UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Viviane Souza Vilela Junqueira dos Reis

# The No-Wait Flow Shop Scheduling Problem with Sequence-Dependent Setup Times: a comprehensive review and an application of the ALNS algorithm

São Carlos

2021

**Viviane Souza Vilela Junqueira dos Reis**

# The No-Wait Flow Shop Scheduling Problem with Sequence-Dependent Setup Times: a comprehensive review and an application of the ALNS algorithm

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

Área de concentração: Processos e Gestão de Operações

Advisor: Prof. Dr. Marcelo Seido Nagano

**Corrected version**
**(Original version available on the Program Unit)**

**São Carlos**
**2021**

Junqueira, Viviane Souza Vilela
S856m The No-Wait Flow Shop Scheduling Problem with Sequence-Dependent Setup Times: a comprehensive review and an application of the ALNS algorithm / Viviane Souza Vilela Junqueira dos Reis ; orientador Marcelo Seido Nagano. – São Carlos, 2021.
72 p. : il. (algumas color.) ; 30 cm.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia de Produção e Área de Concentração em Processos e Gestão de Operações – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2021.

1. LaTeX. 2. abnTeX. 3. Classe USPSC. 4. Editoração de texto. 5. Normalização da documentação. 6. Tese. 7. Dissertação. 8. Documentos (elaboração). 9. Documentos eletrônicos. I. Nagano, Marcelo Seido, orient. II. Título.

# FOLHA DE JULGAMENTO

Candidata: Engenheira **VIVIANE SOUZA VILÉLA JUNQUEIRA DOS REIS**.

Título da dissertação: "O problema de sequenciamento *no-wait flow shop* com tempos de *setup* dependentes da sequência: uma revisão abrangente e uma aplicação do algoritmo ALNS".

Data da defesa: 10/11/2021

| **Comissão Julgadora** | **Resultado** |
|---|---|
| Prof. Dr. **Marcelo Seido Nagano**<br>(Orientador)<br>(Escola de Engenharia de São Carlos/EESC-USP) | Aprovado |
| Prof. Dr. **Leandro Callegari Coelho**<br>(Université Laval/ULaval) | Aprovado |
| Prof. Dr. **Roberto Fernandes Tavares Neto**<br>(Universidade Federal de São Carlos/UFSCar) | Aprovado |

Coordenadora do Programa de Pós-Graduação em Engenharia de Produção:

Profa. Dra. **Janaina Mascarenhas Hornos da Costa**

Presidente da Comissão de Pós-Graduação:

Prof. Titular **Murilo Araujo Romero**

## ACKNOWLEDGEMENTS

I would like to thank the University of Sao Paulo, the Brazilian and Canadian research funding agencies, and the professors and employees of the production engineering department at the Sao Carlos School of Engineering, for the great support received during my master's, which made it possible. Special thanks to professors Marcelo, Maryam and Leandro who guided me in this project.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -Brasil (CAPES) -Finance Code 001; the National Council for Scientific and Technological Development (CNPq) under grants 165306/2018-1, 430137/2018-4 and 306075/2017-2; the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2019-00094 and 2020-00401; and the Emerging Leaders in the Americas Program (ELAP) scholarship. This support is greatly acknowledged.

## ABSTRACT

JUNQUEIRA, V.S.V. **The No-Wait Flow Shop Scheduling Problem with Sequence-Dependent Setup Times: a comprehensive review and an application of the ALNS algorithm**. 2021. 72p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

One of the most important decisions in any manufacturing company is how to schedule the operations on the available machines. In several industries, the nature of the job imposes certain constraints to operations scheduling. In a no-wait flow shop, once a job starts on the first machine, it has to continue being processed on the next ones, without any interruptions. As an extension of the flow shop scheduling, the no-wait version is also very difficult to be solved to optimality within a reasonable time, and many heuristics have been proposed for this problem. This work aims to classify existing solution algorithms proposed to solve the no-wait flow shop scheduling problem with setup times and some of its variants. We show how combining a heuristic to generate a good initial solution, local search procedures, insertion and swapping of job positions and techniques developed originally to solve transportation problems are among the popular and efficient techniques for the problem at hand. We also propose a new solution method based on the well-known Adaptive Large Neighborhood Search (ALNS) algorithm from transportation science. The use of this algorithm aims to minimize the total flow time as a performance measure. As this is a problem with high complexity, to achieve high solution quality in a reasonable time, an acceleration method was also adapted and applied into local search procedures with swapping operations. The results of the new method were compared to the best results in the literature for widespread instances, validating the quality of the method.



