from Ruiz, Maroto e Alcaraz (2005).

It is worth assessing the performance of our heuristic on instances with sequence-dependent setup times only (without no-idle machines). The ARPD and average CPU times for the compared heuristics are presented in Tables 24 and 25. The results show that this time around RN_{70} obtained the best ARPD values for all instance groups (0.34 compared 0.73 of FRB3). Therefore, we conclude that the proposed method generates high quality solutions even when no-idle machines are not considered.

3.3.7 Evaluation of the MILP model and the RN_x heuristic

As the RN_{70} heuristic presented the best results in the previous comparisons, we chose to compare it to the optimal solutions found by the MILP model presented in Section 2.3.1. Table 26 presents the results in terms of ARPD and percentage of optimal solutions found by the RN_{70}. The ARPD represents the distance of the solutions found by the RN_{70} relative to the optimal solution obtained by the MILP for the instance set. The results show that the RN_{70} found near optimum solutions, with an ARPD of 4.12 when we consider all the instances, and in the best case 3.34 for the SSD50 distribution interval. In addition, on average 4% of the solutions found by the RN_{70} heuristic are optimal. The best results are achieved when $n = 10$ and $m = 5$, where the percentage of optimal solutions found can reach up to 17% for the SSD50 and SSD100 distribution intervals. Thus, together with the results presented in the previous benchmarks, we can conclude that the RN_{70} heuristic obtained good solutions and sometimes optimal with excellent computational efficiency. Table 27 shows how the MILP CPU time requirements increased throughout increase over the different number of jobs and machines and the maximum CPU time for each set of instances. We can see that the average CPU time significantly as the number of jobs and machines increased, with up to 8643 seconds (2 hours and 24 minutes) in the worst-case scenario for the set of instances with $n = 20$, $m = 5$ and SSD50 distribution.

Table 23: Tukey test results of the best heuristics, with significance level of 95% in the benchmark from Ruiz, Maroto e Alcaraz (2005). The values in bold mean that there is a significant statistical difference between the algorithms in the first and second column.

n	Heuristic (I)	Heuristic (J)	Mean Difference (I-J)	Standard Error	Significance
20	$FRB4_{70}$	FRB3	0.228	0.086	**0.022**
		RN_{70}	0.456	0.086	**0.000**
	FRB3	$FRB4_{70}$	-0.228	0.086	**0.022**
		RN_{70}	0.228	0.086	**0.022**
	RN_{70}	$FRB4_{70}$	-0.456	0.086	**0.000**
		FRB3	-0.228	0.086	**0.022**
50	$FRB4_{70}$	FRB3	0.193	0.072	**0.021**
		RN_{70}	1.091	0.072	**0.000**
	FRB3	$FRB4_{70}$	-0.193	0.072	**0.021**
		RN_{70}	0.898	0.072	**0.000**
	RN_{70}	$FRB4_{70}$	-1.091	0.072	**0.000**
		FRB3	-0.898	0.072	**0.000**
100	$FRB4_{70}$	FRB3	0.326	0.065	**0.000**
		RN_{70}	1.326	0.065	**0.000**
	FRB3	$FRB4_{70}$	-0.326	0.065	**0.000**
		RN_{70}	1.000	0.065	**0.000**
	RN_{70}	$FRB4_{70}$	-1.326	0.065	**0.000**
		FRB3	-1.000	0.065	**0.000**
200	$FRB4_{70}$	FRB3	0.726	0.073	**0.000**
		RN_{70}	1.515	0.073	**0.000**
	FRB3	$FRB4_{70}$	-0.726	0.073	**0.000**
		RN_{70}	0.789	0.073	**0.000**
	RN_{70}	$FRB4_{70}$	-1.515	0.073	**0.000**
		FRB3	-0.789	0.073	**0.000**
500	$FRB4_{70}$	FRB3	1.316	0.095	**0.000**
		RN_{70}	1.733	0.095	**0.000**
	FRB3	$FRB4_{70}$	-1.316	0.095	**0.000**
		RN_{70}	0.417	0.095	**0.000**
	RN_{70}	$FRB4_{70}$	-1.733	0.095	**0.000**
		FRB3	-0.417	0.095	**0.000**

Table 24: ARPD results in the benchmark from Ruiz, Maroto e Alcaraz (2005) without no-idle machines. The values in bold are the best CPU times, in each line.

n	m	NEH	GH-BM2	$FRB4_{70}$	FRB3	FJSRA-VND	RN_{70}
20	5	4.79	3.04	1.61	1.30	3.27	**0.82**
20	10	3.93	2.44	1.57	1.06	3.64	**0.67**
20	20	2.84	1.80	1.07	0.78	3.36	**0.68**
50	5	4.99	3.28	1.40	0.95	2.09	**0.18**
50	10	4.15	3.02	0.98	0.67	3.01	**0.33**
50	20	3.33	2.51	0.76	0.44	3.62	**0.40**
100	5	5.61	4.36	1.50	1.17	1.87	**0.13**
100	10	4.30	3.36	0.95	0.51	2.14	**0.21**
100	20	3.43	2.63	0.64	0.55	2.77	**0.18**
200	10	4.61	3.86	1.10	0.58	1.24	**0.10**
200	20	3.42	2.85	0.83	0.29	1.79	**0.15**
500	20	3.64	3.22	1.32	0.41	0.68	**0.22**
Average		4.09	3.03	1.15	0.73	2.46	**0.34**

Table 25: Average CPU times in the benchmark from Ruiz, Maroto e Alcaraz (2005) without no-idle machines.

n	m	NEH	GH-BM2	$FRB4_{70}$	FRB3	FJSRA-VND	RN_{70}
20	5	0.00	0.00	0.00	0.00	0.00	0.00
20	10	0.00	0.00	0.00	0.00	0.00	0.00
20	20	0.00	0.00	0.00	0.00	0.00	0.00
50	5	0.00	0.00	0.01	0.01	0.00	0.01
50	10	0.00	0.00	0.01	0.01	0.00	0.01
50	20	0.00	0.00	0.03	0.03	0.01	0.03
100	5	0.00	0.00	0.04	0.05	0.02	0.05
100	10	0.00	0.01	0.09	0.11	0.04	0.11
100	20	0.01	0.02	0.25	0.28	0.10	0.29
200	10	0.01	0.04	0.68	1.01	0.26	0.75
200	20	0.03	0.09	1.59	2.64	0.69	1.90
500	20	0.22	0.65	14.61	47.63	5.33	16.07
Average		0.02	0.07	1.44	4.31	0.54	1.60

Table 26: APRD and percentage of optimum solutions for the proposed RN_{70} heuristic.

Distribution	n	m	ARPD	Percentage of Optimal Solutions
SSD50	10	5	2.88	17%
	10	10	2.92	7%
	15	5	3.61	0%
	15	10	3.06	0%
	20	5	4.22	0%
	Average		3.34	5%
SSD100	10	5	2.85	17%
	10	10	3.50	0%
	15	5	4.88	3%
	15	10	4.42	0%
	20	5	5.63	0%
	Average		4.26	4%
SSD125	10	5	4.56	10%
	10	10	3.23	7%
	15	5	6.23	0%
	15	10	3.97	3%
	20	5	5.88	0%
	Average		4.77	4%
$\overline{\text{Average}}$			4.12	4%

Table 27: Average and maximum CPU time for the MILP model.

Distribution	n	m	CPU Time	
			Average	Maximum
SSD50	10	5	1.07	1.93
	10	10	2.36	12.17
	15	5	19.47	106.81
	15	10	158.08	1284.62
	20	5	687.79	8643.28
	Average		173.75	2009.76
SSD100	10	5	0.79	1.84
	10	10	2.21	13.54
	15	5	10.68	55.13
	15	10	398.78	3671.78
	20	5	135.26	810.05
	Average		109.54	910.47
SSD125	10	5	0.86	2.03
	10	10	1.99	10.64
	15	5	15.05	71.40
	15	10	527.22	3124.45
	20	5	279.20	2808.88
	Average		164.86	1203.48

3.4 Conclusion

In this chapter, the scheduling problem in a mixed no-idle PFSP with sequence dependent setup times was studied. Considering a comprehensive literature review carried out on the no-idle PFSP, it was found that the mixed no-idle PFSP with setup dependent times has not yet been studied despite it being present in current production systems. In a mixed no-idle environment, not all machines need to be in continuous processing. As the problem had not yet been addressed in the literature, an MILP and ways to evaluate the makespan of a sequence were proposed. An acceleration method for assessing the makespan of an insertion neighbourhood was also developed.

A new constructive heuristic, called RN_x, is proposed in this chapter in order to provide the problem with a solution method that obtained good results with computational efficiency. For comparison purposes, the best heuristics of the no-idle flowshop problem with makespan criterion were adapted to the addressed problem.

Based on the statistical comparisons carried out, it was found that the FRB3 method offers the best performance among the main methods adapted in the literature. However, the analyses showed that the proposed RN_x method surpassed the FRB3 heuristic in terms of solution quality and computational efficiency. The statistical results also show that the means are significantly different. Therefore, the proposed heuristic can be considered as an important contribution to the state of the art in constructive heuristics for the no-idle PFSP problem and the variant studied in this chapter. In the next chapter, the mixed no-idle PFSP with total flowtime criterion will be presented.

4 THE MIXED NO-IDLE PFSP WITH SETUP TIMES AND TOTAL FLOWTIME MINIMIZATION

In this chapter, the total flowtime minimisation criterion is addressed given its relevance to the current production dynamics due to its relationship with minimising in-process inventory (LIU; REEVES, 2001). As this is the first time that this problem has been studied, the most efficient heuristics proposed for the no-idle and mixed no-idle PFSP problems with makespan criterion (denoted by $F_m|prmu, no-idle|C_{max}$ and $F_m|prmu, mixed\ no-idle|C_{max}$, according to Graham et al. (1979), as well as PFSP with makespan and total flowtime minimisation ($F_m|prmu|C_{max}$ e $F_m|prmu|\sum C_j$, respectively) were adapted and tested with the purpose of generating a basis of comparison for the proposed heuristics. Moreover, a method for calculating the makespan of a permutation sequence and an acceleration method for the insertion neighbourhood are provided in detail. The heuristics were compared through computational and statistical experiments in an extensive benchmark with 4500 instances. Experiments with the MILP formulation are presented in order to compared the proposed heuristics with optimal solutions for small sized problems instances. The results obtained demonstrate that the proposed heuristics offer high quality solutions with computational efficiency.

The chapter is organised as follows: Section 4.1 analyses the state of the art in heuristics. Section 4.2 proposes new heuristics. In Section 4.3, computational and statistical experiments are performed among the compared heuristics and the MILP model. Finally, Section 4.4 draws the main conclusions of the study.

4.1 Literature Review

This chapter focuses on the state of the art of constructive and improvement heuristics, as well as on the proposal of new heuristics to address the problem.

As mentioned earlier, the mixed no-idle PFSP with a sequence-dependent setup times has not yet been studied in the literature. Therefore, heuristics proposed for other related problems were adapted to our problem. Basically, the following topics were reviewed: heuristics for the PFSP with makepsan criterion ($F_m|prmu|C_{max}$), PFSP with total flowtime criterion ($F_m|prmu|\sum C_j$), no-idle and mixed no-idle PFSP with makespan criterion ($F_m|prmu,\ no-idle|C_{max}$, $F_m|prmu,\ mixed\ no-idle|C_{max}$).

4.1.1 Heuristics for the $F|prmu|C_{max}$ problem

The first studies on heuristics for the PFSP with the makespan criterion can be seen in Palmer (1965), Campbell, Dudek e Smith (1970), Gupta (1971), Dannenbring (1977), Hundal e Rajgopal (1988), Sevast'janov (1995) and Koulamas (1998). Certainly, the most

efficient heuristic for the problem is the well known NEH heuristic of Nawaz, Enscore e Ham (1983) and Fernandez-Viagas e Framinan (2014). Many extensions of this heuristic have been proposed and can be found in the literature (NAGANO; MOCCELLIN, 2002; FRAMINAN; LEISTEN, 2003; LOW; YEH; HUANG, 2004; KALCZYNSKI; KAMBUROWSKI, 2007b; KALCZYNSKI; KAMBUROWSKI, 2008; DONG; HUANG; CHEN, 2008; KALCZYNSKI; KAMBUROWSKI, 2009; RAD; RUIZ; BOROOJERDIAN, 2009; RIBAS; COMPANYS; TORT-MARTORELL, 2010; KALCZYNSKI; KAMBUROWSKI, 2011; FERNANDEZ-VIAGAS; FRAMINAN, 2014; ROSSI; NAGANO; NETO, 2016).

Studies by Rad, Ruiz e Boroojerdian (2009), Fernandez-Viagas e Framinan (2014) and Rossi, Nagano e Neto (2016), point out that currently, the best heuristics for the problem are: NEH from Nawaz, Enscore e Ham (1983), FRB3 and FRB4$_k$ from Rad, Ruiz e Boroojerdian (2009), NEH FF from Fernandez-Viagas e Framinan (2014). The FRB3 and FRB4$_k$ heuristics are both extensions of the NEH heuristic. In the FRB3 heuristic, a local search based on the insertion movements is integrated with the NEH. The principle behind the heuristic is to optimize the partials sequences generated by the NEH heuristic. The FRB4$_k$ limits the number of jobs selected for insertion with the parameter k. In this way, the method partially keeps the optimization of the partial sequences without losing computational efficient. The FRB3 and FRB4$_k$ presented better quality of solution when compared to the NEH heuristic in exchange of additional computational time. The NEH FF (FERNANDEZ-VIAGAS; LEISTEN; FRAMINAN, 2016) is also an extension that apply a tie-breaking method for the partial sequences generated by the NEH. The tie-breaking mechanism is based on the estimation of the idle times in the machines.

4.1.2 Heuristics for the $F|prmu|\sum C_j$ problem

Various studies have focused on studying PFSP with a total flowtime criterion, since its minimisation is important in many practical situations, especially when the objective is to reduce the inventory or the costs involved in maintaining it (RAJENDRAN; ZIEGLER, 1997). Also, several heuristics have been proposed (RAJENDRAN, 1993; WOO; YIM, 1998; LIU; REEVES, 2001; FRAMINAN; LEISTEN; RUIZ-USANO, 2002; FRAMINAN; LEISTEN, 2003; FRAMINAN; LEISTEN; RUIZ-USANO, 2005; NAGANO; MOCCELLIN, 2008; LAHA; SARIN, 2009). Pan e Ruiz (2013), Fernandez-Viagas e Framinan (2015) and Rossi, Nagano e Sagawa (2017) carried out an extensive computational comparison among the heuristics for the problem. According to the performed experiments, the main constructive heuristics are: RZ from Rajendran e Ziegler (1997), LR(x) from Liu e Reeves (2001), LR-NEH(x) from Pan e Ruiz (2013), FF(x) and FF-NEH(x) from Fernandez-Viagas e Framinan (2015) and FF-RN from Rossi, Nagano e Sagawa (2017). Regarding improvement heuristics, the following ones were highlighted: ICH1, ICH2 and ICH3 from Li, Wang e Wu (2009), PR1(x) from Pan e Ruiz (2013) FF-ICH1, FF-ICH2, FF-ICH3, FF-PR1 from Fernandez-Viagas e Framinan (2015).

Concerning constructive heuristics, the RZ was proposed by Rajendran e Ziegler (1997), and consists of two phases. First, the jobs are ordered using a new priority rule based on the lower bounds of the completions times of the jobs. Then, a method based on insertion movements is applied on the sequence generated by the first phase. The LR(x) from Liu e Reeves (2001) constructs the sequences by iteratively inserting unscheduled jobs in the final position of the sequence until all jobs have been scheduled. The parameter x controls how many sequences will be generated by the method, being the one with the lowest total flowtime chosen as solution. Pan e Ruiz (2013) developed the LR-NEH(x) that uses the LR(x) in conjunction to the NEH heuristic to construct the solution. Many heuristics, for total flowtime minimisation use the LR(x) method as an initial solution (ICH1, PR1) or to construct part of the sequence (LR-NEH(x)). That is, developing an improved version of the LR(x) could lead, indirectly, to improving the heuristics that use it. Along these lines, Fernandez-Viagas e Framinan (2015) developed a new version of the LR(x) which is denoted by FF(x). Based on computational comparisons, it was demonstrated that this new version surpasses the LR(x) heuristic, providing better solutions with less computational complexity. The LR-NEH(x) also was modified, replacing the LR(x) by the FF(x), and was denoted as FF-NEH(x). The FF-RN(x, y) from Rossi, Nagano e Sagawa (2017) is based on the FF-NEH(x) algorithm. The method performs a reinsertion of jobs in the partial sequences generated by the NEH heuristic, and similarly to FRB4$_k$, the parameter y limits the number of reinsertions.

Regarding improvement heuristics, Li, Wang e Wu (2009) proposed the ICH1, ICH2 and ICH3 methods. In these heuristics, local searches based on insertion and permutation movements are used to improve the solution generated by the LR(x) heuristic. In PR1(x) Pan e Ruiz (2013), a local search using insertion movements is applied in the solution given by the LR-NEH(x) method. Fernandez-Viagas, Leisten e Framinan (2016) proposed the FF-ICH1, FF-ICH2, FF-ICH3 and FF-PR1 heuristics, which are improved versions of the ICH1, ICH2 and ICH3 and PR1(x) heuristics, respectively. In these versions, the LR(x) and LR-NEH(x) heuristics are replaced by the new FF(x) and FF-NEH(x) methods. Fernandez-Viagas e Framinan (2015) shows that their proposals provided better results both in terms of quality of solution and computational efficiency. Thus, the best constructive heuristics from the literature are the RZ, FF(x), FF-NEH(x) and FF-RN(x, y), and for the improvement heuristics are the FF-ICH1(x), FF-ICH2(x), FF-ICH3(x) and FF-PRI(x).

4.1.3 Heuristics for the $F|prmu, no-idle|C_{max}$ and $F|prmu, mixed\ no-idle|C_{max}$ problems

Concerning the $F|prmu, no-idle|C_{max}$, problem, various studies have focused on developing exact methods (B&B, mixed integer linear programming - MILP, and polynomial exact algorithms) or on investigating the properties and variants of the

problem (ADIRI; POHORYLES, 1982; VACHAJITPAN, 1982; BAPTISTE; HGUNY, 1997; ČEPEK et al., 2000; SAADANI; GUINET; MOALLA, 2003; BAGGA, 2003; KAMBUROWSKI, 2004; NARAIN; BAGGA, 2005a; NARAIN; BAGGA, 2005b; KALCZYNSKI; KAMBUROWSKI, 2007a; GONCHAROV; SEVASTYANOV, 2009; NAGANO; ROSSI; TOMAZELLA, 2017; CHENG; SUN; YU, 2007; SUN et al., 2010; NG et al., 2011; SUN et al., 2012). Likewise, many other studies focus on developing heuristics and metaheuristics for the problem (WOOLLAM, 1986; KALCZYNSKI; KAMBUROWSKI, 2005; SAADANI; GUINET; MOALLA, 2005; PAN; WANG, 2008b; BARAZ; MOSHEIOV, 2008; PAN; WANG, 2008a; RUIZ; VALLADA; FERNÁNDEZ-MARTÍNEZ, 2009; TASGETIREN et al., 2011; DENG; GU, 2012; TASGETIREN et al., 2013b; TASGETIREN et al., 2013a; ZHOU; CHEN; ZHOU, 2014; SHEN; WANG; WANG, 2015).

According to studies conducted by Ruiz, Vallada e Fernández-Martínez (2009), Pan e Ruiz (2014), Nagano, Rossi e Tomazella (2017) and Nagano, Rossi e Martarelli (2018), currently the most efficient heuristics for this problem are: GH-BM2, FRB3 and FRB4$_k$ by Ruiz, Vallada e Fernández-Martínez (2009). The FRB3 and FRB4$_k$ heuristics are known for the $F|prmu|C_{max}$, whereas the GH-BM2 heuristic is an improvement of the GH-BM heuristic by Baraz e Mosheiov (2008). In the experiments GH-BM2 showed considerably better results when compared to the GH-BM.

4.2 Proposed heuristics

In this study, we developed three heuristics based on beam search. Beam-search-based heuristics combine the diversification of the population-based metaheuristics with the computational efficiency of the constructive heuristics (FERNANDEZ-VIAGAS; LEISTEN; FRAMINAN, 2016; FERNANDEZ-VIAGAS; RUIZ; FRAMINAN, 2017; FERNANDEZ-VIAGAS; VALENTE; FRAMINAN, 2018). In a beam search algorithm, partial sequences (also called nodes) are generated at each iteration by inserting jobs in the last position of the sequence. The best ranked N nodes generated are selected to be used in the next iteration. The method continues until nodes with complete sequences of n jobs are obtained; then the best ranked node is chosen to be the final solution of the method.

As FRB3 and FRB4$_k$ heuristics obtained excellent results in previous studies (RAD; RUIZ; BOROOJERDIAN, 2009; ROSSI; NAGANO; NETO, 2016), we decided to use their variants in conjunction with the beam-search concept presented previously. Moreover, we resorted to the idea of the LR-NEH(x) heuristic by Pan e Ruiz (2013), except for the fact that d jobs are inserted using a beam-search-based procedure and the remaining $n - d$ jobs are inserted by variants of the FRB3 and FRB4$_k$ heuristics. This combination resulted in two heuristics, H1(N) and H2(N, k), that use versions of the FRB3 and FRB4$_k$ heuristics, respectively, for constructing the remainder of the sequence. N denotes the number of nodes selected at each iteration; k is the number of jobs selected for reinsertion.

Additionally, an improvement heuristic, H3(N), was developed that carries out a local search, RZ from Rajendran e Ziegler (1997) in the final solution generated by the H1(N) method. The local search is based on the insertion neighbourhood and its high performance has already been demonstrated in various studies (FRAMINAN; GUPTA; LEISTEN, 2004; LI; WANG; WU, 2009; FERNANDEZ-VIAGAS; FRAMINAN, 2015). Before describing the proposed heuristics in more detail, it is important to define the index function used to evaluate the nodes generated by the heuristic.

4.2.1 The index function for nodes evaluation

In order to meet the constraints of the mixed no-idle PFSP with sequence-dependent setup and the total flowtime minimisation we modified the index present in the LR(x) heuristic from Liu e Reeves (2001). Formally, a node, denoted by η_l^v, is linked to a partial sequence π^v with l jobs, $\pi^v = \{\pi_1^v, \ldots, \pi_l^v\}$ and to a set of jobs that have not yet been sequenced, U_v. A node η_l^v can be branched generating other nodes, η_{l+1}^v, by inserting a job that has not yet been sequenced, $J_j \in U_v$, in the last position $l+1$ de π^v, resulting in $\pi^v = \{\pi_1^v, \ldots, \pi_l^v, \pi_{l+1}^v\}$, where $\pi_{l+1}^v = J_j$. Then, an initial node η_l^v is branched into other nodes, denoted by η_{l+1}^v, each one with a different job $J_j \in U_v$ in the last position of π^v. The principle behind the index is to evaluate three results from the insertion of job J_j in the last position: idle time generated, immediate effect of the job inserted in the last position and influence of the jobs that have not yet been sequenced.

The idle time of the generated node η_{l+1}^v is denoted by IT_l^v. However, this evaluation can lose relevance in mixed no-idle scenarios where most machines do not allow idleness. Thus, supposing a sequence $\pi^v = \{\pi_1^v, \ldots, \pi_l^v, \pi_{l+1}^v\}$ of node η_{l+1}^v, to calculate the completion times of the first l jobs of π^v, the normal condition studied in this work is considered, i.e. the mixed no-idle flowshop sequence-dependent setup, making it possible, in this case, to use the calculation method described in Subsection 2.3.2. However, for job J_j, inserted at position $l+1$, π_{l+1}^v, the completion times consider only the setup times, and all machines are considered regular ones. These completion times of job J_j in machine M_i, denoted by $F_{i,[l+1]}$, can be calculated using the expressions below.