**Keywords**: Flow shop scheduling. No-wait. Setup. ALNS.

# RESUMO

JUNQUEIRA, V.S.V. **O problema de sequenciamento no-wait flow shop com tempos de setup dependentes da sequência: uma revisão abrangente e uma aplicação do algoritmo ALNS**. 2021. 72p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

Uma das decisões mais importantes em empresas de manufatura é como sequenciar operações nas máquinas disponíveis. Em várias industrias, a natureza do trabalho impõe certas restrições ao sequenciamento de tarefas. Em um ambiente no-wait flow shop, uma vez que uma tarefa é iniciada na primeira máquina, ela deve continuar sendo processada nas próximas máquinas sem esperas. Assim como o problema de sequenciamento de flow shop, a variação no-wait também traz grande dificuldade de ser otimizada dentro de um tempo razoável, por isso várias heurísticas foram propostas para esse problema. Este trabalho tem como objetivo classificar os algoritmos de solução propostos para resolver o problema de sequenciamento no-wait flow shop com tempos de setup dependentes da sequência e algumas variantes. Mostramos como combinar uma heurística para gerar uma boa solução inicial, procedimentos de busca local, inserção e troca de posições de tarefas e técnicas desenvolvidas originalmente para resolver problemas de transporte estão entre as técnicas mais populares e eficientes para o problema em questão. Também é proposto um novo método de solução baseado no algoritmo Adaptive Large Neighbourhood Search (ALNS) da ciência dos transportes. A utilização deste algoritmo visa minimizar o tempo total de fluxo (TFT) como medida de desempenho. Por se tratar de um problema de alta complexidade, para atingir alta qualidade de solução em um tempo razoável, um método de aceleração também foi adaptado e aplicado em procedimentos de busca local com operações de swap. Os resultados do novo método foram comparados aos melhores resultados da literatura para instâncias bem conhecidas, validando assim a qualidade do método.

**Palavras-chave**: Flow shop scheduling. No-wait. Setup. ALNS.

## LIST OF FIGURES

## LIST OF TABLES

## CONTENTS

# 1 INTRODUCTION

This dissertation unites two works on the no-wait flow shop scheduling problem (NWFSP) with sequence dependent setup times. In the first part a comprehensive review of the literature classifies the solution algorithms proposed for the problem by type of setup time and highlights how these methods are compared among them. Future research suggestions were also made based on the analysis of the review. In the second part of this essay we present a method to solve the problem computationally. This method was adapted from the ALNS algorithm, which was first designed for transportation science problems, to solve a scheduling problem. Afterwards, the method was compared to the well-known Iterated Greedy (IG) algorithm and the results were explored to show the good performance of the ALNS over the IG.

The dissertation is organized as follows. Chapter 2 describes the problem and its importance. It brings details on the classification of setup times, the no-wait constraint, and the complexity of the problem. It presents a review of the literature with the solution methods optimizing the most important performance criteria. It also introduces some perspectives for future research. Chapter 3 presents the mathematical formulation of the problem followed by a description of the techniques used to solve it considering the TFT as objective function. Afterwards, the results of the computational experiments are showed and some conclusions are made based on the performances of the proposed method and the method used as comparison. Chapter 4 brings some final considerations.

# 2 LITERATURE REVIEW

## 2.1 Introduction

The flow shop scheduling problem is among the most studied problems in the scheduling literature and has many applications in manufacturing and service industries. In this problem, jobs follow identical orders over several machines in series (RUIZ; STÜTZLE, 2007; PINEDO, 2008). Maccarthy and Liu (1993) classified scheduling problems considering the type of machine shop environments based on the job flow patterns on these machines. However, the classification includes other well-known environments, such as single machine, parallel machines, and even the job shop. To date several reviews have addressed this very practical optimization problem (FRAMINAN; GUPTA; LEISTEN, 2004; HEJAZI; SAGHAFIAN, 2005; RUIZ; MAROTO, 2005; PAN; RUIZ, 2013).

In this research, we focus on the no-wait flow shop scheduling problem with setup times, which is an important variant of the flow shop scheduling problem (FSP). In this problem, jobs need to be processed with no interruptions on consecutive machines and setup operations are executed before processing each job. A number of studies have provided a significant theoretical basis into the no-wait flow shop problem, e.g., Nagano and Miyata (2016) and Allahverdi (2016). For the flow shop scheduling problem with setup times, we can refer to Allahverdi (2015). This review updates the former ones as we focus on the algorithmic contributions for this NP-hard problem.

The main objective of this review is to provide a comprehensive overview of the problem, its variants, and the solution algorithms proposed in the literature. To the best of our knowledge, no recent review and classification of solution algorithms exist. This work provides insights on the strengths and weaknesses of the proposed solution algorithms. It identifies the variants being studied and the methods with the best performances. It also contributes to the proposition of new heuristics by identifying the most effective mechanisms for different stages of a solution algorithm. Thus, it could be used as a guide for future scheduling research.

The remainder of this study is organized as follows. After providing a general overview on the FSP in Section 2.2, we formally introduce the no-wait flow shop scheduling problem and its variant with setup times in Section 2.3. A classification based on the type of setup is presented in Section 3.2. Section 2.5 brings a recent extension of the problem, the mixed no-wait. We present the conclusions and some perspectives for future research in Section 2.7.

## 2.2 Flow Shop Scheduling Problem

In the FSP, products pass in the same order through several machines disposed in series. Consider a set $J = \{1, ..., n\}$ of independent jobs that have to be processed on a set $M = \{1, ..., m\}$ of machines in the same order and without preemption. In other words, every job $j \in N$ follows the same sequence from the first to the last machine, and has a deterministic and non-negative processing time $p_{ij}$ on every machine $i \in M$ (RUIZ; MAROTO, 2005; RUIZ; STÜTZLE, 2007; PINEDO, 2008). The objective of the FSP is to find a schedule of all products that optimizes a given criterion. Thus, to formulate a scheduling problem mathematically one needs to identify the constraints faced by the company in using its resources or imposed by the processes as well as to determine and prioritize company's key performance indicators (MACCARTHY; LIU, 1993).

Pinedo (2008) and Baker and Trietsch (2009) describe some of the most common performance measures. These measures are usually a function of job completion time $C_j$. The objective function may also be related to job due dates. An important performance measure is the minimization of total flow time (TFT) or total completion time (TCT), which represents the sum of each of the job flow time, the time each job spends in the system from its release time (if exists) until its completion time. The mostly used criterion in the literature is the minimization of the maximum completion time $C_{max} = max_{1 \leq j \leq n}\{C_j\}$, known as the makespan.

These performance measures may have conflicting natures, and the choice of a specific measure depends on the company's strategy and its industry. For example, minimizing the makespan can be associated with the efficient use of resources; therefore, it is appropriate for a strategy that aims to minimize machine idle times or maximizing machine utilization rates. It also applies for manufacturing systems that aim to dispatch a complete batch of products as soon as possible. On the other hand, the TFT objective is commonly used to reflect the average flow time of all orders in the system. In practice, it is known to increase production rate and decrease work-in-process (WIP). Thus, it can be applied to meet the demand of individual products as soon as it is completed and to provide a rapid response to demands (RUIZ; ALLAHVERDI, 2007b).

### 2.2.1 Complexity of the flow shop scheduling problem

The FSP with $m \geq 3$ is strongly NP-hard for all main performance measures (FRAMINAN; GUPTA; LEISTEN, 2004; RUIZ; MAROTO, 2005). Thus, finding an optimal sequence becomes exponentially difficult as the problem size increases, which makes heuristic methods more appropriate than the exact ones (HEJAZI; SAGHAFIAN, 2005).

The version of the problem considered in this paper requires that the processing

sequence remains unchanged on each operation. Thus, the *First In First Out (FIFO)* rule is applied to the queues of jobs on machines (RUIZ; MAROTO, 2005). This classical flow shop problem is referred to as the permutation flow shop scheduling problem (PFSP) and it is also NP-hard (RUIZ; STÜTZLE, 2007; PINEDO, 2008; SUN *et al.*, 2011).

### 2.2.2 Variants of the flow shop scheduling problem

The basic version of the PFSP cannot represent many real world problems, so other characteristics are embedded as side constraints to the PFSP (SUN *et al.*, 2011). These include the no-wait, no-idle, setup times, and release times. Mixed environments including operations from the PFSP and from its variants, such as the no-wait flow shop, can also be considered to better represent a real case.

In the PFSP, an unbound buffer is assumed to exist between machines, so jobs can wait to be processed. As this condition is not valid for most real cases, some variants of the problem arise to represent the case where jobs cannot wait between machines. In the no-wait variant, jobs do not form queues on or between machines, as operations must be processed without interruptions. On the other hand, in a no-idle flow shop, the machines work without any interruption.

Other variants that better describe a flow shop productive system are those with release and setup times constraints. A job release time defines the moment a job is ready for its first operation. The setup time represents the time spent to prepare machines to process a sequence of jobs, which can include obtaining and adjusting tools, inspecting and positioning WIP material, setting jigs and fixtures, and cleaning machines.

A single constraint is not always representative of a complex real-life situation, and in most scheduling applications more than one requirement is considered to better represent the reality. In this study, two of these constraints are reviewed under the No-Wait Flow Shop Scheduling Problem (NWFSP) with setup times.