When job J_j is inserted in the first position, $l = 1$:

$$\begin{cases} F_{1,[1]} = p_{1,[1]} \\ F_{i,[1]} = F_{i-1,[1]} + p_{i,[1]} \\ i = 2, \ldots, m \end{cases} \tag{4.1}$$

For all remaining positions:

$$\begin{cases} F_{1,[l+1]} = C_{1,[l]} + s^1_{[l],[l+1]} + p_{1,[l+1]} \\ F_{i,[l+1]} = \max\left(C_{i,[l]} + s^i_{[l],[l+1]}, F_{i-1,[l+1]}\right) + p_{i,[l+1]} \\ i = 2, \ldots, m \end{cases} \tag{4.2}$$

Note that the completion times of the job occupying position l ($C_{i,[l]}$) remain unchanged if we change the job which occupies position $l+1$ within the sequence. Therefore, it is possible to pre-calculate $C_{i,[l]}$ and use the same values to calculate the completion times of any job J_j that is inserted in position $l+1$. This enables us to quickly calculate the completion times $F_{i,[l+1]}$. After the completion times $F_{i,[l+1]}$ are calculated, the calculation of IT^v_l continues in the same way as is done in the LR(x) heuristic.

$$IT^v_l = (n-l-2) \cdot \sum_{i=2}^{m} \frac{m \cdot max\{F_{i-1,[l+1]} - C_{i,[l]} - s^i_{[l],[l+1]}, 0\}}{i + l \cdot (m-i)/(n-2)} \tag{4.3}$$

The immediate effect of inserting job J_j in the last position $l+1$, FT^v_l, is evaluated using the completion time of job J_j that occupies the last position, $l+1$, in the last machine M_m, $F_{m,[l+1]}$. The aim is to choose those nodes with the lowest added completion time generated by the insertion of the job, thus reducing the total flowtime.

$$FT^v_l = F_{m,[l+1]} \tag{4.4}$$

The influence of jobs that have not yet been sequenced, denoted by AT^v_l, is estimated by the completion time of the last machine of an artificial job, μ, which is inserted at the next position after job J_j, $F_{m,\mu}$. That is, a partial sequence is generated with $l+2$ jobs $\pi^v = \{\pi^v_1, \ldots, \pi^v_l, \pi^v_{l+1}, \pi^v_{l+2}\}$, where the last position is occupied by the artificial job μ, $\pi^v_{l+2} = \mu$. Artificial job μ is assigned with processing and setup times in the machines. Thus, the setup time between the candidate job and the jobs that have not yet been sequenced can be minimised, resulting in a better final solution. The processing times, $p_{i,\mu}$, and the setup times, s^i_μ, of the artificial job μ can be determined according to the expressions below.

$$p_{i,\mu} = \frac{\sum_{J_h \in U, J_h \neq J_j} p_{i,h}}{n-l-1} \tag{4.5}$$

$$s^i_\mu = \frac{\sum_{J_h \in U, J_h \neq J_j} s^i_{j,h}}{n-l-1} \tag{4.6}$$

The influence of the jobs that have not yet been sequenced, denoted by AT^v_l, can be calculated using the expressions below.

$$\begin{cases} F_{1,\mu} = F_{1,[l+1]} + s^1_\mu + p_{1,\mu} \\ F_{i,\mu} = \max\left(F_{i,[l+1]} + s^i_\mu, F_{i-1,\mu}\right) + p_{i,\mu} \\ i = 2, ..., m \end{cases} \tag{4.7}$$

$$AT^v_l = F_{m,\mu} \tag{4.8}$$

Finally, index φ_l^v, that evaluates the generated nodes, η_{l+1}^v, is defined below. The N selected nodes are those that present the lowest value of φ_l^v.

$$\varphi_l^v = IT_l^v + FT_l^v + AT_l^v \tag{4.9}$$

4.2.2 The H1(N), H2(N, k) and H3(N) heuristics

The first heuristic, H1(N), initially generates n partial sequences (nodes) with only one job, resulting in an initial set of nodes $\eta_0^h = \{\eta_0^1, \ldots, \eta_0^n\}$. Then, the set of nodes η_0^h is evaluated using the index φ_0^h ($h = 1, \ldots, n$). The N nodes that obtained the best (lowest) values of φ_0^h are selected for the next step. For each node selected, η_0^v ($v = 1, \ldots, N$), new partial sequences are generated by inserting jobs $J_j \in U^v$ that have not yet been sequenced of node η_0^v at the last position of the sequence π^v. This results in a new set of nodes, $\eta_1^h = \{\eta_1^1, \eta_1^2, \ldots\}$, each one linked to a partial sequence with two jobs, $\pi^v = \{\pi_1^v, \pi_2^v, \ldots\}$. As each node will generate $n-1$ new nodes, there will be $N \cdot (n-1)$ nodes to be evaluated at this iteration. All generated nodes are evaluated using the index φ_1^v. The N best nodes are selected for the next branching. The method continues iteratively until nodes with partial sequences of d jobs are obtained. Then, the partial sequence π^v of node η_d^v with the lowest total flowtime is selected to be the solution for this part of the method. The rest of the sequence is constructed using a variant of the FRB3 heuristic, as described below.

At the end of this beam-search-based procedure, there will be a sequence π with d jobs and $n-d$ jobs still to be sequenced that belong to the set denoted by U. The jobs in U are sorted by a non-descending order of the sum of the processing times (SPT rule). The first job of the order is inserted in the best position of the π sequence. Afterwards, a procedure, based on the insertion neighbourhood, is performed. Instead of reinserting the jobs one by one as the FRB3 heuristic does, our proposal does this reinsertion in pairs of adjacent jobs, π_l and π_{l+1}. Rossi, Nagano e Neto (2016) already demonstrated that this kind of reinsertion in pairs of jobs results in a better optimisation of partial sequences. During this movement, the jobs from the pair are removed from the sequence and the first job, π_l, is tested in all positions of the sequence, and the best position is selected for insertion; then, the second job, π_{l+1} is considered analogously. Afterwards, the second pair, $\{\pi_{l+2}, \pi_{l+3}\}$, is reinserted and so forth, $\{\pi_{l+4}, \pi_{l+5}\}$, until the last pair, $\{\pi_{n-1}, \pi_n\}$, is done. When the reinsertion is finished, the method inserts the second job from the initial ordering, and then the same reinsertion movements in pairs, as explained above. The next jobs of the initial ordering are considered in the same way. The method continues until the complete sequence with n jobs is obtained.

However, it is clear that the reinsertion procedure at each iteration in the heuristic H1(N) is computationally intensive, since, at each iteration $n/2$ pairs of jobs are reinserted and each job from the pair has to be tested in all positions of the sequence. In the heuristic H2(N, k) , the number of reinsertions to be performed is limited, following the

same principle of the FRB4$_k$ heuristic. Thus, if a job of the initial order was inserted in position j, it will be considered the pairs of jobs around this position, $\{\pi_{j-k}, \pi_{j-k+1}\}$, $\{\pi_{j-k+2}, \pi_{j-k+3}\}$, ..., $\{\pi_{j+k-1}, \pi_{j+k}\}$. This allows the method to keep the optimisation of the partial sequences without affecting the computational efficiency.

Finally, the H3(N) heuristic carries out a local search RZ, based on insertion neighbourhood, in the solution generated by the H1(N). The insertion neighbourhood is repeated until the solution does not get any better. The jobs to be reinserted in the sequences follow the order of a reference sequence π^{ref}, which is equal to the best solution found until that moment. The pseudocode presented in Algorithm 2 shows the beam search procedure used in the proposed heuristics for generating the sequence π with d jobs. The heuristics H1(N), H2(N, k) and H3(N) are described in detail in Algorithms 3, 4 and 5, respectively.

Algorithm 2 Beam search based procedure.

U is the set of unscheduled jobs, $U = \{J_1, J_2, ..., J_n\}$.
U_j denotes the job occupying the jth position in the set U.
for $h = 1$ to n **do**
  Generate the node η_0^h.
  $\pi^h = \{J_h\}$
  $U^h = U - J_h$
  Evaluate the node η_0^h using the index function φ_0^h.
end for
Order the generated nodes η_0^h in non-desceding order of φ_0^h.
Select the N first ranked nodes η_0^h to be the new set of nodes $\eta_0^v = \{\eta_0^1, \eta_0^2, ..., \eta_0^N\}$.
for $l = 1$ to $d - 1$ **do**
  $h = 1$
  for $v = 1$ to N **do**
    for $j = 1$ to $n - l$ **do**
      Generate the node η_{l+1}^h from η_l^v.
      Insert the job U_j from U^v in the $l + 1$ position of π^v, resulting π^h.
      Evaluate the node η_{l+1}^h using the index function φ_l^h.
      $U^h = U^v - U_j$
      $h = h + 1$
    end for
  end for
  Order the generated nodes η_{l+1}^h in non-desceding order of φ_l^h.
  Select the N first ranked nodes to be the new set of nodes η_{l+1}^v.
end for
Select the sequence π^v of the node η_d^v that results in the lowest $\sum C_j(\pi^v)$.
$\pi = \pi^v$
$U = U^v$
return $\pi = \{\pi_1, ..., \pi_d\}$ and U.

Algorithm 3 H1(N) heuristic

Call the beam search based procedure (Algorithm 2).
Order the jobs in U according to the lowest value of the sum of processing times, resulting in $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_{n-d}\}$.
for $l = 1$ to $n - d$ **do**
  Insert job α_l in π in the position that results in the lowest $\sum C_j$.
  for $j = 1$ to $l + d - 1$, step $j = j + 2$ **do**
    $\pi' = \pi$
    Remove the jobs π'_j and π'_{j+1} from π'.
    Insert the job π'_j in the position of that results in the lowest $\sum C_j$.
    Insert the job π'_{j+1} in the position of that results in the lowest $\sum C_j$.
    if $\sum C_j(\pi') < \sum C_j(\pi)$ **then**
      $\pi = \pi'$
    end if
  end for
end for
return $\pi = \{\pi_1, ..., \pi_n\}$.

Algorithm 4 H2(N) heuristic

Call the beam search based procedure (Algorithm 2).
Order the jobs in U according to the lowest value of the sum of processing times, resulting in $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_{n-d}\}$.
for $l = 1$ to $n - d$ **do**
  Insert job α_l in π in the position b that results in the lowest $\sum C_j$.
  for $j = \max(1, b - k)$ to $\min(l + d - 1, b + k)$, step $j = j + 2$ **do**
    $\pi' = \pi$
    Remove the jobs π'_j and π'_{j+1} from π'.
    Insert the job π'_j in the position of that results in the lowest $\sum C_j$.
    Insert the job π'_{j+1} in the position of that results in the lowest $\sum C_j$.
    if $\sum C_j(\pi') < \sum C_j(\pi)$ **then**
      $\pi = \pi'$
    end if
  end for
end for
return $\pi = \{\pi_1, ..., \pi_n\}$.

Algorithm 5 H3(N) heuristic

π =H1(N) (Algorithm 3)
Improvement = true
while Improvement = true **do**
  $\pi^{ref} = \pi$
  for $j = 1$ to n **do**
    Improvement = false
    $\pi' = \pi$
    Remove job π_j^{ref} from π'.
    Insert the job π_j^{ref} in the position of π' that results in the lowest $\sum C_j$.
    if $\sum C_j(\pi') < \sum C_j(\pi)$ **then**
      $\pi = \pi'$
      Improvement = true
    end if
  end for
end while
return $\pi = \{\pi_1, ..., \pi_n\}$.

4.3 Computational and statistical experiments

4.3.1 Instances generation

In this study, we used an adaptation of the benchmark proposed by Pan e Ruiz (2014) for the mixed no-idle PFSP problem. We modified the set of tests in order to consider the sequence-dependent setup times of the sequence between jobs in regular machines. We also generated a set of instances to compare the proposed heuristics with the MILP formulation in small sized problems instances.

The benchmark from Pan e Ruiz (2014) consists of seven groups with different mixed no-idle scenarios. The problems are generated through combinations from a number of jobs $n = \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$ and a number of machines $m = \{10, 20, 30, 40, 50\}$, making a total of 50 possible combinations and five replications for each combination ($50 \times 5 = 250$ problems per group). The processing times of the jobs were generated using a uniform distribution within the range $U[1, 99]$. The groups have the following mixed no-idle scenarios:

- Group 1: Only the first 50% of machines are no-idle.
- Group 2: The first 50% machines are regular, the remaining are no-idle machines
- Group 3: The machines alternate between regular and no-idle.
- Group 4: 25% of machines are randomly no-idle.
- Group 5: 50% of machines are randomly no-idle.
- Group 6: 75% of machines are randomly no-idle.

For each group of problems, the setup time between jobs only for the regular machines was considered. Three groups of distribution were used: SSD-50, SSD-100, SSD-125. We used the processing times generated by Pan e Ruiz (2014)

- SSD-50: setup times were generated according to a uniform distribution in the range $U[1, 49]$.
- SSD-100: setup times were generated according to a uniform distribution in the range $U[1, 99]$.
- SSD-125: setup times were generated according to a uniform distribution in the range $U[1, 124]$.

As there are $6 \times 250 = 1500$ problems with different mixed no-idle scenarios, the result is a total of $3 \times 1500 = 4500$ test problems when the three distributions of setup time are considered.

For the MILP formulated in Section 2.3.1 we set a maximum elapsed CPU time limit of four hours to optimally solve the problems. With this time termination criterion, the MILP can optimally solve problems with up to 20 jobs and 5 machines. Thus, for the MILP evaluation we considered the following combination between number of jobs and machines $\{n, m\} = \{10, 5\}$, $\{10, 10\}$, $\{15, 5\}, \{15, 10\}$, $\{20, 5\}$. Five replication were generated for each combination with processing times generated using the uniform distribution [1, 99] . The same six mixed no-idle groups used in the previous benchmarks were used. For the setup times generation we used the distributions SSD-50, SSD-100 and SSD-125. With this settings, $5 \times 5 \times 3 \times 6 = 450$ instances were created.

4.3.2 Compared heuristics

Based on the literature review, we identified the heuristics that could be best adapted to our problem. The methods chosen for comparing the classic permutation flowshop problem ($F_m|prmu|C_{max}$), PFSP with total flowtime criterion $F_m|prmu|C_j$ and no-idle and mixed no-idle PFSP with makespan criterion $F_m|prmu,\ no-idle|C_{max}$ are listed below. These heuristics from the literature were compared with those proposed in this work. Parameters were selected considering those that obtained the best quality solution and computational efficiency in the experiments performed.

- Proposed heuristics.
  - H1(N), $N = \{10, 20\}$.
  - H2(N, k), $k = \{5, 10\}$, $N = \{10, 20\}$.
  - H3(N), $N = \{10\}$.

- Heuristics adapted from the PFSP with makespan criterion problem, $F|\ prmu|C_{max}$. The heuristics were adapted taking into account the non-descending sum of the processing times (Shortest Processing Time - SPT) initial ordering rather than the non-ascending order (Longest Processing Time - LPT), since it was demonstrated that the SPT ordering gives better results when the total flowtime minimisation criterion is under consideration. Although the NEH FF proposed by Fernandez-Viagas e Framinan (2014) showed good performance, it is not possible to adapt the NEH FF heuristic to our problem as the heuristic has procedures based on specific properties of the $F|prmu|C_{max}$ problem; thus, this NEH FF was not considered in this computational experiment.

  - NEH from Nawaz, Enscore e Ham (1983): The heuristic has two phases. In the first phase, jobs are ordered following the LPT rule. As mentioned, in our adapted version, the SPT initial ordering was used instead of the LPT. In the second phase, at each iteration, a job ordered in the first phase is evaluated in all positions of the sequence, and the best position is chosen. The next job from the initial order is considered analogously, and so forth until the n jobs have been sequenced.
  - FRB3 from Rad, Ruiz e Boroojerdian (2009): The FRB3 heuristic carries out a local search based on using the insertion neighbourhood in the partial sequences generated by the NEH heuristic. The method performs reinsertions of all jobs from the sequences in all possible positions after each NEH iteration.
  - FRB4$_k$ from Rad, Ruiz e Boroojerdian (2009): in FRB4$_k$ a limited local search is carried out, where $\pm k$ jobs that are positioned around the job recently inserted by the NEH heuristic are selected be to be reinserted in the sequence.

- Heuristics adapted from the PFSP with total flowtime criterion, $F_m|prmu|\ \sum C_j$. The improvement heuristics ICH2, ICH3, FF-ICH2 and FF-ICH3 use intensive local searches based on permutation movements of pairs of jobs. It is worth highlighting that the acceleration method (Subsection 2.3.4), which allows a significant reduction of computational complexity, cannot be used to evaluate the total flowtime resulting from the permutation of jobs, since it is targeted only for the insertion neighbourhood. As a consequence, these methods become computationally inefficient and therefore, were not selected for adaptation and comparison. As the heuristics FF(x), FF-NEH(x), FF-ICH1(x) and FF-PR1(x) are improved and more recent versions of heuristics LR(x), LR-NEH(x), ICH1 and PR1, only the first group of heuristics was selected for adaptation.

  - RZ from Rajendran e Ziegler (1997): This heuristic has two phases. In the first phase, m sequences are generated by ordering the jobs J_j ($j = 1, \ldots, n$) in

non-descending order of T_j (Expression 4.10), and the best solution is chosen as seed sequence. From Expression 4.10, note that each sequence is generated by an iteration of k ($k = 1, \ldots, m$), as each iteration generates a distinct ordering for the jobs, and consequently different sequences. In our adaptation we maintained this method to generate the seed sequence for the RZ heuristic.

$$T_j = \sum_{i=k}^{m}(m - i + 1) \cdot p_{i,j} \quad k = 1, ..., m \tag{4.10}$$

In the second phase, the seed sequence is improved by inserting jobs within partial sequences, which are successively obtained.

– FF(x) from Fernandez-Viagas e Framinan (2015): First, this heuristic generates an initial ordering of the jobs according to a non-descending order of an index function, $\xi'_{j,k}$, which takes into consideration the idle time of the machines. The first job of the initial ordering is chosen to occupy the first position of the sequence. Afterwards, at each iteration, all jobs J_j belonging to the set of jobs that have not yet been sequenced ($J_j \in U$) are tested in position $k + 1$ of the current sequence. The candidate job with the lowest index value $\xi'_{j,k}$ is permanently inserted in position $(k + 1)$ of the sequence. The procedure ends when n jobs have been sequenced. Heuristic FF(x) can generate multiple complete sequences by choosing a different job to occupy the first position of the sequence. From this choice, the method generates a complete sequence using the same procedure described previously. Parameter x defines the number of complete sequences that will be generated by the heuristic. The $\xi'_{j,k}$ index from a candidate job J_j in the $k + 1$ position can be calculated as the following expressions.

$$\xi'_{j,k} = \frac{(n - k - 2)}{a} \cdot IT'_{j,k} + AT'_{j,k} \tag{4.11}$$

$$IT'_{j,k} = \sum_{i=2}^{m} \frac{m \cdot max\{C_{i-1,j} - C_{i,j}, 0\}}{i - b + k \cdot \dfrac{m - i + b}{n - 2}} \tag{4.12}$$

$$AT'_{j,k} = C_{m,j} \tag{4.13}$$

As this study considers the existence of setup times and also no-idle machines, we adapted the $\xi'_{j,k}$ index in order to consider the conditions addressed in this chapter. For the first k jobs of the partial sequence, the completion times are calculated taking into account the existence of set-up times and the mixed no-idle condition. However, only for the completion times of candidate job J_j is it considered that all machines are regular, as well as the setup existence. Therefore, the $IT'_{j,k}$, component that measures idleness can be evaluated even when most are no-idle machines. As stated in Section 4.1, (FERNANDEZ-VIAGAS; FRAMINAN, 2015) replaced the procedure LR(x), used by several

existing heuristics, by the new method FF(x), resulting in better versions of the heuristics LR-NEH(x) and PR1(x) from Pan e Ruiz (2013) and ICH1, ICH2 and ICH3 from Li, Wang e Wu (2009), called: FF-NEH(x), FF-PR1(x) and FF-ICH1, respectively, which are detailed below.

- FF-NEH(x): The FF-NEH(x) is an improved version of the LR-NEH(x) from Pan e Ruiz (2013), which replaces the LR(x) heuristic by the FF(x). The heuristic first generates a partial sequence of d jobs using the FF(x); then the remaining $n - d$ jobs are inserted into the partial sequence through the NEH heuristic.
- FF-PRI1(x): This method is an improvement heuristic based on the PR1(x) Pan e Ruiz (2013). The heuristic improves each of the solutions generated by the FF-NEH(x) method using the iRZ local search, which is based on insertion neighbourhood movements.
- FF-ICH1(x): The FF-ICH1(x) replaces the LR(x) heuristics used in the ICH1 by the FF(x) from Fernandez-Viagas e Framinan (2015). The ICH1 is a improvement heuristics that combines the local search based on insertion movements in conjunction with an efficient iterative method that is repeated until the solution is no longer improved or until a certain stop criterion is met.
- FF-RN(x, y) from Rossi, Nagano e Sagawa (2017): This method is based on the FF-NEH(x) algorithm from Fernandez-Viagas e Framinan (2015), with the difference that a reinsertion of jobs in the partial sequences generated by the NEH heuristic is performed. In order to make the method computationally viable, the authors limit the selection of y jobs to be reinserted at each iteration. The jobs are selected for reinsertion using an weight index W', being the job to be prioritized the one that results in the lowest total flowtime when removed from the current sequence.

- Heuristics adapted from the no-idle and mixed no-idle PFSP with makespan criterion, $F_m|prmu,\ no-idle|C_{max}$ and $F_m|prmu,\ mixed\ no-idle|C_{max}$. The initial LPT ordering of the GH-BM2 heuristic was replaced by the SPT rule as in this study we aim to minimise the total flowtime but not the makespan.
  - GH-BM2 from Ruiz, Vallada e Fernández-Martínez (2009): This heuristic is a better version of the heuristic from Baraz e Mosheiov (2008). The GH-BM has two phases. In the first phase, the heuristic adds jobs at the end of the sequence, selecting the one that results in the lowest makespan when inserted in the last position. In the second phase, pairs are swapped, testing all the possible matches and swapping them, which result in a lower makespan value. The GH-BM2 replaced the first phase of the heuristic by the NEH from Nawaz,

Enscore e Ham (1983) and the pairwise exchange procedure was carried out in the second phase for an insertion neighbourhood.

4.3.3 Performance measures

The performance measures used to compare the heuristics were: quality solution and computational efficiency. The quality solution was evaluated by the relative deviation (DR_h^t) of the heuristic h in problem t, and can be calculated by the expression below.

$$DR_h^t = 100 \cdot \frac{\sum C_j\left(\pi_h\right)^t - \sum C_j(\pi^*)^t}{\sum C_j(\pi^*)^t} \tag{4.14}$$

where $\sum C_j\left(\pi_h\right)^t$ is the total flowtime provided by the sequence π_h through the heuristic h for problem t. $\sum C_j(\pi^*)^t$ is the best solution found among all heuristics compared for problem t. It can be seen that as the lower is the value of DR_h^t, the closer the heuristic's solution will be to the best result found. The mean average of the relative values ($ARPD_h$) of a heuristic h in conjunction with N problems can be calculated by the expression below:

$$ARPD_h = \frac{\sum_{t=1}^{N} DR_h^t}{N} \tag{4.15}$$

To assess the computational efficiency, the average time in seconds was used (average CPU - ACPU). All compared heuristics were implemented in C++, compiled using Intel C++ and executed in an Intel Xeon E5-2680 @ 2.7 GHz with 16 GB RAM memory. To solve the MILP model we used the IBM CPLEX Optimization Studio (version 12.8) with Python Application Programming Interface (API).

4.3.4 Parameter tunning for the H1(N), H2(N, k) and H3(N)

The heuristics H1(N), H2(N, k) and H3(N) have an important parameter d that controls the number of jobs that will be inserted through the beam-search-based procedure. Tests with values $d = \{n/4, n/2, 3n/4\}$ were carried out aimed at evaluating the heuristics behavior in terms of their quality solution (Average Relative Percentage Deviation - ARPD) and computational efficiency (ACPU). As the H3(N) heuristic uses the H1(N) heuristic for generating the initial solution, only the heuristics H1(N) and H2(N, k) were tested. The heuristics were tested with the number of nodes $N = \{10, 20\}$ and the number of reinsertions $k = \{5, 10\}$. Table 28 present the results in ARPD and ACPU, considering the average resulting from the possible combinations of $N = \{10, 20\}$ and $k = \{5, 10\}$.

Table 28: Parameter testing with different values for d.

Heuristic	d value	ARPD	ACPU
H1	$n/4$	0.75	218.43
	$n/2$	0.79	204.86
	$3n/4$	1.24	163.27
H2	$n/4$	4.15	18.46
	$n/2$	4.17	19.65
	$3n/4$	4.79	17.20

In heuristic H1(N), the tests show very similar ARPDs when $d = \{n/4, n/2\}$ (0.75 and 0.79), although the ACPU is considerably larger for $d = n/4$ (218.4 against 204.8 seconds). Thus, the slightly lower ARPD does not worth the worse computational efficiency when $d = n/4$. For $d = 3n/4$, the ACPU is smaller (163.2 seconds), and the ARPD becomes considerably worse (1.24). This shows that the beam-search-based procedure is computationally efficient because the heuristic speeds up when the number of jobs inserted through the beam-search procedure increases. Therefore, for heuristics H1(N) and H3(N), the best results are obtained when $d = n/2$ and this value will be used for the subsequent experiments. The results are similar for heuristic H2(N, k), where for $d = \{n/4, n/2\}$, the ARPDs are close (4.15 and 4.17), respectively and $d = \{3n/4\}$ with ACPU smaller (17.2 seconds), although the ARPD is worse (4.79). Thus, for heuristic H2(N, k), the best performance considering both ARPD and ACPU is also achieved when $d = n/2$.

4.3.5 Comparison

The results of the heuristics comparison (Subsection 4.3.2) in the set of test problems (Subsection 4.3.1) are presented in Table 29, in terms of solution quality (ARPD) and computational efficiency (ACPU). The best ARPD results achieved were through the improvement heuristic H3(10) with 0.63, although at a ACPU of 221 seconds. This was a result already expected, since the H3(N) heuristic applies a local search based on insertion neighbourhood in the solution generated by the H1(N) heuristic; this results in better quality solutions in exchange for additional computational time. After that, the best results are obtained by the proposed heuristic, H1(N) ($N = \{10, 20\}$), with ARPDs 0.78 and 0.79. After that, the one with the best performance was the heuristic adapted from the literature, FRB3, with ARPD of 1.94 and ACPU of 198.5 seconds. However, the proposed method, H1(20) obtained ARPD considerably smaller (0.78), with ACPU very close to 208 seconds, respectively. The proposed heuristic, H2(N, k) $N = \{10, 20\}$ and $k = \{5, 10\}$) is the third best, with particular emphasis on H2(20, 10) that obtained ARPD of 3.65 and ACPU of 25 seconds. The heuristic H2(N, k) obtained worse solutions when compared to heuristics H1(N) and FRB3, although its ACPU is much smaller, resulting in a good trade-off between ARPD and ACPU. Given these results, heuristic H2(10, 10)

Table 29: ARPD and ACPU for the compared heuristics.

Heuristics	ARPD				ACPU
	SSD50	SSD100	SSD125	Average	
H3(10)	0.57	0.65	0.69	0.63	221.07
H1(20)	0.79	0.80	0.76	0.78	207.64
H1(10)	0.76	0.79	0.82	0.79	202.09
FRB3	1.59	2.00	2.22	1.94	198.47
H2(20, 10)	3.83	3.65	3.46	3.65	25.09
H2(10, 10)	3.86	3.76	3.62	3.75	20.22
FF-RN(5, 10)	4.47	4.50	4.44	4.47	59.61
H2(20, 5)	4.90	4.56	4.28	4.58	19.21
$FRB4_{10}$	4.33	4.73	4.90	4.65	14.71
H2(10, 5)	4.91	4.72	4.46	4.70	14.08
FF-RN(1, 10)	5.31	5.43	5.35	5.37	11.29
FF-RN(5, 5)	5.69	5.50	5.41	5.54	45.65
$FRB4_5$	5.29	5.61	5.91	5.60	8.71
FF-PR1(10)	5.76	6.14	5.90	5.93	41.57
FF-PR1(5)	5.77	6.16	5.95	5.96	30.92
FF-PR1(1)	5.77	6.17	6.01	5.98	20.52
FF-ICH1(5)	5.84	6.64	6.55	6.34	31.93
FF-ICH1(10)	5.84	6.64	6.55	6.34	42.43
FF-ICH1(1)	5.84	6.64	6.55	6.34	24.70
FF-RN(1, 5)	6.59	6.43	6.33	6.45	8.54
GH-BM2	6.95	7.44	7.68	7.36	2.52
NEH FT	9.26	9.48	9.66	9.47	0.79
FF-NEH(10)	14.22	10.33	9.06	11.21	5.78
FF-NEH(5)	14.60	10.70	9.44	11.58	3.00
RZ	11.99	14.05	14.85	13.63	2.06
FF(10)	25.99	18.42	16.11	20.18	0.44
FF(5)	26.57	18.94	16.62	20.71	0.22

outperformed heuristic $FRB4_k$ ($k = \{5, 10\}$) adapted from the literature, which obtained ARPD of 4.65 and 5.60, respectively. No significant change was observed on the ARPD of heuristics H1(N), H2(N, k) and H3(N), as the distribution setup times changed.

It is worth noticing that the heuristics adapted from the $F|prmu|\sum C_j$ problem (RZ, FF(x), FF-NEH(x), $x = \{5, 10\}$, FF-ICH1(x) and FF-PR1(x), FF-RN(x, y)) did not achieve good results, even considering that the problem in this study also addressed the total flowtime minimisation. Among the methods adapted from this problem, the best heuristic was the FF-RN(5, 10) with ARPD of 4.47 and ACPU of 59.6 seconds. The heuristic GH-BM2, adapted from the $F|no-idle|C_{max}$ problem, proved to be computationally very efficient with ACPU of only 0.8 seconds, while at the same time providing reasonably quality solutions with ARPD of 7.36. It can be observed that heuristics FF-NEH(x) and FF(x) benefit from the relevance increase of the setup times in the problem. Note that for FF-NEH(10) in the SSD50 distribution, the ARPD is 14.22 and for the SSD125 set,

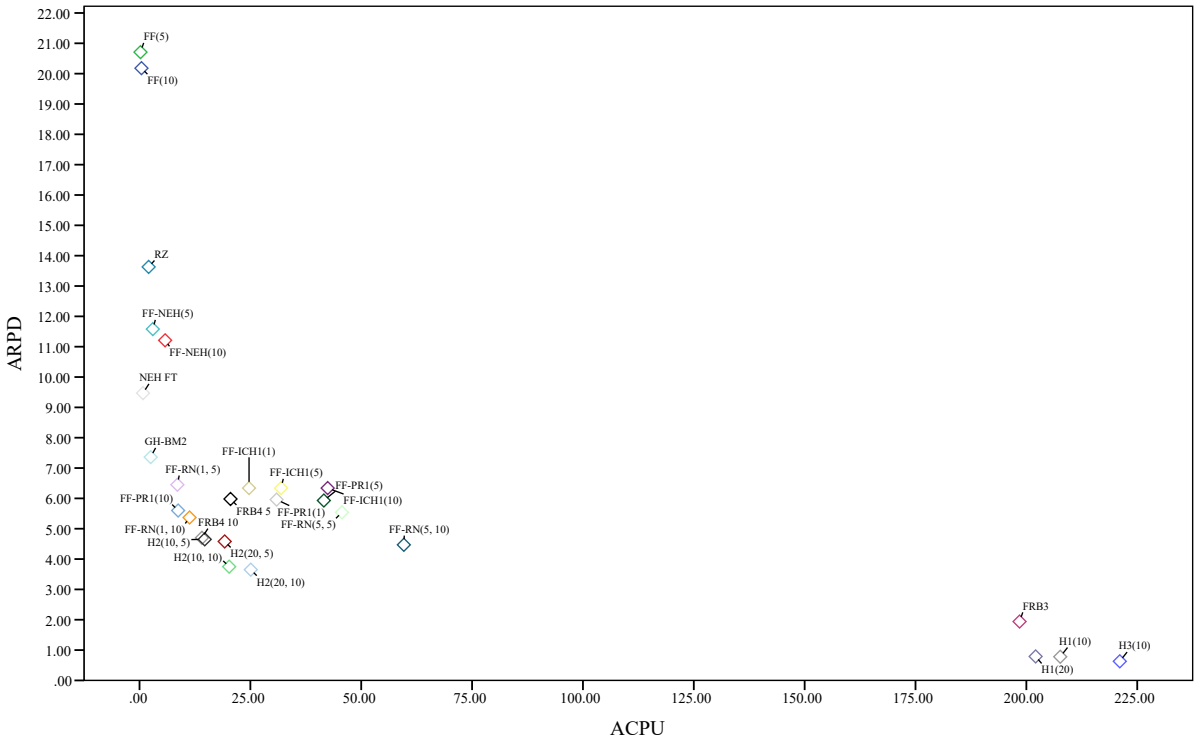

Figure 12: ARPD vs ACPU for the compared heuristics.

the ARPD is 9.06; i.e., a relevant difference in performance. For the other heuristics, no significant change was observed in the results as the setup times distribution were changed. The results are shown in Figure 12, where it can be observed that the proposed heuristics H1(N), H2(N, k) and H3(N) are dominants in terms of ARPD and ACPU. The results, in more detail, with a variation in the number of jobs n, are presented in Tables 30 and 31.

Table 30: ARPD for each set of problems arranged by the number of jobs.

Heuristic	Number of jobs										Average
	50	100	150	200	250	300	350	400	450	500	
RZ	9.62	11.63	12.70	13.28	13.93	14.40	14.73	15.43	15.14	15.44	13.63
NEH FT	8.43	9.00	9.14	9.42	9.64	9.58	9.68	10.07	9.75	9.95	9.47
GH-BM2	5.76	6.47	6.85	7.29	7.54	7.58	7.77	8.18	7.95	8.16	7.36
$FRB4_5$	4.26	4.94	5.23	5.51	5.73	5.79	5.96	6.24	6.11	6.25	5.60
$FRB4_{10}$	3.36	4.02	4.25	4.62	4.77	4.85	4.96	5.27	5.18	5.27	4.65
FRB3	2.27	2.01	2.05	1.94	1.89	1.90	1.83	1.93	1.81	1.74	1.94
FF(5)	18.24	19.71	20.26	20.51	21.31	21.19	21.40	21.82	21.30	21.36	20.71
FF(10)	17.33	19.02	19.65	19.98	20.81	20.73	20.94	21.40	20.92	20.99	20.18
FF-NEH(5)	9.77	10.78	11.10	11.38	11.74	11.94	12.07	12.50	12.19	12.32	11.58
FF-NEH(10)	9.10	10.28	10.69	11.02	11.42	11.64	11.76	12.20	11.90	12.04	11.21
FF-ICH1(1)	4.51	5.32	5.92	6.27	6.36	6.63	6.90	7.05	7.11	7.37	6.34
FF-ICH1(5)	4.51	5.32	5.92	6.27	6.36	6.63	6.90	7.05	7.11	7.37	6.34
FF-ICH1(10)	4.51	5.32	5.92	6.27	6.36	6.63	6.90	7.05	7.11	7.37	6.34
FF-PR1(1)	4.34	5.09	5.48	5.86	6.06	6.34	6.46	6.68	6.69	6.84	5.98
FF-PR1(5)	4.29	5.05	5.44	5.83	6.03	6.33	6.45	6.68	6.68	6.83	5.96
FF-PR1(10)	4.17	4.98	5.41	5.82	6.02	6.31	6.44	6.67	6.68	6.82	5.93
FF-RN(1, 5)	4.72	5.39	5.89	6.35	6.63	6.79	6.84	7.28	7.22	7.36	6.45
FF-RN(1, 10)	3.70	4.36	4.87	5.32	5.44	5.66	5.78	6.14	6.10	6.27	5.37
FF-RN(5, 5)	3.15	4.30	4.80	5.42	5.76	5.94	6.15	6.56	6.53	6.72	5.54
FF-RN(5, 10)	2.29	3.23	3.78	4.33	4.66	4.83	5.06	5.45	5.43	5.64	4.47
H1(10)	1.45	1.02	0.93	0.80	0.71	0.68	0.60	0.60	0.58	0.54	0.79
H1(20)	1.43	1.08	0.97	0.79	0.71	0.60	0.63	0.59	0.51	0.52	0.78
H2(10, 5)	3.65	4.10	4.29	4.70	4.78	4.82	4.95	5.23	5.20	5.27	4.70
H2(10, 10)	2.81	3.17	3.39	3.69	3.87	3.87	4.01	4.21	4.15	4.29	3.75
H2(20, 5)	3.58	3.96	4.26	4.60	4.65	4.71	4.85	5.02	5.00	5.15	4.58
H2(20, 10)	2.74	3.04	3.33	3.60	3.71	3.74	3.88	4.12	4.10	4.20	3.65
H3(10)	1.11	0.78	0.73	0.64	0.58	0.57	0.49	0.51	0.48	0.44	0.63

Table 31: ACPU for each set of problems arranged by the number of jobs.

Heuristic	Number of jobs										Average
	50	100	150	200	250	300	350	400	450	500	
RZ	0.01	0.06	0.22	0.50	0.94	1.59	2.37	3.49	4.81	6.56	2.06
NEH FT	0.00	0.03	0.08	0.18	0.33	0.58	0.89	1.37	1.89	2.57	0.79
GH-BM2	0.01	0.08	0.26	0.60	1.13	1.91	2.91	4.33	6.04	7.96	2.52
$FRB4_5$	0.04	0.28	0.89	2.05	3.81	6.49	9.94	14.83	20.74	27.98	8.71
$FRB4_{10}$	0.07	0.46	1.49	3.41	6.39	10.86	16.76	25.03	34.68	47.95	14.71
FRB3	0.13	1.63	7.84	23.38	53.71	108.52	192.41	328.51	506.66	761.88	198.47
FF(5)	0.00	0.02	0.05	0.09	0.14	0.20	0.28	0.37	0.48	0.59	0.22
FF(10)	0.01	0.04	0.09	0.17	0.27	0.40	0.56	0.74	0.95	1.17	0.44
FF-NEH(5)	0.02	0.10	0.34	0.75	1.41	2.29	3.44	5.02	7.16	9.49	3.00
FF-NEH(10)	0.04	0.21	0.64	1.44	2.60	4.45	6.67	9.90	13.86	18.02	5.78
FF-ICH1(1)	0.06	0.45	1.75	4.33	8.91	16.17	24.86	43.92	63.15	83.36	24.70
FF-ICH1(5)	0.09	0.65	2.86	7.08	14.30	25.45	38.85	56.56	74.48	99.00	31.93
FF-ICH1(10)	0.15	1.11	4.30	10.30	20.74	35.46	50.94	71.23	98.27	131.80	42.43
FF-PR1(1)	0.05	0.41	1.55	4.01	7.78	14.47	23.93	36.05	49.26	67.66	20.52
FF-PR1(5)	0.11	0.74	2.60	6.40	12.54	22.04	36.03	50.85	73.38	104.53	30.92
FF-PR1(10)	0.14	1.10	3.87	9.14	17.95	31.48	48.84	69.41	98.71	135.01	41.57
FF-RN(1, 5)	0.03	0.24	0.81	1.90	3.64	6.24	9.67	14.44	20.44	27.95	8.54
FF-RN(1, 10)	0.05	0.34	1.13	2.64	5.16	8.50	12.73	19.22	26.86	36.22	11.29
FF-RN(5, 5)	0.17	1.27	4.29	10.13	19.61	33.76	52.82	77.42	109.35	147.68	45.65
FF-RN(5, 10)	0.25	1.75	5.81	13.40	25.66	44.23	68.82	103.47	142.95	189.71	59.61
H1(10)	0.18	1.92	8.74	26.09	59.31	119.37	202.58	354.81	580.70	857.04	221.07
H1(20)	0.14	1.59	7.56	23.29	54.49	109.22	187.62	330.71	523.82	782.41	202.09
H2(10, 5)	0.17	1.80	8.17	24.35	56.65	113.72	200.60	340.18	530.62	800.13	207.64
H2(10, 10)	0.08	0.45	1.48	3.52	6.73	10.88	16.13	23.81	32.97	44.77	14.08
H2(20, 5)	0.09	0.60	2.05	4.95	9.27	15.23	23.25	34.83	47.52	64.37	20.22
H2(20, 10)	0.10	0.62	2.19	5.44	10.07	15.47	22.57	32.64	44.45	58.53	19.21
H3(10)	0.15	0.81	2.62	6.31	12.35	20.02	28.89	42.47	58.66	78.61	25.09

Furthermore, the statistical Tukey test was performed aimed at verifying if there is a statistical difference at 95% confidence level among the means obtained through the best tested methods (FRB3, FRB4$_k$, H1(N), H2(N, k) and H3(N)). The results obtained from the test are shown in Figure 13. It can be observed from this figure that there is no overlapping of the range of error bars at the 95% confidence level, demonstrating, therefore, that the methods compared are statistically different. Given these results, it can be concluded that the proposed heuristics H1(N), H2(N, k) and H3(N) surpass those adapted from the literature.

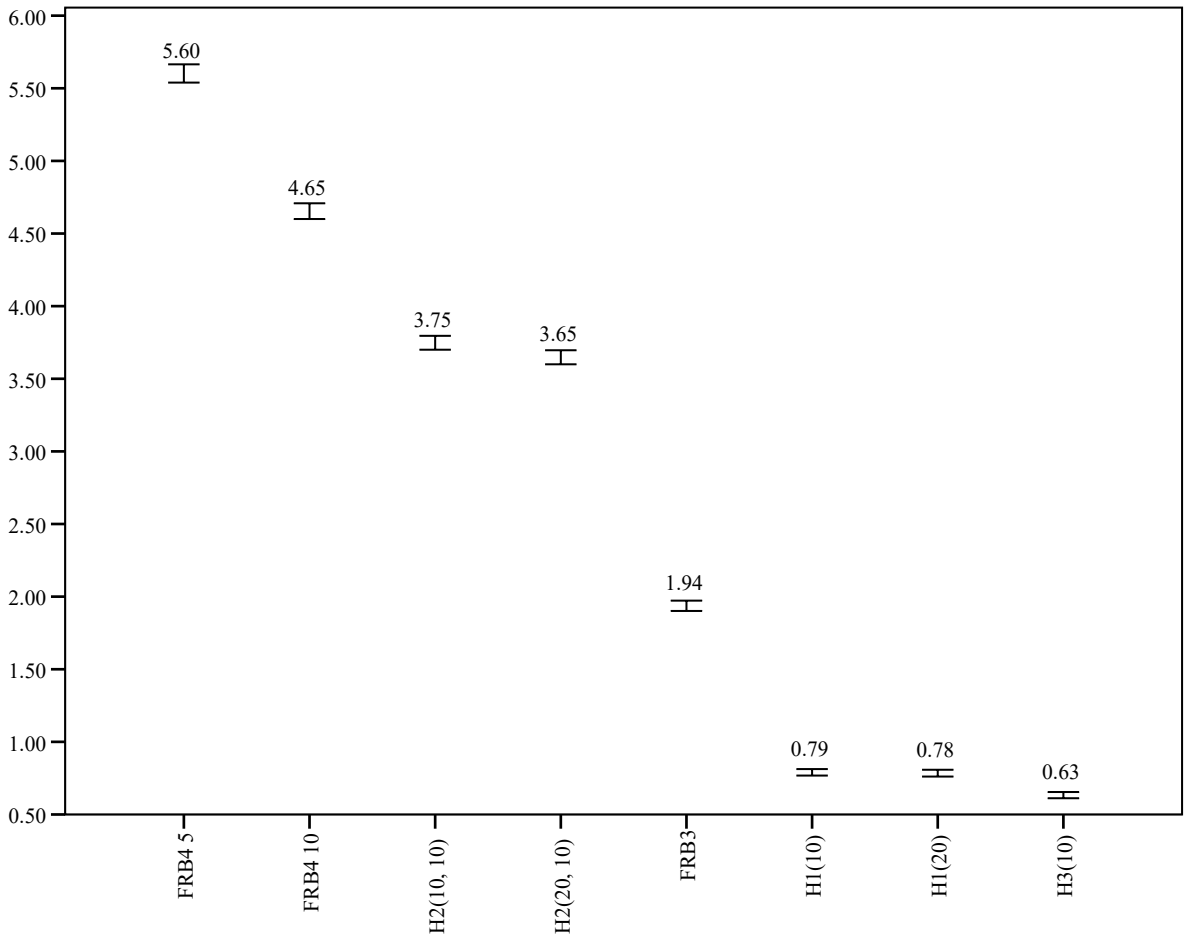

Figure 13: Means plot for the heuristics in all distributions with 95% confidence intervals.

As the proposed heuristics presented the best results in the previous comparison, we choose the H1(N), H2(N, k) and H(N) to be compared against the optimal solutions found by the MILP formulation (Section 2.3.1). We used the set of parameters $N = \{10\}$ and $k = \{5\}$. The Table 32 presents the results in terms of ARPD and Table 33 shows the percentage of optimal solutions found by the proposed heuristics. The ARPD represents the distance of the solutions found by the proposed heuristics relative to the optimal solution

obtained by the MILP for the instance set. The results show that the proposed heuristics find near optimum solutions, with an ARPD of 4.34, 4.99 and 3.75 for the H1(10), H2(20, 5) and H3(10), respectively, when we considers all the instances. In addition, in average 6.67%, 7.11% and 10.67% of the solutions found by the H1(10), H2(20, 5) and H3(10) heuristics are optimal, respectively. The best results are achieved when $n = 10$ and $m = 5$, where the percentage of optimal solutions found can reach up to 30% for the SSD-100 and SS-D125 distributions intervals. Table 34 shows that the MILP CPU time requirements grow considerably as the number of jobs and machines increases. As can be seen the maximum CPU time can reach up to 3 hours and 36 minutes (129738 seconds) in the worst case scenario for the set of instances with $n = 20$, $m = 5$ and SS-D125 distribution. Thus, together with the results presented in the previous benchmarks we can conclude that the our proposed heuristics obtain high quality solutions with computational efficiency.

Table 32: APRD for the proposed heuristics when compared to optimal solutions.

Distribution	n	m	H1(10)	H2(10, 5)	H3(10)
SSD-50	10	5	3.02	3.43	2.77
	10	10	2.48	2.82	2.07
	15	5	4.46	4.35	3.81
	15	10	4.18	4.55	3.23
	20	5	5.04	5.98	4.63
SSD-100	10	5	3.18	3.08	2.71
	10	10	3.14	3.58	2.41
	15	5	4.96	6.50	4.23
	15	10	5.11	4.91	4.48
	20	5	6.19	7.84	5.74
SSD-125	10	5	2.66	3.37	1.90
	10	10	2.54	2.87	2.23
	15	5	6.51	7.56	5.36
	15	10	4.46	6.04	3.77
	20	5	7.18	7.96	6.83
Average			4.34	4.99	3.75

Table 33: Percentage of optimum solutions for the proposed heuristics.

Distribution	n	m	H1(10)	H2(10, 5)	H3(10)
SSD-50	10	5	6.67	6.67	16.67
	10	10	16.67	10.00	20.00
	15	5	6.67	6.67	10.00
	15	10	0.00	3.33	0.00
	20	5	0.00	0.00	3.33
SSD-100	10	5	20.00	30.00	26.67
	10	10	10.00	6.67	16.67
	15	5	3.33	0.00	6.67
	15	10	0.00	0.00	3.33
	20	5	0.00	0.00	0.00
SSD-125	10	5	20.00	23.33	30.00
	10	10	16.67	10.00	20.00
	15	5	0.00	10.00	6.67
	15	10	0.00	0.00	0.00
	20	5	0.00	0.00	0.00
Average			6.67	7.11	10.67

Table 34: Average and maximum CPU time for the MILP model.

Distribution	n	m	Average	Maximum CPU time
SSD-50	10	5	1.06	2.11
	10	10	1.87	3.56
	15	5	14.80	71.62
	15	10	585.47	5981.66
	20	5	478.49	5958.69
SSD-100	10	5	0.98	1.94
	10	10	2.56	13.93
	15	5	10.73	51.08
	15	10	653.41	7191.43
	20	5	148.59	1598.09
SSD-125	10	5	1.19	5.94
	10	10	2.77	11.31
	15	5	32.89	264.33
	15	10	167.48	1369.74
	20	5	2430.59	12973.83
Average			302.19	12973.83

4.4 Conclusion

In this chapter, for the first time, the mixed no-idle PFSP with sequence-dependent setup times and total flowtime minimisation is addressed. Based on a literature review conducted on related problems, high quality heuristic methods were selected in order to adapt them to the problem under study. Moreover, the constructive heuristics H1(N) and H2(N, k) and the improvement heuristic H3(N) were developed. The heuristics were based on the beam search algorithm. At each iteration, nodes are generated by inserting jobs in the last position of partial sequences. The nodes are evaluated through an index function developed according to the generated idleness and to the influence of the job inserted and of the jobs that have not yet been sequenced in the sequence. The best ranked sequences (nodes) are selected to remain for the next iteration. Additionally, the beam search strategy was combined with a constructive procedure that uses variants of the FRB3 and FRB4$_k$ heuristic to construct a final solution; thus, the method can optimise the partial sequences (nodes) generated.

The proposed methods were exhaustively compared through statistical and computational experiments with adapted heuristics in an extensive set of problems with 4500 instances. Heuristics H1(N) and H3(N) obtained the best results, delivering considerably

better solutions than FRB3 (the best method adapted from the literature). Another highlight was the constructive heuristic H2(N, k), with a good trade-off between computational cost and quality solution. The statistical tests that were carried out demonstrated that the solutions generated by the proposed heuristics are statistically better than those obtained by the adapted methods. The proposed heuristics were also compared with the optimal solutions found by the MILP formulation. The results showed that our proposal can generate near optimal solutions for small sized problems instances. Therefore, based on the results presented, it can be asserted that the proposed methods are an important contribution to the state of the art in heuristics for the problem considered in this study. In the next chapter address the mixed no-idle PFSP with total tardiness criterion.

5 THE MIXED NO-IDLE PFSP WITH SETUP TIMES AND TOTAL TARDINESS MINIMISATION

The total tardiness criterion is essential for the current production systems, as surveys of industrial scheduling practice show that meeting customer due dates is a critical concern for many manufacturing systems (RAMAN, 1995). According to Sen e Gupta (1984), when a job is not completed by its due date, certain costs are incurred, i.e. direct dealing with the customer, paperwork, telephone calls, executive time taken up; penalty clauses in the contract; loss of goodwill resulting in an increased probability of losing the customer for some or all future jobs or perhaps in a damaged reputation which will turn other customers away; and expediting (the job is moved quickly through the machines at the possible cost of extra set-ups, double handling of material, inefficient use of workmen and machinery).

In this chapter, the total tardiness minimisation criterion in a mixed no-idle PFSP is addressed given its relevance for the current manufacturing systems. As this is the first time that this problem has been studied, the most efficient heuristics and metaheuristics proposed for the no-idle PFSP problems with total tardiness criterion (denoted by $F_m|prmu, no-idle|C_{max}$, according to Graham et al. (1979)), as well as the PFSP with total tardiness minimisation ($F_m|prmu|\sum T_j$, respectively) were adapted and tested with the purpose of generating a basis of comparison for the proposed heuristics. The methods were compared through computational and statistical experiments in an new benchmark. The results obtained demonstrate that the proposed methods offer high quality solutions with computational efficiency.

The chapter is organised as follows: Section 5.1 analyses the state of the art in heuristics and metaheuristics. Section 5.2 proposes new methods. In Section 5.3, computational and statistical experiments are performed among the compared heuristics. Finally, Section 5.4 draws the main conclusions of the chapter.

5.1 Literature Review

As mentioned earlier, the mixed no-idle PFSP with a sequence-dependent setup times has not yet been studied in the literature. Therefore, we provide a background on heuristics and metaheuristics proposed for other related problems. Basically, the following problems were reviewed: PFSP with total tardiness criterion ($F_m|prmu|\sum T_j$), no-idle PFSP with total tardiness criterion ($F_m|prmu,\ no-idle|\sum T_j$).

5.1.1 The $F_m|prmu|\sum T_j$ problem

The first to study the $F_m|prmu|\sum T_j$ problem was Gelders e Sambandam (1978). They developed four constructive heuristics. Later, the well known NEH heuristic of Nawaz, Enscore e Ham (1983) was adapted by Kim (1993). The NEH heuristic has two phases. In the first phase, jobs are previously ordered in a non-ascending order of the sum of their processing times, also known as Longest Processing Time (LPT). In the second phase, at each iteration, a job ordered in the first phase is evaluated in all possible positions of the programmed jobs partial sequence. The next job from the ordering is considered analogously, and so forth until the n jobs have been sequenced. The NEH_{EDD} differs from NEH by ordering the jobs in the first phase in non-increasing order of due dates (Earliest Due Date dispatch rule – EDD), and by evaluating the sub-sequences during the second phase by their total tardiness instead of their makespan. Kim, Lim e Park (1996) also developed the ENS1 and ENS2 heuristics that start from the NEHedd solution and apply an improvement procedure based on insertion and interchange of jobs, respectively.

Parthasarathy e Rajendran (1998) proposed a simulated annealing (denoted as SAH) algorithm with two perturbation schemes, the Random Insertion Perturbation Scheme and the Curtailed Random Insertion Perturbation Scheme. The proposed SA algorithm with two schemes is evaluated against the heuristic from Kim (1993).

Armentano e Ronconi (1999) developed a tabu search-based algorithm, and compared their proposal with the NEH heuristic and with a Branch-and-Bound algorithm. Four different scenarios of due dates were tested.

Hasija e Rajendran (2004) presented a heuristic algorithm based on the simulated annealing (denoted as HR in this work). The proposed algorithm uses two different perturbation schemes and a new improvement scheme. The authors compared their metaheuristic with the algorithms from Parthasarathy e Rajendran (1998) and Armentano e Ronconi (1999)

Framinan e Leisten (2008) proposed a hybrid algorithm (HA) that uses the Variable Neighbourhood Search (VNS) concept of varying the neighbourhood, but apply it to the destruction and construction phases of the IG algorithm from Ruiz e Stützle (2007). The algorithm is compared with the work from Parthasarathy e Rajendran (1998) and Hasija e Rajendran (2004)

Vallada e Ruiz (2010) developed three genetic algorithms (GAPR, GAPR2, and GADV) that included techniques like path relinking, local search and a procedure to control the diversity of the population. The algorithms outperformed Hasija e Rajendran (2004) and Parthasarathy e Rajendran (1998). The best results were obtained by the GAPR.

An evolutionary algorithm (EA) was proposed by Cura (2015) that outperformed

both GAPR by Vallada e Ruiz (2010) and HR by Hasija e Rajendran (2004). The algorithm included a mating procedure specifically designed for the problem, a local search with two different neighbourhood sizes, and a revision procedure.

Li et al. (2015) proposed six different composite heuristics (denoted as CH_i ($i = \{1, \ldots, 6\}$). Each composite heuristic used NEH_{EDD} as initial solution and apply an improvement procedure based on insertion or/and interchange neighbourhood. Also, Trajectory Scheduling Methods (TSM) are presented and compared against Vallada e Ruiz (2010).

Fernandez-Viagas e Framinan (2015) evaluated NEH_{EDD} and identified a significant number of ties between sub-sequences within the same set, especially in the initial stages of the second phase, when the partial sequence contains a small number of jobs. The authors then developed tie-breaking criteria which substantially improve the heuristics performance. A total of five criteria were proposed: First position (first tie – FT); Last position (last tie – LT); Smallest makespan (MS); Smallest total flowtime (TF); Smallest total earliness (TE).

Karabulut (2016) proposed an the iterated greedy algorithm, denoted as KIG, The proposed iterated greedy algorithm applied a new formula for temperature calculation for acceptance criterion and the algorithm is hybridized with a random search algorithm. The performance of the proposed method is tested against the iterated greedy from Ruiz e Stützle (2007).

Fernandez-Viagas, Valente e Framinan (2018) proposed a beam-search-based constructive heuristic (denoted as BS) that estimates the quality of partial sequences without a complete evaluation of their objective function. In addition, using this constructive heuristic as initial solution, eight variations of an iterated-greedy-based algorithm are proposed. The BS heuristic outperformed the NEHedd by Kim (1993) and the heuristics from Fernandez-Viagas e Framinan (2015) in terms of quality of solutions and computational effort. Regarding the computational evaluation of metaheuristics, the best algorithms were the Iterated Algorithm with Random Adjacent Swap (IA RAS) and the Iterated Algorithm with Greedy Insertion and Insertion Local Search (IA GI ILS). The methods outperformed the algorithms HA by Fernandez-Viagas e Framinan (2015), GAPR by Vallada e Ruiz (2010), EA by Cura (2015) and KIG by Karabulut (2016).

To summarise, many heuristics and metaheuristics algorithms have been proposed in the literature to solve the $F_m|prmu|\sum T_j$ problem. The most promising metaheuristics was proposed by Fernandez-Viagas, Valente e Framinan (2018), that outperformed several other algorithms from previous studies. Under the above considerations, the main methods for the aforementioned problem are listed below:

- Heuristics:

  - NEH_{EDD} by (KIM, 1993).
  - CH_i ($i = \{1, \ldots, 6\}$) by Li et al. (2015).
  - BS heuristic by Fernandez-Viagas, Valente e Framinan (2018).

- Metaheuristics:

  - IA RAS by Fernandez-Viagas, Valente e Framinan (2018).
  - IA GI ILS by Fernandez-Viagas, Valente e Framinan (2018).

5.1.2 The $F_m|prmu,\ no-idle|\sum T_j$ problem

The total tardiness criterion in a no-idle PFSP was studied only recently by Tasgetiren et al. (2011). They proposed a Differential Evolution Algorithm with Variable Parameter Search (vpsDE). The algorithm was compared with a method known as Random Key Genetic Algorithm (RKGA) from Bean (1994). The method proposed by Tasgetiren et al. (2011) outperformed the RKGA, providing statistically better solutions.

The Discrete Artificial Bee Colony (DABC) was proposed by Tasgetiren et al. (2013a). The authors also developed an acceleration method to calculate total tardiness for the insertion neighbour applied to the $F_m|no-idle, prmu|\sum T_j$ problem. The results show that the DABC method was highly competitive when compared to a genetic algorithm.

A constructive heuristic for the $F_m|no-idle, prmu|\sum T_j$ was presented in Nagano, Rossi e Tomazella (2017), denoted as I(f_j,d)-ICH. The heuristic was compared to the FRB3 heuristic from Ruiz, Vallada e Fernández-Martínez (2009) and the NEH_{EDD} by Kim (1993) adapted in this case for the no-idle PFSP. The proposed heuristic is a combination of two procedures: I(f_j, d) and Insertion Constructive Heuristic (ICH). I(f_j, d) generates a partial sequence containing $n \cdot d$ jobs ($0 \leq d \leq 1$), rounded to the nearest integer, by allocating each unscheduled job at the last available position and using a minimization criterion f_j to choose which of the sub-sequences is kept to the next step. Three criteria are suggested for this method: makespan ($f_j = C_{max}$); total flowtime ($f_j = \sum C_j$); and total earliness ($f_j = \sum E_j = \sum max(0, d_j - C_j)$). The ICH procedure consists of the insertion of jobs in a partial sequence and reinsertion movements, which reinserts a pair of jobs at once. The best results were obtained by the I($\sum C_j$, 0.6)-ICH version.

Shao, Pi e Shao (2017) developed a hybrid discrete teaching-learning-based meta-heuristics (HDTLM). The HDTLM applies a probabilistic model based on the selected elite learners where the best learner is employed to generate a series of position sequences, and the concept of consensus permutation is employed to replace the mean individual. In the discrete learning phase, according to different levels of learners, the whole class is first divided into two classes, one is the elite class, and the other is the ordinary class, and then all of learners in these two classes would be assigned into three layers (top layer, middle

layer, bottom layer), and the proposed learning phase adopts the order of top-down to spread the knowledge. The HDTLM outperformed the algorithms vpsDE by Tasgetiren et al. (2011) and DABC by Tasgetiren et al. (2013a)

According to the studies from Shao, Pi e Shao (2017) and Nagano, Rossi e Tomazella (2017) we can identify the most promising heuristics and metaheuristics proposed for the $F_m|prmu,\ no-idle|\sum T_j$, which are the following:

- Heuristics:
  - I($\sum C_j,\ 0.6$)-ICH by Nagano, Rossi e Tomazella (2017).
- Metaheuristics:
  - DABC by Tasgetiren et al. (2013a).
  - HDTLM by Shao, Pi e Shao (2017).

5.2 Proposed Heuristics

In this study, we developed a heuristic based on the beam search algorithm. Beam-search-based heuristics were developed with success for many scheduling problems (FERNANDEZ-VIAGAS; LEISTEN; FRAMINAN, 2016; FERNANDEZ-VIAGAS; VALENTE; FRAMINAN, 2018). In a beam search algorithm, partial sequences (nodes) are generated at each iteration by appending jobs in the last position of the sequence. The best ranked N nodes generated are selected to be used in the next iteration, and so on. The method continues until nodes with complete sequences of n jobs are obtained; then the best ranked node is chosen to be the final solution of the method.

As the I($\sum C_j$)-ICH heuristic from Nagano, Rossi e Tomazella (2017) obtained excellent results, we decided to use their variants in conjunction with the beam-search concept presented by (FERNANDEZ-VIAGAS; VALENTE; FRAMINAN, 2018). Our heuristic construct part sequence using the beam-search concept, inserting d unscheduled jobs, then the rest of the jobs are inserted using a variation of the NEH_{EDD} heuristic (denoted as ICH). We choose to construct part of the sequence using the beam-search-based heuristic in reason of the number of ties generated between partial sequences in the initial iterations of the NEH heuristic (i.e. a problem with 50 jobs and 10 machines in Figure 14), which was already pointed out by Fernandez-Viagas e Framinan (2015). This significant number of ties is mostly due to many partial sequences resulting in total tardiness equal to zero in the initial iterations. Before describing the proposed heuristics in more detail, it is important to define the index function used to evaluate the nodes generated by the heuristic.

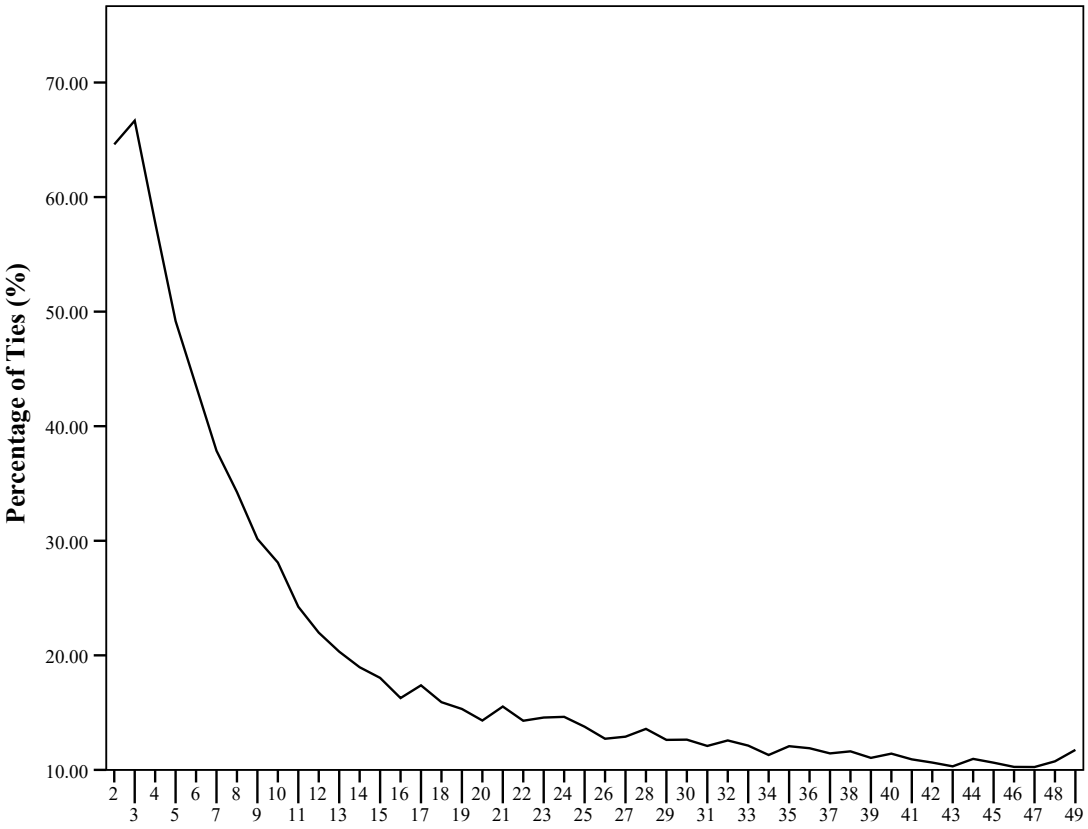

Figure 14: Percentage of ties between partial sequences.

Formally, a node, denoted by η_l^v, is linked to a partial sequence π^v with l jobs, $\pi^v = \{\pi_1^v, \ldots, \pi_l^v\}$ and to a set of jobs that have not yet been sequenced, U_v. A node η_l^v can be branched generating other nodes, η_{l+1}^v, by inserting a job that has not yet been sequenced, $J_j \in U_v$, in the last position $l+1$ de π^v, resulting in $\pi^v = \{\pi_1^v, \ldots, \pi_l^v, \pi_{l+1}^v\}$, where $\pi_{l+1}^v = J_j$. Then, an initial node η_l^v is branched into other nodes, denoted by η_{l+1}^v, each one with a different job $J_j \in U_v$ in the last position of π^v. The principle behind the index is to evaluate three results from the insertion of job J_j in the last position: idle time generated, immediate effect of the job inserted in the last position and influence of the jobs that have not yet been sequenced.

The idle time of the generated node η_{l+1}^v is denoted by IT_{l+1}^v. However, this evaluation can lose relevance in when most machines are no-idle. Thus, supposing a sequence $\pi^v = \{\pi_1^v, \ldots, \pi_l^v, \pi_{l+1}^v\}$ of node η_{l+1}^v, to calculate the completion times of the first l jobs of π^v the method uses the calculation method described in Subsection 2.3.2. However, for the job J_j, inserted at position $l+1$, π_{l+1}^v, the completion times consider only the setup times, and all machines are considered as regular (allow idleness). These completion times of job J_j in machine M_i, denoted by $C_{i,[l+1]}^*$, can be calculated using the expressions below.

When job J_j is inserted in the first position, $l = 1$:

$$\begin{cases} C^*_{1,[1]} = p_{1,[1]} \\ C^*_{i,[1]} = C^{regular}_{i-1,[1]} + p_{i,[1]} \\ i = 2, \ldots, m \end{cases} \tag{5.1}$$

For all remaining positions:

$$\begin{cases} C^*_{1,[l+1]} = C_{1,[l]} + s^1_{[l],[l+1]} + p_{1,[l+1]} \\ C^*_{i,[l+1]} = \max\left(C_{i,[l]} + s^i_{[l],[l+1]}, C^*_{i-1,[l+1]}\right) + p_{i,[l+1]} \\ i = 2, \ldots, m \end{cases} \tag{5.2}$$

Note that the completion times of the job occupying position l ($C_{i,[l]}$) remain unchanged if we change the job which occupies position $l+1$ within the sequence. Therefore, it is possible to pre-calculate $C_{i,[l]}$ and use the same values to calculate the completion times of any job J_j that is inserted in position $l+1$. This enables us to quickly calculate the completion times $C^*_{i,[l+1]}$. After the completion times $C^*_{i,[l+1]}$ are calculated. Finally, the sum of the idle times between the jobs, IT^v_l, are calculated as follows.

$$IT^v_{l+1} = \sum_{i=2}^{m} max\left(C'_{i-1,[l+1]} - C_{i,[l]} - s^i_{[l],[l+1]}, 0\right) \tag{5.3}$$

The immediate effect of inserting job J_j in the last position $l+1$, is evaluated using the completion time of job J_j that occupies the last position, $l+1$, in the last machine M_m, also denoted as $MK_j = C'_{m,[l+1]}$, the earliness $E_j = \max(d_{[k]} - C'_{[k],m})$ and the tardiness $T_j = \max(C'_{[l+1],m} - d_{[l+1]}, 0)$.

Finally, index I^v_l, that evaluates the generated nodes, η^v_{l+1}, is defined below. We used weighted factors in the index to balance the effect of each component of the present expression. The notation $|M'|$ represents the number of no-idle machines. The N selected nodes are those that present the lowest value of I^v_{l+1}.

$$I^v_{l+1} = \frac{|M'| \cdot (n-k-1)}{4 \cdot (m-1)} \cdot IT_j + MK_j + (k + 2.00) \cdot (E_j + T_j) \tag{5.4}$$

At the end of this beam-search-based procedure, there will be a sequence π with d jobs and $n - d$ jobs still to be sequenced that belong to the set denoted by U. The first job of U is inserted in the best position of the π sequence. Afterwards, a procedure, based on the ICH from Nagano, Rossi e Tomazella (2017), is performed. The ICH procedure consists of the insertion of jobs in a partial sequence and reinsertion movements. Differently from FRB3, which reinserts a single job on each step, ICH reinserts a pair of jobs at

once. The purpose of this change is to allow a bigger perturbation on the partial sequence, broadening the search area for a better result. The procedure starts with a partial sequence π containing $n \cdot d$ jobs, and performs $(n - d)$ iterations to insert the remaining jobs in U. The insertion is done similarly to NEH, taking the first job from set U and testing it in all positions of π, choosing as a partial solution the one with smallest total tardiness value. Then, the procedure reinserts the jobs from the current sequence. Instead of reinserting the jobs one by one as the FRB3 heuristic does, the ICH pairs of adjacent jobs are reinserted, π_l and π_{l+1}. Rossi, Nagano e Neto (2016), Nagano, Rossi e Tomazella (2017) already demonstrated that this kind of reinsertion in pairs of jobs results in a better optimisation of partial sequences. During this movement, the jobs from the pair are removed from the sequence and the first job, π_l, is tested in all positions of the sequence, and the best position is selected for insertion; then, the second job, π_{l+1} is considered analogously. Afterwards, the second pair, $\{\pi_{l+2}, \pi_{l+3}\}$, is reinserted and so forth, $\{\pi_{l+4}, \pi_{l+5}\}$, until the last pair, $\{\pi_{n-1}, \pi_n\}$, is done. When the reinsertion is finished, the method inserts the second job from the initial ordering, and then the same reinsertion movements in pairs, as explained above. The next jobs of the initial ordering are considered in the same way. The method continues until the complete sequence with n jobs is obtained. After the sequence is completed with n jobs the heuristic carries out a simple local search based on insertion neighbourhood. This improvement procedure reinserts all jobs in all positions, when a better solution is found the current best solution is updated.

The pseudocode presented in Algorithm 6 shows the beam search procedure used in the proposed heuristics for generating the sequence π with d jobs. The heuristic BS-ICH(N) is described in detail in Algorithms 7.

5.3 Computational and statistical experiments

5.3.1 Instances generation

In this chapter's comparison we generated a mixed no-idle scenario where the machines alternate between no-idle and regular machines. Each problem is generated with combinations between a number of jobs $n = \{50, 100, 150, 200, 250, 300\}$ and a number of machines $m = \{10, 30, 50\}$, totalling $6 \times 3 = 18$ possible combinations. Three replications were generated for each combination, resulting in a total of $18 \times 3 = 54$ problems. We used an uniform distribution [1,99] to generated the processing times.

In this chapter, we address the mixed no-idle flowshop problem with the additional condition of sequence-dependent setup times on regular machines. We added sequence-dependent setup times to regular machines with three different distributions:

- SSD-50: setup times with uniform distribution in the interval [1, 49] (limited to 50% of the limit for the processing time interval).

Algorithm 6 Beam search based procedure.

U is the set of unscheduled jobs, $U = \{J_1, J_2, ..., J_n\}$.
U_j denotes the job occupying the jth position in the set U.
for $h = 1$ to n **do**
  Generate the node η_0^h.
  $\pi^h = \{J_h\}$
  $U^h = U - J_h$
  Evaluate the node η_0^h using the index function φ_0^h.
end for
Order the generated nodes η_0^h in non-desceding order of φ_0^h.
Select the N first ranked nodes η_0^h to be the new set of nodes $\eta_0^v = \{\eta_0^1, \eta_0^2, ..., \eta_0^N\}$.
for $l = 1$ to $d - 1$ **do**
  $h = 1$
  for $v = 1$ to N **do**
    for $j = 1$ to $n - l$ **do**
      Generate the node η_{l+1}^h from η_l^v.
      Insert the job U_j from U^v in the $l + 1$ position of π^v, resulting π^h.
      Evaluate the node η_{l+1}^h using the index function φ_l^h.
      $U^h = U^v - U_j$
      $h = h + 1$
    end for
  end for
  Order the generated nodes η_{l+1}^h in non-desceding order of φ_l^h.
  Select the N first ranked nodes to be the new set of nodes η_{l+1}^v.
end for
Select the sequence π^v of the node η_d^v that results in the lowest $\sum T_j(\pi^v)$.
$\pi = \pi^v$
$U = U^v$
return $\pi = \{\pi_1, ..., \pi_d\}$ and U.

- SSD-100: setup times with uniform distribution in the interval [1, 99] (limited to 100% of the limit for the processing time interval).

- SSD-125: setup times with uniform distribution in the interval [1, 124] (limited to 125% of the limit for the processing time interval).

The due dates were generated using the parameter $\tau = 1, 3$, where the due date of job J_j is $d_j = \tau \cdot \sum_{i=1}^{m} p_{i,j}$. Therefore, the new benchmark for the mixed no-idle PFSP consists of three sequence-dependent setup times distribution intervals (SSD-50, SSD-100, SSD-125) with 54 problems for each distribution. For each problem, two due date scenario were tested, when $\tau = 1$ and $\tau = 3$. Thus, the total number of tests for the benchmark is $54 \times 3 \times 2 = 324$ instances.

Algorithm 7 BS-ICH(N) heuristic

Call the beam search based procedure (Algorithm 2).
for $l = 1$ to $n - d$ **do**
  Insert job U_l in π in the position that results in the lowest $\sum T_j$.
  for $j = 1$ to $l + d - 1$, step $j = j + 2$ **do**
    $\pi' = \pi$
    Remove the jobs π'_j and π'_{j+1} from π'.
    Insert the job π'_j in the position of that results in the lowest $\sum T_j$.
    Insert the job π'_{j+1} in the position of that results in the lowest $\sum T_j$.
    if $\sum T_j(\pi') < \sum T_j(\pi)$ **then**
      $\pi = \pi'$
    end if
  end for
end for
for $j = 1$ to n **do**
  $\pi' = \pi$
  Remove job π_j^{ref} from π'.
  Insert the job π_j^{ref} in the position of π' that results in the lowest $\sum T_j$.
  if $\sum T_j(\pi') < \sum T_j(\pi)$ **then**
    $\pi = \pi'$
  end if
end for
return $\pi = \{\pi_1, ..., \pi_n\}$.

5.3.2 Compared heuristics

Based on the literature review, we identified the heuristics and metaheuristics that could be best adapted to our problem. The methods chosen are listed below. These heuristics and metaheuristics from the literature were compared with those proposed in this work. The adapted heuristics were modified only in the total tardiness evaluation of the sequences with the purpose that the mixed no-idle flowshop with the sequence-dependent setup times is considered in the objective function. To evaluate the total tardiness of a sequence, we used the method presented in Section 2.3.2. We also implemented the acceleration procedure described in Section 2.3.4, which allowed a large increase in calculation speed. The BS heuristic from (FERNANDEZ-VIAGAS; VALENTE; FRAMINAN, 2018) could not be adapted for the addressed problem, as the index used to evaluate the nodes are based on idle times, and explore specific characteristic from the $F|\ prmu|\sum T_j$ problem. As the IA RAS, CH2, CH3, CH4, CH5 and CH6 are partially or manly based on permutation of jobs and could not benefit from the acceleration method (Section 2.3.4) we excluded these methods from the comparisons, as they would be in severe disadvantage compared to the other methods. In the IA GI ILS ((FERNANDEZ-VIAGAS; VALENTE; FRAMINAN, 2018)), we substitute the BS heuristic for our proposed version BS-ICH, this new version was denoted IG GI ILS in this work. We integrated the BS-ICH heuristic in the initialization phase of the metaheuristics DABC and HDTLM. These new versions were denoted as $\text{DABC}_{\text{BS-ICH}}$

and $\text{HTLM}_{\text{BS-ICH}}$, and only one individual is generated using the BS-ICH for the initial population. For the stop rule, the metaheuristics were run with $T_{max} = t \cdot n \cdot m/1000$ seconds with $t = \{250, 500\}$.

- Proposed heuristics.
  - BS-ICH
- Proposed metaheuristics:
  - $\text{DABC}_{\text{BS-ICH}}$, based on the DABC from Tasgetiren et al. (2013a);
  - $\text{HDTLM}_{\text{BS-ICH}}$, based on the HTLM from Shao, Pi e Shao (2017);
  - IG GI $\text{ILS}_{\text{BS-ICH}}$, based on the IA GI ILS from Fernandez-Viagas, Valente e Framinan (2018).
- Heuristics and metaheuristics adapted from the PFSP with total tardiness criterion, $F|\ prmu|\sum T_j$.
  - Heuristics:
    - CH1 by Li et al. (2015);
    - BS heuristic by Fernandez-Viagas, Valente e Framinan (2018).
- Heuristics and metaheuristics adapted from the no-idle PFSP with total tardiness criterion, $F_m|prmu,\ no-idle|\ \sum C_j$.
  - Heuristics:
    - I($\sum C_j$, 0.6)-ICH by Nagano, Rossi e Tomazella (2017).
  - Metaheuristics:
    - DABC by Tasgetiren et al. (2013a);
    - HDTLM by Shao, Pi e Shao (2017).

5.3.3 Performance measures

The performance measures used to compare the heuristics were: quality solution and computational efficiency. The quality solution was evaluated by the relative deviation (DR_h^t) of the heuristic h in problem t, and can be calculated by the expression below.

$$DR_h^t = 100 \cdot \frac{\sum T_j\,(\pi_h)^t - \sum T_j(\pi^*)^t}{\sum T_j(\pi^*)^t} \tag{5.5}$$

where $\sum T_j\,(\pi_h)^t$ is the total tardiness provided by the sequence π_h through the heuristic h for problem t. The best solution for problem t is denoted by $\sum T_j(\pi^*)^t$. It can be seen that as the lower is the value of DR_h^t, the closer the heuristic's solution will be to the

best result found. The mean average of the relative values ($ARPD_h$) of a heuristic h in conjunction with N problems can be calculated by the expression below:

$$ARPD_h = \frac{\sum_{t=1}^{N} DR_h^t}{N} \tag{5.6}$$

To assess the computational efficiency, the average time in seconds was used (average CPU - ACPU). All compared heuristics were implemented in C++, compiled using Intel C++ and executed in an Intel Xeon E5-2680 @ 2.7 GHz with 16 GB RAM memory.

5.3.4 Comparisons between heuristics

The results for the compared heuristics presented in Section 5.3.2 in the set of test problems described in Section 5.3.1 are presented in Table 35, in terms of solution quality (ARPD) and computational efficiency (ACPU). The results for all setup times distributions and due dates scenarios are presented in Table 36. The complete results for different setup times and due dates times scenarios are presented in Tables 38, 39, 40, 41, 42 and 43. The ACPU values are presented in Table 37. The results clearly show that the new proposed heuristic BS-ICH(N) ($N = \{2, 10, 15, n/10\}$) presented the best results, with an ARPD of 2.90 and an ACPU around of 13 seconds for $N = n/10$. After that, the one with the best performance was the heuristics adapted from the literature, I(CT)-ICH and CH1, with worse ARPD of 5.27 and 6.64 and approximately ACPU of 13 and 1.2 seconds, respectively. The CH1 presented a good trade-off between computational efficiency and quality of solution. The NEH_{EDD} with different tie breaking mechanism (LT, TE, TF, FT, MK) obtained the worst results, and no significant different between the versions were found. Figure 15 shows that the proposed heuristics BS-ICH($n/10$) is significant different at 95% confidence level when compared to the CH1 and I(CT)-ICH heuristics. Therefore, the core idea behind BS-ICH of using a beam-search based heuristic in conjunction with the improved NEH extension turned up to be a important contribution for the state-of-the-art in heuristics.

When we compare different distributions, the BS-ICH obtained the best results in the SSD-125, with an ARPD of 2.83 for BS-ICH($n/10$). Considering all setup and due dates scenarios, the best overall results is obtained by BS-ICH($n/10$) when $n = \{250\}$ and $m = \{50\}$ with a resulted ARPD of 0.84. The heuristics usually presents best ARPD when $\tau = 1$. In addition, best solutions are obtained by BS-ICH, I(CT)-ICH and CH1 as the number of machines and jobs grow (Figures 16 and 17). Thus, the heuristics benefit from problems with large instances.

Table 35: ARPD and ACPU values for the compared heuristics in different distributions and due date scenarios.

Heuristics	SSD-50		Average	SSD-100		Average	SSD-125		Average	ARPD	ACPU
	$\tau = 1$	$\tau = 3$	$\tau = \{1, 3\}$	$\tau = 1$	$\tau = 3$	$\tau = \{1, 3\}$	$\tau = 1$	$\tau = 3$	$\tau = \{1, 3\}$		
BS-ICH($n/10$)	2.74	3.30	3.02	2.80	2.92	2.86	2.58	3.07	2.83	2.90	12.83
BS-ICH(10)	2.60	3.31	2.96	2.85	2.82	2.83	2.95	3.01	2.98	2.92	12.81
BS-ICH(2)	3.06	3.36	3.21	2.81	3.01	2.91	2.94	3.28	3.11	3.08	12.82
BS-ICH(15)	2.95	3.42	3.18	2.93	3.01	2.97	2.87	3.33	3.10	3.08	12.81
I(CT)-ICH	5.14	5.76	5.45	5.13	5.47	5.30	4.90	5.24	5.07	5.27	12.76
CH1	6.31	7.16	6.74	6.33	6.85	6.59	6.38	6.84	6.61	6.64	1.19
NEH_{EDD} LT	9.50	10.71	10.11	9.42	10.00	9.71	9.66	9.89	9.77	9.86	0.13
NEH_{EDD} TE	9.51	10.93	10.22	9.37	10.01	9.69	9.69	9.89	9.79	9.90	0.12
NEH_{EDD} TF	9.51	10.60	10.05	9.37	10.12	9.74	9.69	10.17	9.93	9.91	0.08
NEH_{EDD} FT	9.51	10.82	10.17	9.37	10.28	9.83	9.69	9.96	9.82	9.94	0.07
NEH_{EDD} MK	9.50	10.93	10.21	9.42	10.09	9.75	9.66	10.13	9.89	9.95	0.13

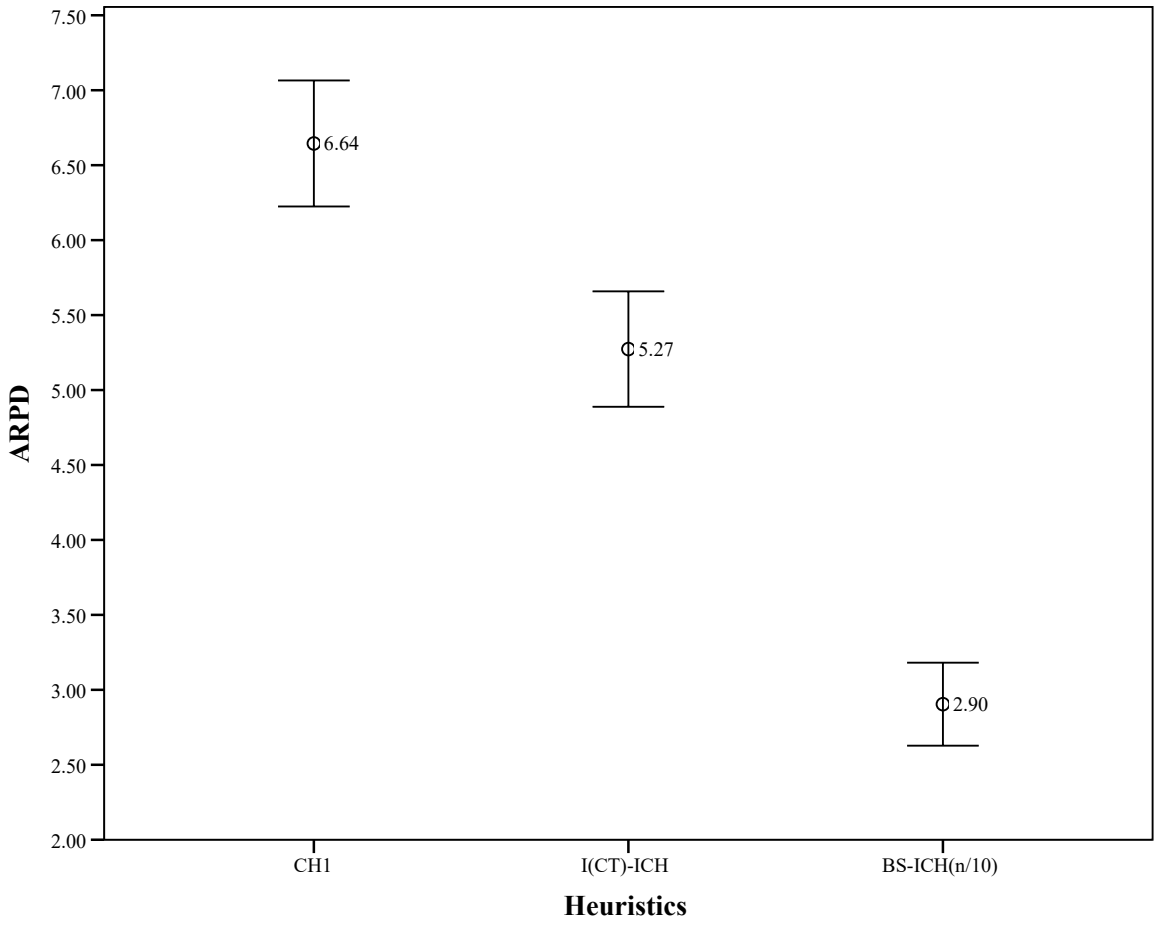

Figure 15: Means plot for the heuristics in all distributions with 95% confidence intervals.

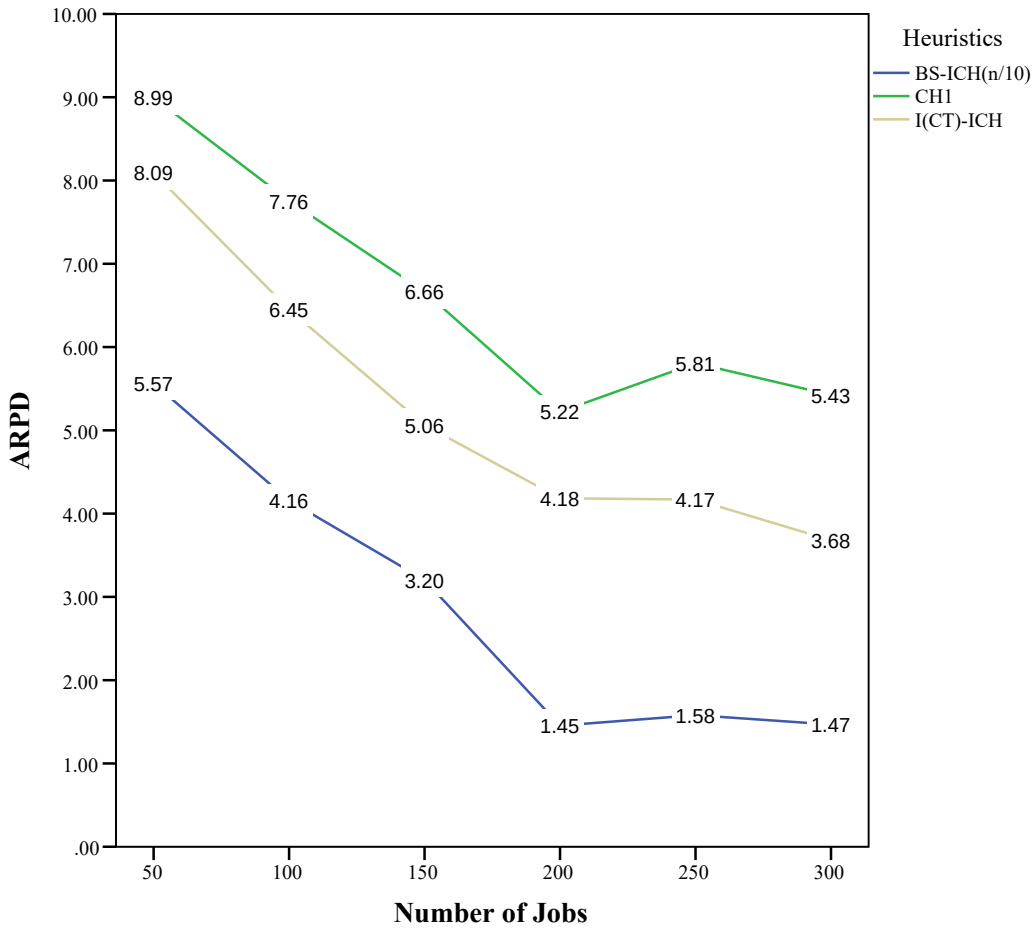

Figure 16: ARPD grouped by number of jobs.

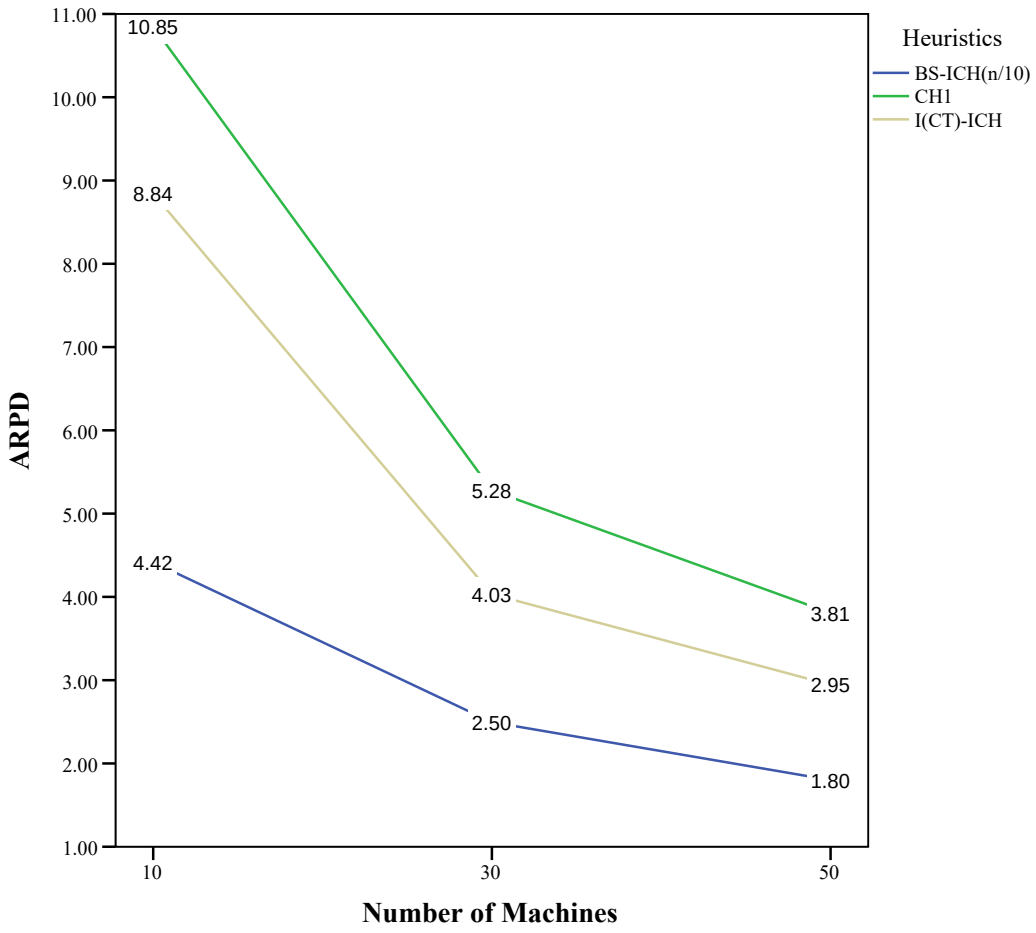

Figure 17: ARPD grouped by number of jobs.

Table 36: ARPD values for the compared heuristics for all setup times distributions and due dates scenarios. The best results are highlighted in bold.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH(n/10)
50	10	22.54	22.02	23.21	22.69	22.32	15.05	13.55	10.46	9.51	9.56	**9.04**
50	30	10.46	10.43	10.53	10.27	10.08	7.01	6.44	4.84	**4.11**	4.42	4.65
50	50	7.25	6.99	7.09	7.07	7.16	4.93	4.29	3.08	3.32	3.29	**3.01**
100	10	19.77	19.82	19.59	19.18	20.42	13.08	11.23	6.78	7.07	**6.50**	7.07
100	30	8.04	7.68	8.05	8.32	7.78	5.61	4.33	3.08	**2.78**	3.26	**2.78**
100	50	6.49	6.58	6.44	6.44	6.57	4.57	3.79	2.65	**2.62**	2.75	**2.62**
150	10	15.87	15.89	15.93	15.93	15.75	11.24	8.84	**4.58**	5.00	5.46	5.46
150	30	7.51	7.32	7.33	7.43	7.43	5.14	3.70	2.70	**2.23**	2.49	2.49
150	50	5.34	5.31	5.29	5.28	5.20	3.61	2.65	1.73	1.72	**1.66**	**1.66**
200	10	12.81	12.54	12.76	12.63	12.71	7.91	6.54	2.11	1.80	2.52	**1.24**
200	30	6.65	6.68	6.72	6.70	6.82	4.29	3.27	1.58	1.55	1.81	**1.50**
200	50	5.13	5.20	5.21	5.21	5.22	3.47	2.74	1.71	**1.53**	1.71	1.63
250	10	15.21	15.29	15.22	15.22	15.18	9.70	7.12	2.86	2.62	3.11	**2.25**
250	30	7.11	7.09	7.17	7.17	6.96	4.88	3.44	1.70	1.75	**1.64**	1.65
250	50	4.34	4.46	4.31	4.40	4.37	2.84	1.95	0.87	**0.76**	0.80	0.84
300	10	12.32	12.33	12.22	12.29	12.19	8.11	5.78	1.86	**1.39**	1.56	1.45
300	30	7.08	6.96	7.01	7.00	7.09	4.74	3.01	**1.76**	1.83	1.81	1.90
300	50	4.98	4.96	5.06	5.11	4.96	3.42	2.25	1.06	**1.04**	1.14	1.05
Average		9.94	9.86	9.95	9.91	9.90	6.64	5.27	3.08	2.92	3.08	**2.90**

Table 37: ACPU values for the compared heuristics for all setup times distributions and due dates scenarios.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH($n/10$)
50	10	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.04	0.04	0.04	0.04
50	30	0.00	0.00	0.00	0.00	0.00	0.02	0.08	0.08	0.09	0.09	0.08
50	50	0.00	0.01	0.01	0.00	0.01	0.03	0.12	0.13	0.14	0.14	0.13
100	10	0.01	0.01	0.01	0.01	0.01	0.09	0.52	0.53	0.54	0.55	0.54
100	30	0.01	0.02	0.02	0.01	0.02	0.15	0.86	0.87	0.88	0.89	0.87
100	50	0.02	0.03	0.03	0.02	0.03	0.22	1.26	1.26	1.27	1.28	1.24
150	10	0.02	0.02	0.03	0.02	0.02	0.22	1.52	1.56	1.57	1.58	1.57
150	30	0.04	0.06	0.06	0.04	0.06	0.50	3.77	3.70	3.72	3.72	3.62
150	50	0.06	0.09	0.10	0.06	0.09	0.81	5.78	5.73	5.71	5.72	5.50
200	10	0.06	0.09	0.09	0.06	0.09	0.81	7.87	8.11	8.02	8.09	8.00
200	30	0.08	0.12	0.13	0.08	0.12	1.25	10.66	10.83	10.67	10.65	10.38
200	50	0.10	0.16	0.17	0.11	0.16	1.66	13.98	14.17	14.06	13.57	13.74
250	10	0.06	0.10	0.10	0.07	0.10	1.02	10.77	10.89	10.93	10.81	10.95
250	30	0.11	0.19	0.19	0.12	0.18	1.65	19.67	20.15	19.53	19.55	19.76
250	50	0.21	0.34	0.36	0.23	0.32	3.41	35.06	35.69	35.56	35.87	35.75
300	10	0.15	0.26	0.24	0.16	0.24	2.70	30.92	31.25	31.23	31.45	31.37
300	30	0.14	0.24	0.23	0.15	0.24	2.31	28.74	28.32	28.75	28.29	28.73
300	50	0.28	0.48	0.48	0.30	0.45	4.46	58.01	57.40	57.80	58.29	58.73
Average		0.07	0.13	0.13	0.08	0.12	1.19	12.76	12.82	12.81	12.81	12.83

Table 38: ARPD values for the compared heuristics in the SSD-50 distribution and $\tau = 1$.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH($n/10$)
50	10	18.34	18.34	18.34	18.34	18.34	12.04	11.98	9.23	7.76	8.60	7.98
50	30	9.99	9.99	9.99	9.99	9.99	6.73	6.68	5.35	4.97	4.39	4.96
50	50	6.58	6.58	6.58	6.58	6.58	4.25	3.65	2.87	2.44	3.11	3.14
100	10	19.49	19.49	19.49	19.49	19.49	13.27	10.82	7.66	5.94	7.60	5.94
100	30	8.63	8.63	8.63	8.63	8.63	6.13	4.57	3.24	2.67	3.29	2.67
100	50	6.25	6.25	6.25	6.25	6.25	4.42	3.90	2.83	2.85	2.78	2.85
150	10	15.35	15.35	15.35	15.35	15.35	11.27	9.29	4.97	4.33	4.78	4.78
150	30	7.94	7.94	7.94	7.94	7.94	5.40	3.07	2.90	2.18	2.48	2.48
150	50	5.26	5.26	5.26	5.26	5.26	3.66	2.92	1.90	1.52	1.49	1.49
200	10	11.85	11.85	11.85	11.85	11.85	7.51	5.32	1.73	0.80	1.87	1.34
200	30	6.78	6.78	6.78	6.78	6.78	4.54	3.56	1.72	1.31	1.86	1.73
200	50	5.67	5.67	5.67	5.67	5.67	3.69	3.16	1.74	1.68	2.24	1.60
250	10	14.71	14.71	14.71	14.71	14.71	8.90	6.85	2.72	2.52	2.68	2.50
250	30	6.92	6.92	6.92	6.92	6.92	4.61	3.75	1.39	1.42	1.58	1.41
250	50	4.14	3.87	3.87	4.14	4.14	2.35	2.14	0.65	0.56	0.69	0.51
300	10	11.53	11.53	11.53	11.53	11.53	6.83	5.56	1.44	1.23	1.06	1.05
300	30	6.52	6.52	6.52	6.52	6.52	4.58	2.82	1.53	1.36	1.42	1.82
300	50	5.30	5.30	5.30	5.30	5.30	3.39	2.55	1.24	1.26	1.13	1.15
Average		9.51	9.50	9.50	9.51	9.51	6.31	5.14	3.06	2.60	2.95	2.74

Table 39: ARPD values for the compared heuristics in the SSD-50 distribution and $\tau = 3$.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH($n/10$)
50	10	29.77	25.75	30.36	25.12	30.47	20.44	16.94	14.07	13.09	12.59	11.70
50	30	13.17	14.12	14.48	13.30	13.39	8.90	8.55	5.64	5.60	6.95	7.05
50	50	9.07	8.72	8.30	8.30	8.66	5.70	4.59	3.99	4.42	3.53	3.76
100	10	21.44	23.13	21.07	21.01	24.39	14.00	11.73	7.12	6.57	6.10	6.57
100	30	8.28	8.30	9.20	9.95	7.72	5.52	5.02	3.91	3.04	3.72	3.04
100	50	6.55	7.63	6.53	6.53	7.25	4.23	3.91	2.47	3.05	2.15	3.05
150	10	17.80	17.35	17.42	17.42	17.21	12.88	9.69	4.56	5.09	7.20	7.20
150	30	8.11	7.28	7.86	8.48	6.75	5.15	3.07	2.82	2.23	2.43	2.43
150	50	5.37	5.66	5.44	5.43	5.30	4.03	2.92	1.84	1.77	1.92	1.92
200	10	13.58	11.86	13.32	12.58	12.79	7.79	5.85	1.77	2.57	1.69	0.72
200	30	6.59	6.36	6.86	6.73	6.68	3.97	3.45	1.52	1.41	1.24	1.53
200	50	5.16	5.28	5.41	5.41	5.78	3.53	2.81	1.49	1.79	1.85	1.55
250	10	14.58	14.81	14.32	14.29	14.52	8.75	7.20	1.92	2.48	3.02	2.47
250	30	6.20	7.18	6.98	6.98	6.58	4.73	3.77	1.42	1.37	1.45	1.11
250	50	4.72	5.09	4.48	4.76	4.88	3.11	2.67	0.89	1.01	0.77	0.98
300	10	11.99	12.15	12.05	11.87	11.56	7.77	5.60	2.09	1.42	1.89	1.68
300	30	7.24	7.06	7.19	7.15	7.40	4.74	3.28	1.80	1.67	1.70	1.78
300	50	5.15	5.15	5.46	5.41	5.34	3.66	2.60	1.20	1.03	1.33	0.94
Average		10.82	10.71	10.93	10.60	10.93	7.16	5.76	3.36	3.31	3.42	3.30

Table 40: ARPD values for the compared heuristics in the SSD-100 distribution and $\tau = 1$.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH($n/10$)
50	10	20.59	20.59	20.59	20.59	20.59	13.42	12.35	9.49	8.65	7.74	7.97
50	30	9.06	9.06	9.06	9.06	9.06	5.57	3.98	3.42	2.99	2.88	3.62
50	50	6.55	6.55	6.55	6.55	6.55	4.89	4.31	2.52	3.42	3.67	2.95
100	10	16.61	17.50	17.50	16.61	16.61	12.06	9.95	6.69	7.46	6.13	7.46
100	30	7.84	7.84	7.84	7.84	7.84	5.65	4.19	2.45	2.63	2.97	2.63
100	50	6.36	6.42	6.36	6.36	6.36	3.99	3.34	2.70	2.32	2.60	2.32
150	10	15.16	15.16	15.16	15.16	15.16	10.56	8.28	3.87	5.68	5.74	5.74
150	30	6.94	6.94	6.94	6.94	6.94	4.61	3.77	2.37	2.05	2.28	2.28
150	50	5.37	5.37	5.37	5.37	5.37	3.59	2.64	1.55	1.43	1.48	1.48
200	10	11.21	11.21	11.21	11.21	11.21	7.20	7.87	1.08	1.26	2.16	0.77
200	30	7.13	7.13	7.13	7.13	7.13	4.83	3.30	1.66	2.31	1.87	1.61
200	50	4.71	4.71	4.71	4.71	4.71	3.61	2.59	1.64	1.21	1.45	1.96
250	10	15.27	15.27	15.27	15.27	15.27	9.14	8.01	3.36	2.49	3.87	2.31
250	30	7.18	7.18	7.18	7.18	7.18	5.17	3.58	1.47	2.11	1.73	1.82
250	50	4.38	4.38	4.38	4.38	4.38	2.77	1.72	0.88	0.78	0.81	1.02
300	10	12.63	12.63	12.63	12.63	12.63	8.63	6.79	2.54	1.21	1.67	1.48
300	30	6.95	6.95	6.95	6.95	6.95	4.86	3.20	1.89	2.13	2.24	2.09
300	50	4.75	4.75	4.75	4.75	4.75	3.35	2.48	1.07	1.12	1.42	0.97
Average		9.37	9.42	9.42	9.37	9.37	6.33	5.13	2.81	2.85	2.93	2.80

Table 41: ARPD values for the compared heuristics in the SSD-100 distribution and $\tau = 3$.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-CH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH($n/10$)
50	10	23.30	23.83	23.90	25.89	20.92	15.16	14.91	9.11	9.25	9.51	10.69
50	30	11.46	10.71	10.55	10.18	9.34	6.75	4.42	4.25	3.21	3.40	4.46
50	50	7.28	7.05	7.32	7.24	7.67	5.23	4.74	3.17	3.30	3.56	2.65
100	10	20.74	18.43	19.14	17.57	21.59	14.84	10.29	5.85	6.95	5.07	6.95
100	30	8.24	6.83	8.27	8.27	7.28	5.83	4.40	3.28	2.39	3.62	2.39
100	50	7.07	6.55	6.67	6.67	6.81	5.17	3.48	2.50	2.98	3.50	2.98
150	10	15.55	16.25	15.72	15.72	16.08	10.54	8.50	5.06	4.85	4.41	4.41
150	30	7.22	7.46	6.95	6.95	7.82	5.28	4.05	3.37	2.07	2.41	2.41
150	50	6.01	5.49	5.79	5.75	5.40	3.72	2.92	1.77	1.35	1.69	1.69
200	10	13.37	12.79	12.63	12.63	13.53	7.80	8.17	2.08	1.02	1.74	1.55
200	30	6.59	7.04	7.16	7.16	7.17	4.00	3.22	0.85	1.54	2.39	0.98
200	50	4.84	4.87	4.88	4.88	4.96	3.33	2.70	1.44	1.77	1.57	1.50
250	10	16.12	16.04	16.15	16.15	16.02	10.84	8.62	3.97	3.30	3.67	2.12
250	30	7.08	6.88	7.06	7.06	6.41	4.91	3.38	1.40	1.78	1.53	1.44
250	50	4.51	4.56	4.44	4.44	4.21	2.76	1.79	0.95	0.92	0.70	0.84
300	10	13.44	12.87	12.82	13.39	12.95	8.88	6.89	2.42	0.68	2.36	1.84
300	30	7.42	7.39	7.33	7.33	7.54	4.99	3.49	1.74	2.35	1.87	2.28
300	50	4.86	4.90	4.82	4.82	4.52	3.27	2.52	1.07	1.04	1.13	1.35
Average		10.28	10.00	10.09	10.12	10.01	6.85	5.47	3.01	2.82	3.01	2.92

Table 42: ARPD values for the compared heuristics in the SSD-125 distribution and $\tau = 1$.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH(n/10)
50	10	21.48	21.48	21.48	21.48	21.48	12.80	11.67	8.73	8.49	8.77	7.24
50	30	9.09	9.09	9.09	9.09	9.09	6.72	7.15	5.12	3.53	4.58	3.62
50	50	6.58	6.58	6.58	6.58	6.58	4.62	4.04	3.05	3.30	2.62	2.75
100	10	20.49	20.49	20.49	20.49	20.49	12.70	12.04	6.40	7.84	6.58	7.84
100	30	7.10	6.59	6.59	7.10	7.10	4.78	3.50	2.50	2.73	2.43	2.73
100	50	6.23	6.23	6.23	6.23	6.23	4.44	3.94	2.30	2.05	2.87	2.05
150	10	14.96	14.96	14.96	14.96	14.96	10.14	7.81	4.43	4.55	3.92	3.92
150	30	7.17	7.17	7.17	7.17	7.17	5.01	4.19	2.49	2.93	2.53	2.53
150	50	4.73	4.73	4.73	4.73	4.73	3.05	2.07	1.40	2.12	1.76	1.76
200	10	13.81	13.81	13.81	13.81	13.81	8.35	6.19	2.74	2.71	4.19	1.22
200	30	6.28	6.28	6.28	6.28	6.28	4.33	3.17	2.29	1.18	1.75	1.52
200	50	5.27	5.27	5.27	5.27	5.27	3.38	2.58	2.42	1.61	1.50	1.80
250	10	15.44	15.44	15.44	15.44	15.44	10.55	6.13	2.54	2.57	2.24	1.30
250	30	7.24	7.24	7.24	7.24	7.24	4.69	2.92	2.27	2.02	1.29	1.92
250	50	4.37	4.37	4.37	4.37	4.37	3.08	1.71	0.62	0.75	0.93	0.96
300	10	12.35	12.35	12.35	12.35	12.35	8.41	5.17	1.15	2.12	1.48	1.27
300	30	6.81	6.81	6.81	6.81	6.81	4.50	2.45	1.55	1.92	1.49	1.32
300	50	4.94	4.94	4.94	4.94	4.94	3.31	1.46	0.86	0.64	0.70	0.74
Average		9.69	9.66	9.66	9.69	9.69	6.38	4.90	2.94	2.95	2.87	2.58

Table 43: ARPD values for the compared heuristics in the SSD-125 distribution and $\tau = 3$.

n	m	NEH_{EDD} FT	NEH_{EDD} LT	NEH_{EDD} MK	NEH_{EDD} TF	NEH_{EDD} TE	CH1	I(CT)-ICH	BS-ICH(2)	BS-ICH(10)	BS-ICH(15)	BS-ICH($n/10$)
50	10	21.78	22.13	24.59	24.73	22.13	16.41	13.43	12.11	9.80	10.16	8.65
50	30	9.99	9.58	9.99	9.99	9.58	7.38	7.85	5.25	4.38	4.29	4.20
50	50	7.44	6.43	7.19	7.19	6.94	4.90	4.41	2.85	3.02	3.28	2.79
100	10	19.87	19.85	19.87	19.87	19.92	11.59	12.54	6.96	7.63	7.52	7.63
100	30	8.17	7.91	7.79	8.11	8.09	5.77	4.31	3.11	3.21	3.53	3.21
100	50	6.46	6.43	6.59	6.59	6.56	5.21	4.20	3.07	2.49	2.59	2.49
150	10	16.42	16.29	16.99	16.99	15.72	12.08	9.45	4.58	5.47	6.68	6.68
150	30	7.67	7.13	7.13	7.13	7.95	5.42	4.05	2.27	1.91	2.82	2.82
150	50	5.31	5.32	5.16	5.16	5.12	3.61	2.42	1.89	2.13	1.63	1.63
200	10	13.04	13.71	13.71	13.71	13.04	8.80	5.85	3.27	2.44	3.46	1.82
200	30	6.51	6.48	6.12	6.12	6.88	4.08	2.94	1.42	1.54	1.73	1.61
200	50	5.11	5.44	5.34	5.34	4.94	3.25	2.59	1.57	1.11	1.68	1.36
250	10	15.13	15.46	15.46	15.46	15.13	10.04	5.92	2.65	2.36	3.15	2.82
250	30	8.04	7.17	7.66	7.66	7.40	5.16	3.24	2.28	1.77	2.26	2.17
250	50	3.94	4.51	4.34	4.34	4.24	2.95	1.64	1.19	0.56	0.93	0.75
300	10	12.01	12.45	11.97	11.97	12.15	8.17	4.67	1.52	1.67	0.92	1.40
300	30	7.53	7.02	7.27	7.27	7.30	4.78	2.83	2.03	1.52	2.14	2.14
300	50	4.85	4.72	5.09	5.45	4.88	3.53	1.90	0.90	1.13	1.11	1.13
Average		9.96	9.89	10.13	10.17	9.89	6.84	5.24	3.28	3.01	3.33	3.07

5.3.5 Comparisons between metaheuristics

The results for the metaheuristics are presented in Table 44. The ARPD for all setup times distributions and due dates scenarios are presented in Table 45. The detailed results are shown in Tables 46, 47, 48, 49, 50 and 51. The best results was achieved by the $\text{DABC}_{\text{BS-ICH}}$ with an ARPD of 0.71 for $t = \{500\}$. The use of BS-ICH in the initialization of the metaheuristics considerably improves the results. As example, the DABC without the BS-ICH results in an ARPD of 1.16, compared to the $\text{DABC}_{\text{BS-ICH}}$ with a much lower ARPD of 0.71, the same thing happens with HTLM. From the results, while in the original problem $(F_m|prmu,\ no-idle|\sum T_j)$, the HTLM obtained best results, the DABC proposal obtains best solutions when the mixed no-idle PFSP with setup times is considered. The IG GI ILS shows good results for large problems ($n \geq 200$) and worse solutions in smaller instances. This behaviour can be seen in all setup time distributions and due date times scenarios. We also test the metaheuristics to verify if the ARPD are statistically different in a 95% confidence interval (Figure 18).

While the metaheuristics outperformed the heuristics by a large margin, it is important to note the difference in efficiency. For example, $\text{DABC}_{\text{BS-ICH}}$ obtains very similar results to BS-ICH($n/10$) when $n \geq 250$ and $m \geq 10$. When $n = 250$ and $m = 50$, $\text{DABC}_{\text{BS-ICH}}$ obtain an ARPD of 0.49 for $t = 500$, that is a $T_{max} = t \cdot n \cdot m/1000 = 500 \cdot 250 \cdot 50/1000 = 6.250$ seconds, while the BS-ICH($n/10$) has an ACPU around 36 seconds and an ARPD of 0.84 for the same problem instance group. Therefore, the results shows that the BS-ICH($n/10$) is highly competitive even when compared to metaheuristics. In addition, the use of BS-ICH with the metaheuristics is a valid proposal as it statistically improves the solution quality of the metaheuristics.

Table 44: ARPD values for the metaheuristics in different distributions and due date times.

Heuristics	SSD-50		Average	SSD-100		Average	SSD-125		Average	ARPD
	$\tau = 1$	$\tau = 3$	$\tau = \{1, 3\}$	$\tau = 1$	$\tau = 3$	$\tau = \{1, 3\}$	$\tau = 1$	$\tau = 3$	$\tau = \{1, 3\}$	
$DABC_{BS\text{-}ICH,\ t = 500}$	0.67	0.65	0.66	0.68	0.80	0.74	0.61	0.86	0.73	0.71
$DABC_{BS\text{-}ICH,\ t = 250}$	1.05	1.06	1.06	1.14	1.25	1.19	1.04	1.25	1.15	1.13
$DABC_{t = 500}$	1.05	0.98	1.01	1.13	1.18	1.16	1.17	1.43	1.30	1.16
$HTLM_{BS\text{-}ICH,\ t = 500}$	1.22	1.33	1.28	1.31	1.51	1.41	1.54	1.60	1.57	1.42
IG GI $ILS_{t = 500}$	1.30	1.87	1.59	1.44	1.45	1.44	1.54	1.55	1.54	1.53
$HTLM_{BS\text{-}ICH,\ t = 250}$	1.36	1.48	1.42	1.49	1.64	1.56	1.69	1.73	1.71	1.56
IG GI $ILS_{t = 250}$	1.51	2.03	1.77	1.65	1.66	1.65	1.70	1.76	1.73	1.72
$HTLM_{t = 500}$	1.37	1.45	1.41	1.85	1.88	1.86	1.91	2.03	1.97	1.75
$DABC_{t = 250}$	1.68	1.52	1.60	1.95	1.84	1.89	1.81	2.05	1.93	1.81
$HTLM_{t = 250}$	1.66	1.69	1.68	2.12	2.29	2.20	2.22	2.36	2.29	2.06

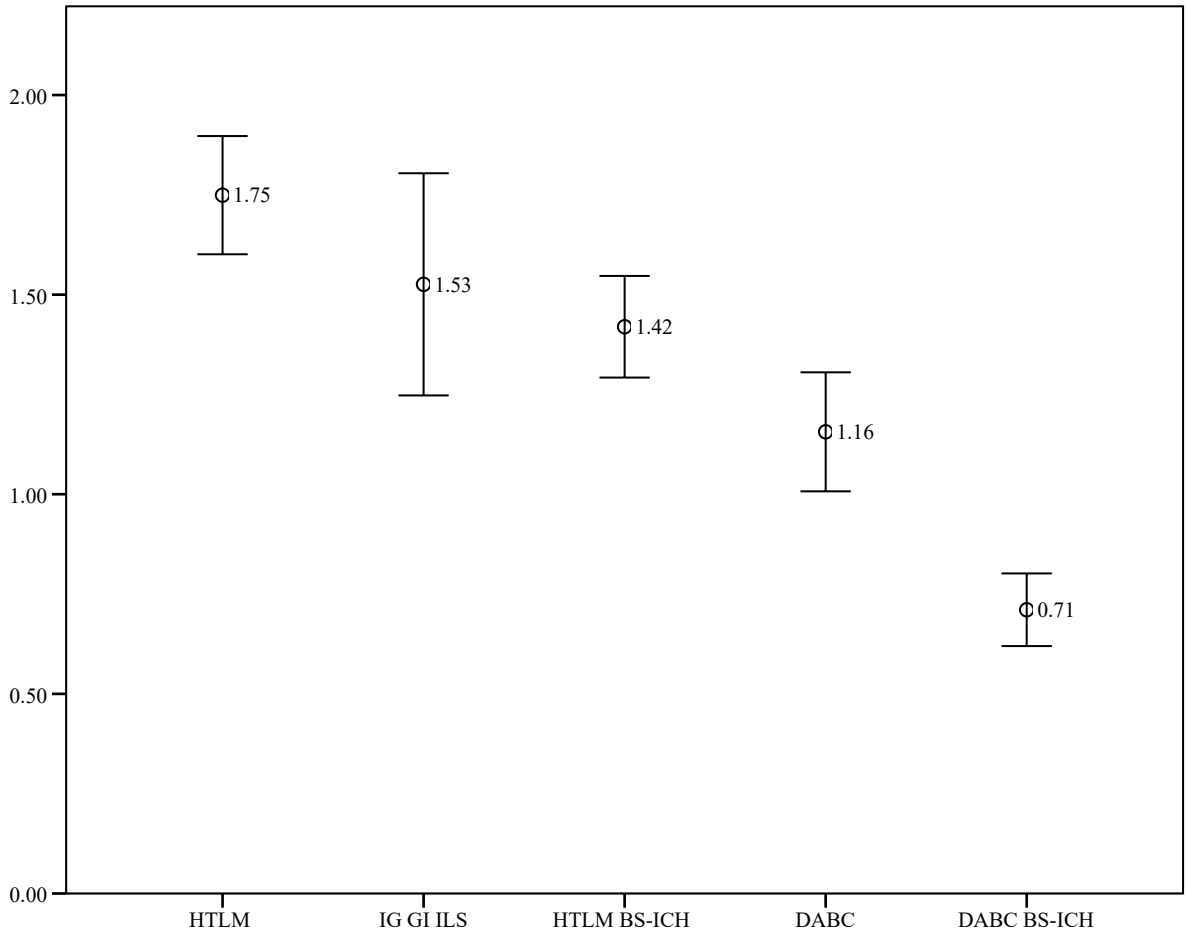

Figure 18: Means plot for the metaheuristics in all distributions with 95% confidence intervals and $T_{max} = 500 \cdot n \cdot m$.

Table 45: ARPD values for the metaheuristics for all setup times distributions and due dates scenarios. The best results are highlighted in bold.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	0.71	0.89	2.41	2.43	6.93	**0.26**	0.41	2.08	1.92	6.92
50	30	0.34	0.41	1.10	1.04	2.75	0.14	**0.08**	0.88	0.81	2.72
50	50	0.20	0.22	0.65	0.71	1.51	0.10	**0.08**	0.48	0.63	1.47
100	10	1.93	2.02	2.71	3.13	7.07	**0.36**	0.55	2.40	2.77	7.07
100	30	0.67	0.61	1.62	1.39	1.94	0.17	**0.16**	1.46	1.24	1.92
100	50	0.72	0.63	1.43	1.35	0.89	**0.31**	0.37	1.29	1.26	0.78
150	10	2.81	2.72	3.73	3.22	1.37	**1.38**	1.45	3.33	2.88	0.99
150	30	1.07	1.15	1.47	1.44	1.27	**0.38**	0.39	1.25	1.21	1.15
150	50	0.80	0.68	1.02	0.97	1.00	0.36	**0.30**	0.89	0.89	0.90
200	10	2.98	0.90	2.24	1.34	1.80	1.96	**0.45**	1.50	1.27	1.80
200	30	1.35	1.14	1.30	1.19	0.80	0.81	**0.70**	1.16	1.15	**0.70**
200	50	1.31	1.17	1.46	1.27	0.36	0.89	0.88	1.28	1.20	**0.13**
250	10	5.07	1.88	4.64	2.46	0.78	3.80	1.63	3.90	2.34	**0.07**
250	30	2.10	1.48	2.01	1.54	0.54	1.45	1.30	1.86	1.44	**0.19**
250	50	1.27	0.67	1.06	0.64	0.29	0.79	0.49	0.89	0.61	**0.19**
300	10	4.91	1.30	4.24	1.27	0.72	4.37	1.25	3.52	1.26	**0.43**
300	30	2.55	1.56	2.38	1.76	0.58	1.99	1.43	1.99	1.72	**0.00**
300	50	1.77	0.93	1.58	0.97	0.31	1.29	0.85	1.33	0.97	**0.01**
Average		1.81	1.13	2.06	1.56	1.72	1.16	**0.71**	1.75	1.42	1.53

Table 46: ARPD values for the compared metaheuristics in the SSD-50 distribution and $\tau = 1$.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	0.76	0.49	1.23	1.57	5.56	0.11	0.26	1.21	1.47	5.56
50	30	0.26	0.55	1.12	1.19	3.88	0.08	0.12	0.72	0.77	3.78
50	50	0.17	0.32	0.60	0.72	1.05	0.11	0.07	0.59	0.58	1.02
100	10	2.14	1.89	2.00	2.52	5.94	0.20	0.48	1.33	2.18	5.94
100	30	0.54	0.27	1.43	1.49	1.99	0.05	0.12	1.43	1.13	1.99
100	50	0.86	0.88	1.51	1.56	1.04	0.56	0.50	1.44	1.44	0.96
150	10	2.90	2.14	2.79	2.60	1.44	1.41	1.23	2.75	2.30	0.74
150	30	1.16	1.06	1.23	1.07	1.05	0.51	0.11	1.00	0.96	1.03
150	50	0.92	0.72	1.08	0.80	0.70	0.40	0.39	0.99	0.71	0.66
200	10	2.19	1.13	1.22	0.76	0.80	1.39	0.97	0.38	0.71	0.80
200	30	1.41	1.43	1.67	1.10	0.43	0.85	1.04	1.50	1.10	0.34
200	50	1.41	1.26	1.52	1.34	0.39	0.97	1.08	1.30	1.30	0.12
250	10	4.66	1.72	2.97	2.34	1.05	3.30	1.32	2.87	2.34	0.01
250	30	1.98	1.35	1.96	1.20	0.38	1.30	1.08	1.75	0.94	0.00
250	50	1.06	0.43	0.63	0.44	0.21	0.56	0.34	0.40	0.44	0.20
300	10	4.23	0.77	3.61	1.18	0.66	4.14	0.73	2.01	1.17	0.32
300	30	1.86	1.63	1.90	1.36	0.29	1.50	1.47	1.71	1.36	0.00
300	50	1.77	0.95	1.44	1.14	0.26	1.38	0.83	1.31	1.14	0.00
Average		1.68	1.05	1.66	1.36	1.51	1.05	0.67	1.37	1.22	1.30

Table 47: ARPD values for the compared metaheuristics in the SSD-50 distribution and $\tau = 3$.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	0.18	0.91	2.56	2.96	10.52	0.18	0.46	2.43	1.79	10.52
50	30	0.28	0.62	1.35	0.97	2.99	0.24	0.05	1.26	0.76	2.99
50	50	0.28	0.19	0.64	0.61	2.56	0.16	0.12	0.22	0.48	2.51
100	10	1.62	1.69	1.65	2.51	6.57	0.29	0.20	1.65	2.42	6.57
100	30	0.44	0.67	1.52	1.30	2.51	0.05	0.16	1.36	1.19	2.43
100	50	0.45	0.44	1.54	1.20	1.11	0.00	0.31	1.37	1.18	1.11
150	10	2.12	2.25	2.70	3.04	1.48	0.95	1.49	2.19	2.80	1.30
150	30	0.71	0.92	1.20	1.21	1.35	0.20	0.13	0.84	1.03	1.29
150	50	0.77	0.63	0.76	0.86	0.95	0.19	0.13	0.76	0.80	0.92
200	10	2.50	0.52	1.08	1.66	2.57	1.91	0.12	0.53	1.48	2.57
200	30	1.10	1.22	1.36	0.97	0.56	0.78	0.65	1.03	0.93	0.46
200	50	1.02	0.97	1.05	1.18	0.46	0.57	0.64	0.98	1.12	0.33
250	10	4.77	2.31	4.28	2.22	0.61	3.36	2.14	3.60	2.22	0.03
250	30	1.61	0.88	1.58	1.34	0.51	1.15	0.76	1.50	1.34	0.23
250	50	1.46	0.78	1.11	0.73	0.31	0.94	0.58	1.01	0.60	0.23
300	10	4.04	1.53	2.39	1.20	0.61	3.34	1.53	2.39	1.17	0.20
300	30	2.12	1.60	2.01	1.67	0.46	1.89	1.47	1.65	1.67	0.00
300	50	1.87	0.89	1.66	0.96	0.32	1.36	0.76	1.34	0.96	0.03
Average		1.52	1.06	1.69	1.48	2.03	0.98	0.65	1.45	1.33	1.87

Table 48: ARPD values for the compared metaheuristics in the SSD-100 distribution and $\tau = 1$.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	0.66	0.61	2.12	1.18	6.10	0.17	0.03	1.30	0.83	6.10
50	30	0.56	0.28	0.98	0.88	2.47	0.20	0.05	0.74	0.71	2.47
50	50	0.20	0.21	0.56	0.74	1.32	0.06	0.00	0.46	0.59	1.32
100	10	2.26	1.93	3.26	3.13	7.46	0.39	0.38	2.83	2.32	7.46
100	30	0.71	0.74	1.77	1.33	1.83	0.06	0.13	1.50	0.98	1.82
100	50	0.87	0.65	1.25	1.38	1.03	0.10	0.52	1.19	1.36	0.79
150	10	3.05	3.59	3.67	3.25	1.12	1.31	1.40	3.59	2.86	0.88
150	30	0.98	1.19	1.37	1.48	1.24	0.43	0.45	1.35	1.34	0.89
150	50	0.70	0.56	0.77	0.91	0.88	0.23	0.27	0.65	0.64	0.82
200	10	3.06	0.68	1.06	1.01	1.26	1.25	0.68	0.86	1.01	1.26
200	30	1.51	1.05	1.16	1.54	1.36	0.98	0.72	1.16	1.50	1.25
200	50	1.37	1.27	1.68	1.10	0.11	0.90	0.91	1.46	1.09	0.00
250	10	5.01	1.57	4.85	2.27	1.01	3.36	1.21	4.22	2.10	0.00
250	30	2.35	1.60	2.20	1.94	0.87	1.52	1.35	1.95	1.71	0.54
250	50	1.23	0.74	1.17	0.73	0.30	0.75	0.35	1.10	0.70	0.14
300	10	6.05	1.46	6.16	1.08	0.59	5.42	1.46	5.51	1.08	0.20
300	30	2.69	1.42	2.47	1.90	0.54	1.97	1.33	1.90	1.84	0.00
300	50	1.87	0.92	1.67	1.00	0.23	1.34	0.92	1.44	1.00	0.00
Average		1.95	1.14	2.12	1.49	1.65	1.13	0.68	1.85	1.31	1.44

Table 49: ARPD values for the compared metaheuristics in the SSD-100 distribution and $\tau = 3$.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	0.43	1.05	2.39	2.49	6.51	0.36	0.62	2.12	2.38	6.51
50	30	0.16	0.38	0.92	1.14	1.80	0.10	0.02	0.88	1.00	1.75
50	50	0.15	0.09	0.70	0.72	1.39	0.13	0.04	0.50	0.69	1.39
100	10	1.37	2.09	3.88	3.15	6.95	0.00	0.78	3.15	3.01	6.95
100	30	0.80	0.52	1.41	0.99	1.67	0.02	0.16	1.34	0.94	1.66
100	50	0.82	0.74	1.55	1.55	0.71	0.58	0.45	1.49	1.26	0.71
150	10	2.40	3.06	4.28	3.49	2.38	1.23	1.72	3.22	3.05	1.83
150	30	1.06	1.53	1.63	1.85	1.29	0.38	0.74	1.57	1.43	1.23
150	50	0.80	0.85	1.16	1.02	0.90	0.52	0.29	0.98	1.02	0.70
200	10	2.68	1.33	3.07	0.83	1.02	1.57	0.32	1.82	0.83	1.02
200	30	1.36	0.91	1.05	1.29	0.95	0.51	0.56	0.88	1.25	0.94
200	50	1.43	1.18	1.55	1.56	0.57	1.01	0.92	1.15	1.43	0.27
250	10	5.60	1.70	5.16	3.12	0.82	4.27	1.39	4.18	2.75	0.25
250	30	2.00	1.30	1.68	1.56	0.67	1.38	1.22	1.40	1.56	0.37
250	50	1.35	0.84	1.08	0.66	0.51	0.66	0.66	0.90	0.64	0.39
300	10	5.92	1.80	5.25	0.68	0.27	4.95	1.63	4.48	0.68	0.10
300	30	2.97	1.81	2.84	2.35	0.95	2.41	1.68	2.50	2.28	0.00
300	50	1.76	1.24	1.58	1.00	0.47	1.20	1.11	1.34	1.00	0.01
Average		1.84	1.25	2.29	1.64	1.66	1.18	0.80	1.88	1.51	1.45

Table 50: ARPD values for the compared metaheuristics in the SSD-125 distribution and $\tau = 1$.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	1.36	1.00	2.54	3.15	6.14	0.37	0.21	2.31	2.05	6.13
50	30	0.47	0.33	1.02	0.92	2.08	0.20	0.07	0.87	0.75	2.08
50	50	0.23	0.29	0.63	0.76	1.15	0.00	0.24	0.40	0.73	1.09
100	10	1.86	2.45	2.35	3.77	7.84	0.12	1.16	2.35	3.61	7.84
100	30	0.28	0.50	1.37	1.14	1.70	0.06	0.15	1.27	1.14	1.70
100	50	0.55	0.63	1.40	1.18	0.60	0.22	0.21	0.98	1.13	0.50
150	10	2.24	2.27	4.17	2.44	1.26	0.79	0.96	3.49	2.12	0.86
150	30	1.34	1.15	1.89	1.75	1.50	0.77	0.53	1.84	1.56	1.50
150	50	0.80	0.50	1.00	1.03	1.14	0.35	0.28	0.90	0.92	1.05
200	10	2.96	0.79	3.73	2.52	2.71	2.41	0.00	3.11	2.52	2.71
200	30	1.71	0.95	1.35	1.16	0.79	1.04	0.57	1.35	1.16	0.57
200	50	1.38	1.38	1.48	1.35	0.35	0.86	0.95	1.36	1.19	0.09
250	10	5.13	1.21	5.71	2.42	0.57	4.34	1.15	4.89	2.35	0.13
250	30	2.24	1.65	2.20	1.65	0.50	1.39	1.36	2.14	1.58	0.00
250	50	1.30	0.66	1.16	0.74	0.31	1.01	0.54	0.98	0.74	0.20
300	10	4.82	1.22	4.11	1.96	1.32	4.28	1.22	3.18	1.96	1.23
300	30	2.49	1.06	2.46	1.79	0.50	1.88	0.85	2.03	1.64	0.00
300	50	1.45	0.60	1.36	0.62	0.19	1.05	0.53	1.02	0.62	0.00
Average		1.81	1.04	2.22	1.69	1.70	1.17	0.61	1.91	1.54	1.54

Table 51: ARPD values for the compared metaheuristics in the SSD-125 distribution and $\tau = 3$.

n	m	$t = 250$					$t = 500$				
		DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS	DABC	$\text{DABC}_{\text{BS-ICH}}$	HTLM	$\text{HTLM}_{\text{BS-ICH}}$	IG GI ILS
50	10	0.87	1.29	3.60	3.26	6.72	0.41	0.86	3.10	2.99	6.72
50	30	0.33	0.31	1.19	1.16	3.27	0.02	0.20	0.83	0.87	3.27
50	50	0.14	0.21	0.78	0.72	1.57	0.12	0.03	0.75	0.71	1.50
100	10	2.33	2.06	3.11	3.72	7.63	1.15	0.30	3.11	3.08	7.63
100	30	1.24	0.96	2.25	2.11	1.94	0.79	0.26	1.84	2.03	1.94
100	50	0.74	0.45	1.35	1.22	0.83	0.39	0.25	1.24	1.19	0.64
150	10	4.14	3.00	4.74	4.47	0.51	2.60	1.89	4.73	4.18	0.33
150	30	1.17	1.06	1.48	1.30	1.20	0.00	0.41	0.91	0.92	0.98
150	50	0.82	0.78	1.32	1.23	1.46	0.49	0.45	1.06	1.22	1.25
200	10	4.52	0.94	3.30	1.25	2.44	3.25	0.58	2.31	1.10	2.44
200	30	1.02	1.27	1.18	1.08	0.73	0.73	0.69	1.06	0.97	0.66
200	50	1.22	0.98	1.48	1.09	0.26	1.01	0.79	1.41	1.07	0.00
250	10	5.26	2.76	4.85	2.36	0.63	4.15	2.56	3.64	2.28	0.00
250	30	2.46	2.13	2.46	1.51	0.34	1.96	2.04	2.43	1.49	0.00
250	50	1.20	0.55	1.20	0.51	0.11	0.80	0.48	0.95	0.51	0.00
300	10	4.42	1.00	3.94	1.52	0.87	4.12	0.93	3.54	1.52	0.52
300	30	3.18	1.83	2.58	1.50	0.74	2.29	1.78	2.14	1.50	0.00
300	50	1.92	1.00	1.75	1.08	0.37	1.39	0.93	1.51	1.08	0.04
Average		2.05	1.25	2.36	1.73	1.76	1.43	0.86	2.03	1.60	1.55

5.4 Conclusion

In this chapter, the mixed no-idle PFSP with sequence-dependent setup times and total flowtime minimisation is studied. Based on a literature review conducted on related problems, the best heuristic and metaheuristics were adapted to the problem. In addition, a new heuristic, denoted as BS-ICH, was proposed. We integrated the BS-ICH with the metaheuristics from the literature. The proposed methods were exhaustively compared through statistical and computational experiments with adapted heuristics and metaheuristics. Among the heuristics, the best results were achieved by the BS-ICH($n/10$), delivering considerably better solutions than I(CT)-ICH, which was the best method from the $F_m|prmu, no-idle|\sum T_j$ problem. According to the results, the used of the BS-ICH in conjunction with the metaheuristics considerably improves the solution quality of the metaheuristics. As a result, the best metaheuristic was the $\text{DABC}_{\text{BS-ICH}}$, which is the DABC algorithm with the BS-ICH used in the initial population generation. Thus, based on the results presented, it can be asserted that the proposed method is a significant contribution for the state of the art in heuristics and metaheuristics for the problem considered in this chapter.

6 CONCLUSIONS, RESULTS AND FUTURE RESEARCH

6.1 Conclusions

In this Thesis, we have addressed the mixed no-idle permutation flowshop scheduling problem. This is the first time that this problem is studied in the literature. It has been showed that the problem has relevance in real manufacturing layouts, and deals with establishing the sequence of jobs in the shop according to a specific objective function under specific characteristics of the problem, namely the setup times between the jobs and the mixed no-idle machines.

The goal of this Thesis was therefore to provide a further insight into this important problem, both deeply analysing and developing new efficient methods to solve it. In order to deal with this goal, several general research objectives were identified in Section 1.1, which have been addressed along the four parts of this Thesis as follows:

OBJ1. To provide in-depth analysis of the mixed no-idle PFSP with sequence-dependent setup times, presenting mathematical models, formulations and calculations methods for the addressed objective functions.

In Section 2.3 the mixed no-idle PFSP with sequence-dependent setup times was formally presented. We developed an MILP formulation for the problem in Section 2.3.1, the variables and expressions were explained in detail. As the problem was not yet studied in the literature, we presented methods for calculation for the makespan, total flowtime and total tardiness criteria (Section 2.3.2). As explained, the acceleration procedure is essential to provide efficiency for the new proposed methods. In Section 2.3.3 and 2.3.4 we provide acceleration methods to calculate the makespan, total flowtime and total tardiness in an insertion neighbourhood.

OBJ2.To review the no-idle PFSP, PFSP with setup times and classical PFSP literature for the most common objectives, i.e. makespan, total flowtime or total tardiness minimisation.

The mixed no-idle PFSP with setup times and makespan minimization was considered in Chapter 3. As the problem is new, we reviewed the no-idle PFSP and the PFSP with setup times under makespan criterion (denoted as $F_m|prmu,\ no-idle|C_{max}$ and $F_m|prmu,\ s^i_{j,k}|C_{max}$, respectively). The literature review covered the last 35 years, where many heuristics and metaheuristics were developed for the problem. We classified the methods and presented if the algorithm was outperformed by another method from the literature (Tables 2 and 3).

Chapter 4 studied the mixed no-idle PFSP with setup times and total flowtime. For this problem, the following topics were reviewed: heuristics for the PFSP with makespan criterion ($F_m|prmu|C_{max}$), PFSP with total flow-time criterion ($F_m|prmu|\sum C_j$), no-idle and mixed no-idle PFSP with makespan criterion ($F_m|prmu,\ no-idle|C_{max}$, $F_m|prmu,\ mixed\ no-idle|C_{max}$). In the review, over 50 papers were reviewed. Most of the heuristics proposed in the literature are variants of the traditional NEH.

In Chapter 5 the total tardiness criterion was addressed. The following problems were reviewed: PFSP with total tardiness criterion ($F_m|prmu|\sum T_j$), no-idle PFSP with total tardiness criterion ($F_m|prmu,\ no-idle|\sum T_j$). Both heuristics and metaheuristics algorithms were reviewed.

OBJ3. To provide efficient methods to solve the mixed no-idle PFSP with sequence dependent setup times for makespan, total flowtime or total tardiness minimisation.

Based on the state-of-the-wart in heuristics, we developed several methods in order to solve the mixed no-idle PFSP with setup times under different criteria. All the new methods were tailor-made for their specific problems. Delving into particular characteristics of each criteria with the objective to give an edge for the new heuristics. The main proposed methods are summarised as follows:

- A efficient constructive heuristic, denoted as RN, was proposed in Chapter 4 (makespan criterion). The heuristic inserts d jobs using a greedy heuristic, and then the rest of the $n-d$ are inserted into the sequence using an NEH heuristic variant. The heuristic takes account the idle times between the jobs and the setup times to generate a index that chooses the jobs to be inserted. The NEH variant is based on reinsertion procedures that optimized the partials sequences generated by the NEH.
- Three heuristics based on beam search, called H1, H2 and H3, were proposed in Chapter 5 (total flowtime criterion). In developed beam search algorithms, partial sequences are generated at each iteration by inserting jobs in the last position of the sequence. The best ranked N nodes generated are selected to be used in the next iteration. The method continues until nodes with complete sequences of $n-d$ jobs are obtained; then the best ranked node is chosen to be the final solution of the method. The rest of the sequence are constructed using a variants of the FRB3 and FRB4 heuristics. The H3 is an improvement heuristic, which carries out a local search in the final solution generated by the H1 method.
- In Chapter 5, a new heuristic, denoted as BS-ICH, was proposed. The concept is similar to the heuristics from Chapters 3 and 4, with the exception of the new

index which takes into account the specific characteristics of the total tardiness criterion. The new method was also integrated in the initial phases of metaheuristics procedures from the literature.

OBJ4. To demonstrate the efficiency and good performance of the new proposed methods thought extensive computational and statistical experiments.

In this Thesis, each proposed heuristic was always compared with the state-of-the-art algorithms adapted from related scheduling problems. As the mixed no-idle PFSP with setup times is new, we developed a new benchmark with up to 4500 problems instances to test the algorithms. The indicators to measure the computational effort and the solution quality were also explained. To carry out a fair, comparison we presented in detail the compared methods, as well the same programming language C++ and the same computer were used to compare all methods. To ensure the repeatability and reproducibility of our proposed heuristic, we included a clear pseudo code (Algorithm 2) in this Thesis. The results and problem instances are available at laor.prod.eesc.usp.br, as recommended by the Good Laboratory Practice for Optimization Research (GLP4OPT) practices (KENDALL et al., 2016). The following computational results were presented in each chapter:

- For the makespan minimization (Chapter 3), the analyses showed that the proposed RN_x method surpassed the state-of-the-art algorithms both in solution quality and computational efficiency. Algorithms adapted from the $F|prmu, no-idle|C_{max}$ and $F|prmu, s^i_{j,k}|C_{max}$ problems. The statistical results also show that the means are significantly different. In addition, the proposed methods generated near optimal solutions for small size problems.

- For the total flowtime criterion (Chapter 4), heuristics H1(N) and H3(N) obtained the best results, delivering better solutions than the best methods adapted from the literature. The constructive heuristic H2(N, k) presented a good trade-off between computational cost and quality solution. Thought statistical tests, were demonstrated that the solutions generated by the proposed heuristics are statistically better than those obtained by the adapted methods. The proposed heuristics were also compared with the optimal solutions found by the MILP formulation. The results showed that our proposal can generate near optimal solutions in some cases.

- For the total tardiness criterion, addressed in Chapter 5, both heuristics and metaheuristics were tested. The proposed method BS-ICH showed the best results, outperforming several methods from the literature. Also, the new heuristic was used in conjunction with metaheuristics. The results show that the used of BS-ICH with metaheuristics significantly improves the solution quality of the algorithms. As a

result, the proposed methods can be seen as a contribution towards the development of new and better metaheuristics and not restricted only to heuristics methods.

6.2 Results

The following papers were published in indexed journals throughout the development of this Thesis.

- ROSSI, F. L.; NAGANO, M. S. Heuristics for the mixed no-idle flowshop with sequence-dependent setup times and total flowtime criterion. **Expert Systems with Applications**, v. 125, p. 40–54, 2019.
- ROSSI, F. L.; NAGANO, M. S.; SAGAWA, J. K. An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion. **The International Journal of Advanced Manufacturing Technology**, v. 90, n. 1, p. 93–107, 2017. ISSN 1433-3015.
- NAGANO, M. S.; ROSSI, F. L.; MARTARELLI, N. J. High-performing heuristics to minimize flowtime in no-idle permutation flowshop. **Engineering Optimization**, Taylor & Francis, v. 0, n. 0, p. 1–14, 2018.
- NAGANO, M. S.; ROSSI, F. L.; TOMAZELLA, C. P. A new efficient heuristic method for minimizing the total tardiness in a no-idle permutation flow shop. **Production Engineering**, v. 11, n. 4, p. 523–529, 2017.
- ROSSI, F. L.; NAGANO, M. S.; NETO, R. F. T. Evaluation of high performance constructive heuristics for the flow shop with makespan minimization. **The International Journal of Advanced Manufacturing Technology**, v. 87, n. 1, p. 125–136, 2016.

6.3 Future research lines

For future work, the constructive heuristic proposed in Chapters 3, 4 and 5 can be considered for other flowshop scheduling problems. Furthermore, since the second phase of the heuristic RN (Section 3.2.2) does not consider the setup times (except for the makespan calculation) it can be easily adapted, including for those problems without no-idle machines or setup times.

The proposal of metaheuristics for the problem remains open with the aim of providing high-quality solutions. The literature review showed that several metaheuristics were proposed for related problems. Therefore, in future studies, the methods developed in Chapters 3 and 4 could be integrated and compared with metaheuristics procedures.

Finally, the mixed no-idle machines and the sequence-dependent setup times can be extended to other manufacturing layouts, i.e hybrid flowshop, job shop, distributed flowshop, among others.

BIBLIOGRAPHY

ADIRI, I.; POHORYLES, D. Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 29, n. 3, p. 495–504, 1982.

ALLAHVERDI, A. The third comprehensive survey on scheduling problems with setup times/costs. **European Journal of Operational Research**, Elsevier, v. 246, n. 2, p. 345–378, 2015.

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. **Omega**, Elsevier, v. 27, n. 2, p. 219–239, 1999.

ALLAHVERDI, A. et al. A survey of scheduling problems with setup times or costs. **European journal of operational research**, Elsevier, v. 187, n. 3, p. 985–1032, 2008.

ALLAHVERDI, A.; SOROUSH, H. M. The significance of reducing setup times/setup costs. **European Journal of Operational Research**, v. 187, n. 3, p. 978–984, 2008.

ARMENTANO, V. A.; RONCONI, D. P. Tabu search for total tardiness minimization in flowshop scheduling problems. **Computers & Operations Research**, v. 26, n. 3, p. 219–235, 1999.

BAGGA, P. C. Minimizing total elapsed time subject to zero total idle time of machines in n x 3 flowshop problem. **Indian J. pure appl. Math**, v. 34, n. 2, p. 219–228, 2003.

BAPTISTE, P.; HGUNY, L. K. A branch and bound algorithm for the F/no- idle/Cmax. In: **Proceedings of the international conference on industrial engineering and production management, IEPM**. [S.l.: s.n.], 1997. v. 97, p. 429–438.

BARAZ, D.; MOSHEIOV, G. A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time. **European Journal of Operational Research**, Elsevier, v. 184, n. 2, p. 810–813, 2008.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. **ORSA journal on computing**, INFORMS, v. 6, n. 2, p. 154–160, 1994.

BENAVIDES, A. J.; RITT, M. Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops. **Computers & Operations Research**, v. 66, n. Supplement C, p. 160–169, 2016.

BENKALAI, I. et al. The migrating birds optimization metaheuristic for the permutation flow shop with sequence dependent setup times. **IFAC-PapersOnLine**, Elsevier, v. 49, n. 12, p. 408–413, 2016.

_____. Improving the migrating birds optimization metaheuristic for the permutation flow shop with sequence-dependent set-up times. **International Journal of Production Research**, Taylor & Francis, v. 55, n. 20, p. 6145–6157, 2017.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M. L. A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. **Management Science**, INFORMS, v. 16, n. 10, p. B630–B637, 1970.

ČEPEK, O. et al. Note: On the two-machine no-idle flowshop problem. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 47, n. 4, p. 353–358, 2000.

CHENG, M.; SUN, S.; YU, Y. A note on flow shop scheduling problems with a learning effect on no-idle dominant machines. **Applied Mathematics and Computation**, Elsevier, v. 184, n. 2, p. 945–949, 2007.

CORWIN, B. D.; ESOGBUE, A. O. Two machine flow shop scheduling problems with sequence dependent setup times: {A} dynamic programming approach. **Naval Research Logistics Quarterly**, Wiley Online Library, v. 21, n. 3, p. 515–524, 1974.

CRAINIC, T. G.; TOULOUSE, M. Parallel strategies for meta-heuristics. In: **Handbook of metaheuristics**. [S.l.]: Springer, 2003. p. 475–513.

CURA, T. An evolutionary algorithm for the permutation flowshop scheduling problem with total tardiness criterion. **International Journal of Operational Research**, Inderscience Publishers, v. 22, n. 3, p. 366–384, 2015.

DANNENBRING, D. G. An Evaluation of Flow Shop Sequencing Heuristics. **Management Science**, v. 23, n. 11, p. 1174–1182, 1977.

DAS, S. R.; GUPTA, J. N. D.; KHUMAWALA. A savings index heuristic algorithm for flowshop scheduling with sequence dependent set-up times. **Journal of the Operational Research Society**, Springer, v. 46, n. 11, p. 1365–1373, 1995.

DENG, G.; GU, X. A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion. **Computers and Operations Research**, v. 39, n. 9, p. 2152–2160, 2012.

DONG, X.; HUANG, H.; CHEN, P. An improved NEH-based heuristic for the permutation flowshop problem. **Computers and Operations Research**, v. 35, n. 12, p. 3962–3968, 2008.

DORIGO, M. et al. **Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings**. [S.l.]: Springer, 2008. v. 5217.

FERNANDEZ-VIAGAS, V. The permutation flowshop scheduling problem: analysis, solution procedures and problem extensions. 2016.

FERNANDEZ-VIAGAS, V.; FRAMINAN, J. M. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. **Computers and Operations Research**, v. 45, p. 60–67, 2014.

_____. NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. **Computers & Operations Research**, v. 60, p. 27–36, 2015.

_____. A beam-search-based constructive heuristic for the {PFSP} to minimise total flowtime. **Computers & Operations Research**, Elsevier, v. 81, p. 167–177, 2017.

FERNANDEZ-VIAGAS, V.; LEISTEN, R.; FRAMINAN, J. M. A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. **Expert Systems with Applications**, v. 61, n. Supplement C, p. 290–301, 2016.

FERNANDEZ-VIAGAS, V.; RUIZ, R.; FRAMINAN, J. M. A new vision of approximate methods for the permutation flowshop to minimise makespan: {S}tate-of-the-art and computational evaluation. **European Journal of Operational Research**, Elsevier, v. 257, n. 3, p. 707–721, 2017.

FERNANDEZ-VIAGAS, V.; VALENTE, J. M. S.; FRAMINAN, J. M. Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. **Expert Systems with Applications**, v. 94, p. 58–69, 2018.

FRAMINAN, J. M.; GUPTA, J. N. D.; LEISTEN, R. A Review and Classification of Heuristics for Permutation Flow-Shop Scheduling with Makespan Objective. **The Journal of the Operational Research Society**, Palgrave Macmillan Journals, v. 55, n. 12, p. 1243–1255, 2004. ISSN 01605682, 14769360.

FRAMINAN, J. M.; LEISTEN, R. An efficient constructive heuristic for flowtime minimisation in permutation flow shops. **Omega**, v. 31, n. 4, p. 311–317, 2003.

_____. Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm. **International Journal of Production Research**, Taylor & Francis, v. 46, n. 22, p. 6479–6498, 2008.

FRAMINAN, J. M.; LEISTEN, R.; RUIZ-USANO, R. Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. **European Journal of Operational Research**, v. 141, n. 3, p. 559–569, 2002.

_____. Comparison of heuristics for flowtime minimisation in permutation flowshops. **Computers and Operations Research**, Pergamon, v. 32, n. 5, p. 1237–1254, may 2005.

GAJPAL, Y.; RAJENDRAN, C.; ZIEGLER, H. An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 30, n. 5-6, p. 416–424, 2006.

GELDERS, L. F.; SAMBANDAM, N. Four simple heuristics for scheduling a flow-shop. **International Journal of Production Research**, Taylor & Francis, v. 16, n. 3, p. 221–231, 1978.

GONCHAROV, Y.; SEVASTYANOV, S. The flow shop problem with no-idle constraints: A review and approximation. **European Journal of Operational Research**, Elsevier, v. 196, n. 2, p. 450–456, 2009.

GRAHAM, R. L. et al. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. **Annals of Discrete Mathematics**, v. 5, p. 287–326, 1979.

GUPTA, J. N. D. A Functional Heuristic Algorithm for the Flowshop Scheduling Problem. **Journal of the Operational Research Society**, v. 22, n. 1, p. 39–47, mar 1971. ISSN 1476-9360.

_____. Heuristic algorithms for multistage flowshop scheduling problem. **AIIE Transactions**, Taylor & Francis, v. 4, n. 1, p. 11–18, 1972.

GUPTA, J. N. D.; DARROW, W. P. The two-machine sequence dependent flowshop scheduling problem. **European Journal of Operational Research**, Elsevier, v. 24, n. 3, p. 439–446, 1986.

HASIJA, S.; RAJENDRAN, C. Scheduling in flowshops to minimize total tardiness of jobs. **International Journal of Production Research**, Taylor & Francis, v. 42, n. 11, p. 2289–2301, 2004.

HUANG, J. D. et al. Minimizing makespan in a two-stage flow shop with parallel batch-processing machines and re-entrant jobs. **Engineering Optimization**, Taylor & Francis, v. 49, n. 6, p. 1010–1023, 2017.

HUNDAL, T. S.; RAJGOPAL, J. An extension of Palmer's heuristic for the flow shop scheduling problem. **International Journal of Production Research**, Taylor & Francis, v. 26, n. 6, p. 1119–1124, 1988.

INCE, Y. et al. A discrete artificial bee colony algorithm for the permutation flowshop scheduling problem with sequence-dependent setup times. In: IEEE. **Evolutionary Computation (CEC), 2016 IEEE Congress on**. [S.l.], 2016. p. 3401–3408.

JOHNSON, S. M. Optimal two-and three-stage production schedules with setup times included. **Naval Research Logistics (NRL)**, Wiley Online Library, v. 1, n. 1, p. 61–68, 1954.

KALCZYNSKI, P. J.; KAMBUROWSKI, J. A Heuristic for Minimizing the Makespan in No-idle Permutation Flow Shops. **Comput. Ind. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 49, n. 1, p. 146–154, 2005.

_____. On no-wait and no-idle flow shops with makespan criterion. **European journal of Operational research**, Elsevier, v. 178, n. 3, p. 677–685, 2007.

_____. On the NEH heuristic for minimizing the makespan in permutation flow shops. **Omega**, v. 35, n. 1, p. 53–60, 2007.

_____. An improved NEH heuristic to minimize makespan in permutation flow shops. **Computers and Operations Research**, v. 35, n. 9, p. 3001–3008, 2008.

_____. An empirical analysis of the optimality rate of flow shop heuristics. **European Journal of Operational Research**, v. 198, n. 1, p. 93–101, 2009.

_____. On Recent Modifications And Extensions Of The Neh Heuristic For Flow Shop Sequencing. **Foundations of Computing and Decision Sciences**, Vol. 36, N, p. 18–33, 2011.

KAMBUROWSKI, J. More on three-machine no-idle flow shops. **Computers & Industrial Engineering**, Elsevier, v. 46, n. 3, p. 461–466, 2004.

KARABULUT, K. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. **Computers & Industrial Engineering**, v. 98, p. 300–307, 2016.

KENDALL, G. et al. Good laboratory practice for optimization research. **Journal of the Operational Research Society**, Taylor & Francis, v. 67, n. 4, p. 676–689, 2016.

KIM, Y.-D. Heuristics for Flowshop Scheduling Problems Minimizing Mean Tardiness. **Journal of the Operational Research Society**, v. 44, n. 1, p. 19–28, jan 1993. ISSN 1476-9360.

KIM, Y.-D.; LIM, H.-G.; PARK, M.-W. Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. **European Journal of Operational Research**, v. 91, n. 1, p. 124–143, 1996.

KOULAMAS, C. A new constructive heuristic for the flowshop scheduling problem. **European Journal of Operational Research**, v. 105, n. 1, p. 66–71, 1998.

LAHA, D.; SARIN, S. C. A heuristic to minimize total flow time in permutation flow shop. **Omega**, v. 37, n. 3, p. 734–739, 2009.

LI, X. et al. Trajectory Scheduling Methods for minimizing total tardiness in a flowshop. **Operations Research Perspectives**, v. 2, p. 13–23, 2015.

LI, X.; WANG, Q.; WU, C. Efficient composite heuristics for total flowtime minimization in permutation flow shops. **Omega**, v. 37, n. 1, p. 155–164, 2009.

LIU, J.; REEVES, C. R. Constructive and composite heuristic solutions to the P//sum C_i scheduling problem. **European Journal of Operational Research**, v. 132, n. 2, p. 439–452, 2001.

LIU, W.; JIN, Y.; PRICE, M. A new improved NEH heuristic for permutation flowshop scheduling problems. **International Journal of Production Economics**, v. 193, n. Supplement C, p. 21–30, 2017.

LOW, C.; YEH, J.-Y.; HUANG, K.-I. A robust simulated annealing heuristic for flow shop scheduling problems. **The International Journal of Advanced Manufacturing Technology**, v. 23, p. 762–767, 2004.

MACCARTHY, B. L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, Taylor & Francis, v. 31, n. 1, p. 59–79, 1993.

MIRABI, M. Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 55, n. 1-4, p. 317–326, 2011.

_____. A novel hybrid genetic algorithm to solve the sequence-dependent permutation flowshop scheduling problem. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 71, n. 1-4, p. 429–437, 2014.

NAGANO, M. S.; MOCCELLIN, J. V. A high quality solution constructive heuristic for flow shop sequencing. **Journal of the Operational Research Society**, v. 53, n. 12, p. 1374–1379, 2002.

_____. Reducing Mean Flow Time in Permutation Flow Shop. **The Journal of the Operational Research Society**, Palgrave Macmillan Journals, v. 59, n. 7, p. 939–945, 2008. ISSN 01605682, 14769360.

NAGANO, M. S.; ROSSI, F. L.; MARTARELLI, N. J. High-performing heuristics to minimize flowtime in no-idle permutation flowshop. **Engineering Optimization**, Taylor & Francis, v. 0, n. 0, p. 1–14, 2018.

NAGANO, M. S.; ROSSI, F. L.; TOMAZELLA, C. P. A new efficient heuristic method for minimizing the total tardiness in a no-idle permutation flow shop. **Production Engineering**, v. 11, n. 4, p. 523–529, 2017.

NARAIN, L.; BAGGA, P. C. Flowshop/no-idle scheduling to minimise the mean flowtime. **The ANZIAM Journal**, Cambridge University Press, v. 47, n. 2, p. 265–275, 2005.

_____. Flowshop/no-idle scheduling to minimize total elapsed time. **Journal of Global Optimization**, Springer, v. 33, n. 3, p. 349–367, 2005.

NAWAZ, M.; ENSCORE, E. E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. **Omega**, v. 11, n. 1, p. 91–95, 1983.

NG, C. T. et al. Flowshop scheduling of deteriorating jobs on dominating machines. **Computers & Industrial Engineering**, Elsevier, v. 61, n. 3, p. 647–654, 2011.

PAGNOZZI, F.; STÜTZLE, T. Speeding up local search for the insert neighborhood in the weighted tardiness permutation flowshop problem. **Optimization Letters**, Springer, v. 11, n. 7, p. 1283–1292, 2017.

PALMER, D. S. Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum. **Journal of the Operational Research Society**, v. 16, n. 1, p. 101–107, mar 1965.

PAN, Q.-K.; RUIZ, R. Local search methods for the flowshop scheduling problem with flowtime minimization. **European Journal of Operational Research**, Elsevier, v. 222, n. 1, p. 31–43, 2012.

_____. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. **Computers and Operations Research**, v. 40, n. 1, p. 117–128, 2013.

_____. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. **Omega**, v. 44, p. 41–50, 2014.

PAN, Q.-K.; WANG, L. A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems. **European Journal of Industrial Engineering**, v. 2, n. 3, p. 279–297, 2008.

_____. No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. **The International Journal of Advanced Manufacturing Technology**, v. 39, n. 7, p. 796–807, nov 2008.

_____. Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. **Omega**, Elsevier, v. 40, n. 2, p. 218–229, 2012.

PARTHASARATHY, S.; RAJENDRAN, C. SCHEDULING TO MINIMIZE MEAN TARDINESS AND WEIGHTED MEAN TARDINESS IN FLOWSHOP AND FLOWLINE-BASED MANUFACTURING CELL. **Computers & Industrial Engineering**, v. 34, n. 2, p. 531–546, 1998.

PESSOA, L. S.; ANDRADE, C. E. Heuristics for a flowshop scheduling problem with stepwise job objective function. **European Journal of Operational Research**, Elsevier, 2017.

PINEDO, M. L. **Scheduling: theory, algorithms, and systems**. [S.l.]: Springer, 2016.

RAD, S. F.; RUIZ, R.; BOROOJERDIAN, N. New high performing heuristics for minimizing makespan in permutation flowshops. **Omega**, v. 37, n. 2, p. 331–345, 2009.

RAJENDRAN, C. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. **International Journal of Production Economics**, v. 29, n. 1, p. 65–73, 1993.

RAJENDRAN, C.; ZIEGLER, H. An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. **European Journal of Operational Research**, v. 103, n. 1, p. 129–138, 1997.

RAMAN, N. Minimum tardiness scheduling in flow shops: Construction and evaluation of alternative solution approaches. **Journal of Operations Management**, v. 12, n. 2, p. 131–151, 1995.

RIBAS, I.; COMPANYS, R.; TORT-MARTORELL, X. Comparing three-step heuristics for the permutation flow shop problem. **Computers and Operations Research**, v. 37, n. 12, p. 2062–2070, 2010.

_____. Efficient heuristics for the parallel blocking flow shop scheduling problem. **Expert Systems with Applications**, v. 74, n. Supplement C, p. 41–54, 2017.

RÍOS-MERCADO, R. Z.; BARD, J. F. Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. **Computers & Operations Research**, v. 25, n. 5, p. 351–366, 1998.

_____. Heuristics for the flow line problem with setup costs. **European Journal of Operational Research**, Elsevier, v. 110, n. 1, p. 76–98, 1998.

_____. A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. **IIE transactions**, Springer, v. 31, n. 8, p. 721–731, 1999.

_____. The flow shop scheduling polyhedron with setup times. **Journal of Combinatorial Optimization**, Springer, v. 7, n. 3, p. 291–318, 2003.

RÍOS-MERCADO, R. Z. et al. An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times. **Journal of Heuristics**, Springer, v. 5, n. 1, p. 53–70, 1999.

ROSSI, F. L.; NAGANO, M. S.; NETO, R. F. T. Evaluation of high performance constructive heuristics for the flow shop with makespan minimization. **The International Journal of Advanced Manufacturing Technology**, v. 87, n. 1, p. 125–136, 2016.

ROSSI, F. L.; NAGANO, M. S.; SAGAWA, J. K. An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion. **The International Journal of Advanced Manufacturing Technology**, v. 90, n. 1, p. 93–107, 2017.

RUIZ, R.; MAROTO, C.; ALCARAZ, J. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. **European Journal of Operational Research**, Elsevier, v. 165, n. 1, p. 34–54, 2005.

_____. Two new robust genetic algorithms for the flowshop scheduling problem. **Omega**, Elsevier, v. 34, n. 5, p. 461–476, 2006.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 177, n. 3, p. 2033–2049, 2007.

_____. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. **European Journal of Operational Research**, Elsevier, v. 187, n. 3, p. 1143–1159, 2008.

RUIZ, R.; VALLADA, E.; FERNÁNDEZ-MARTÍNEZ, C. Scheduling in Flowshops with No-Idle Machines. In: CHAKRABORTY, U. K. (Ed.). **Computational Intelligence in Flow Shop and Job Shop Scheduling**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 21–51.

SAADANI, N. E. H.; GUINET, A.; MOALLA, M. Three stage no-idle flow-shops. **Computers & industrial engineering**, Elsevier, v. 44, n. 3, p. 425–434, 2003.

_____. A travelling salesman approach to solve the F/no-idle/Cmax problem. **European Journal of Operational Research**, v. 161, n. 1, p. 11–20, 2005.

SEN, T.; GUPTA, S. K. A state-of-art survey of static scheduling research involving due dates. **Omega**, v. 12, n. 1, p. 63–76, 1984.

SEVAST'JANOV, S. Vector Summation in Banach Space and Polynomial Algorithms for Flow Shops and Open Shops. **Mathematics of Operations Research**, v. 20, n. 1, p. 90–103, 1995.

SHAO, W.; PI, D.; SHAO, Z. Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion. **Applied Soft Computing**, v. 54, n. Supplement C, p. 164–182, 2017.

SHEIBANI, K. A fuzzy greedy heuristic for permutation flow-shop scheduling. **Journal of the Operational Research Society**, Springer, v. 61, n. 5, p. 813–818, 2010.

SHEN, J.-n.; WANG, L.; WANG, S.-y. A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. **Knowledge-Based Systems**, Elsevier, v. 74, p. 167–175, 2015.

Simons Jr, J. V. Heuristics in flow shop scheduling with sequence dependent setup times. **Omega**, Elsevier, v. 20, n. 2, p. 215–225, 1992.

SIOUD, A.; GAGNÉ, C. Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times. **European Journal of Operational Research**, Elsevier, v. 264, n. 1, p. 66–73, 2018.

SLACK, N. et al. **Administração da produção**. [S.l.]: Atlas São Paulo, 2009. v. 2.

SRIKAR, B. N.; GHOSH, S. A {MILP} model for the n-job, m-stage flowshop with sequence dependent set-up times. **International Journal of Production Research**, Taylor & Francis, v. 24, n. 6, p. 1459–1474, 1986.

Stafford Jr, E. F.; TSENG, F. T. On the Srikar-Ghosh MILP model for the iV x M SDST flowshop problem. **The International Journal Of Production Research**, Taylor & Francis, v. 28, n. 10, p. 1817–1830, 1990.

_____. Two models for a family of flowshop sequencing problems. **European Journal of Operational Research**, Elsevier, v. 142, n. 2, p. 282–293, 2002.

STÜTZLE, T. An Ant Approach to the Flow Shop Problem. In: **In Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98**. [S.l.]: Verlag, 1997. p. 1560–1564.

SUN, L. L. L. et al. A note on flow shop scheduling problems with deteriorating jobs on no-idle dominant machines. **European Journal of Operational Research**, Elsevier, v. 200, n. 1, p. 309–311, 2010.

SUN, L.-Y. L.-H. L.-Y. et al. Flow shop makespan minimization scheduling with deteriorating jobs under dominating machines. **International Journal of Production Economics**, Elsevier, v. 138, n. 1, p. 195–200, 2012.

TAILLARD, E. Some efficient heuristic methods for the flow shop sequencing problem. **European journal of Operational research**, Elsevier, v. 47, n. 1, p. 65–74, 1990.

_____. Benchmarks for basic scheduling problems. **european journal of operational research**, Elsevier, v. 64, n. 2, p. 278–285, 1993.

TASGETIREN, M. F. et al. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. **European journal of operational research**, Elsevier, v. 177, n. 3, p. 1930–1947, 2007.

_____. A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. **International Journal of Production Research**, Taylor & Francis, v. 49, n. 16, p. 5033–5050, 2011.

_____. A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion. **Applied Mathematical Modelling**, Elsevier, v. 37, n. 10, p. 6758–6779, 2013.

_____. A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. **Computers and Operations Research**, v. 40, n. 7, p. 1729–1743, 2013.

TSENG, F. T.; GUPTA, J. N. D.; Stafford Jr, E. F. A penalty-based heuristic algorithm for the permutation flowshop scheduling problem with sequence-dependent set-up times. **Journal of the Operational Research Society**, Taylor & Francis, v. 57, n. 5, p. 541–551, 2006.

TSENG, F. T.; Stafford Jr, E. F. Two MILP models for the N x M SDST flowshop sequencing problem. **International Journal of Production Research**, Taylor & Francis, v. 39, n. 8, p. 1777–1809, 2001.

USKUP, E.; SMITH, S. B. A branch-and-bound algorithm for two-stage production-sequencing problems. **Operations Research**, INFORMS, v. 23, n. 1, p. 118–136, 1975.

VACHAJITPAN, P. Job sequencing with continuous machine operation. **Computers & Industrial Engineering**, Elsevier, v. 6, n. 3, p. 255–259, 1982.

VALLADA, E.; RUIZ, R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. **Omega**, v. 38, n. 1, p. 57–67, 2010.

VANCHIPURA, R.; SRIDHARAN, R. Development and analysis of constructive heuristic algorithms for flow shop scheduling problems with sequence-dependent setup times. **The International Journal of Advanced Manufacturing Technology**, v. 67, n. 5, p. 1337–1353, jul 2013.

VANCHIPURA, R.; SRIDHARAN, R.; BABU, A. S. Improvement of constructive heuristics using variable neighbourhood descent for scheduling a flow shop with sequence dependent setup time. **Journal of Manufacturing Systems**, v. 33, n. 1, p. 65–75, 2014.

VOLLMANN, T. E. **Manufacturing planning and control for supply chain management**. [S.l.: s.n.], 2005.

WANG, Y. et al. Iterated local search algorithms for the sequence-dependent setup times flow shop scheduling problem minimizing makespan. In: **Foundations of Intelligent Systems**. [S.l.]: Springer, 2014. p. 329–338.

WOO, H.-S.; YIM, D.-S. A heuristic algorithm for mean flowtime objective in flowshop scheduling. **Computers and Operations Research**, v. 25, n. 3, p. 175–182, 1998.

WOOLLAM, C. R. Flowshop with no idle machine time allowed. **Computers & industrial engineering**, Elsevier, v. 10, n. 1, p. 69–76, 1986.

ZHOU, Y.; CHEN, H.; ZHOU, G. Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem. **Neurocomputing**, v. 137, p. 285–292, 2014.