## 2.3 No-Wait Flow Shop Scheduling Problem with Setup Times

The no-wait requirement in scheduling problems appears when interruptions are not allowed between two operations of a job on consecutive machines, i.e., there must be a continuous flow of operations along the production line. For this to happen, a delay on the first machine, which can cause idle time on others, may be imposed to guarantee the production without waiting times (PAN; RUIZ, 2013). This constraint represents numerous real-life applications, which explains the attention it has received since the 1970s. It occurs mainly in industries aiming to reduce the WIP inventory and in those with specific characteristics in their raw materials and manufacturing processes. Examples include the plastic molding and silverware production, in which a no-wait constraint is applied to prevent the degradation of the material being processed (NAGANO; MIYATA,

2016). In the steel and chemical industries, no-wait is a property of the production process; as in the first case, the high temperature steel must not wait between two consecutive operations or the material would have to be reheated, generating a waste of energy. This constraint is also found in the service industry, such as in surgery scheduling, where patients are better cared when a continuous service is offered (ALLAHVERDI, 2016).

Additionally, setup times were first included in a flow shop context by Corwin and Esogbue (1974) for a 2-machine setting. Although the majority of scheduling studies still ignores this requirement, setup operations are commonly found in manufacturing and service industries. The time spent for preparation of resources to execute the tasks of a sequence of jobs does not add value to the final customer, but it generates a cost. Ignoring setup times in a scheduling study can compromise the solution quality. Some applications in which setup times play an important role are those where different products are manufactured by the same multi-function machines (RUIZ; MAROTO; ALCARAZ, 2005). Moreover, in the textile and plastic industries a setup is required when products with different colors are produced. In these cases, the duration of the setup will also depend on the permutation of products in the production sequence (RUIZ; ALLAHVERDI, 2007a). Another example arises in the manufacturing of electronic products, such as assembling printed circuit board and manufacturing semiconductors (ALLAHVERDI, 2015).

When setup time is not sequence-dependent it only depends on the job $j$ to be processed on machine $i$ and is denoted as $s_{ij}$. Setup time is also handled better when considered separately from the processing time, as job setup on a subsequent machine can be performed while the job is still in process on the preceding machine. This improves machine utilization and may reduce the total completion time (ALDOWAISAN; ALLAHVERDI, 1998).

Sequence-dependent setup times cannot be incorporated into job processing times and must be defined for each possible sequence of jobs. The sequence-dependent setup time $s_{ilj}$ is then the time required to configure machine $i$ when job $j$ is scheduled immediately after job $l$.

Importantly, when both characteristics, no-wait and setup, are considered simultaneously, setup operations must be executed on machines before a job's arrival to satisfy the no-wait constraint.

## 2.4 The literature on NWFSP with Setup Times

In order to determine the best existing methods in the literature to solve the problem under analysis, a comprehensive review on the deterministic NWFSP with setup times has been conducted. Studies that aim to minimize a single objective are initially classified according to the type of setup time. Then, the criterion being optimized, the

number of machines in the shop and the solution algorithms are specified. In addition, comparisons made by the authors against existing methods are also analysed. It will be shown that due to the NP-hard nature of the problem, most of the proposed solution methods are heuristic and metaheuristic algorithms.

To describe the variants of the problem in a standardized manner, the $\alpha/\beta/\gamma$ notation introduced by Graham *et al.* (1979) is used where $\alpha$ characterizes the production environment according to the machine setting and the number of machines, for example $\alpha$ as $F_m$ represents a flow shop environment ($F$) with $m$ machines, $\beta$ defines additional constraints and processing characteristics, such as setup and no-wait, and finally $\gamma$ describes the objective to be optimized.

According to Allahverdi (2015), scheduling problems considering setup times are primarily classified as family versus non-family. Jobs can be divided into families due to similarities in terms of setup. Then, only setup times between jobs from different families are needed. Non-family means that all jobs have their own setup times.

This chapter is then organized as follows. The NWFSP with non-family sequence-independent setup times is presented in Section 2.4.1. Papers with non-family sequence-dependent setup times are presented in Section 2.4.2. Family setup time studies appear in Section 2.4.3.

### 2.4.1 Non-Family Sequence-Independent Setup Time

Gupta, Strusevich and Zwaneveld (1997) introduced the NWFSP with setup times under a $F2|no\text{-}wait, s_{ij}, r_{ij}|C_{max}$ problem where pre- and post-operational tasks, such as setups and removal times, are separated from processing times. The authors reduced this problem to the travelling salesman problem (TSP) to minimize makespan and showed that it is solvable in $O(n \log n)$ time by the Gilmore-Gomory algorithm. Aldowaisan and Allahverdi (1998) also addressed the two-machine NWFSP with distinct setup time, but optimizing the TFT instead of the makespan. The authors developed an optimal solution algorithm for two special cases and a heuristic procedure for the general problem.

Aldowaisan (2001) proposed a new heuristic with local and global dominance relations and a lower bound was generated to be used in a branch-and-bound algorithm and to evaluate the new heuristic, which performs better than the previous one. Moreover, Aldowaisan and Allahverdi (2004), based on the concepts of blocking (insertion technique) and looping, presented several heuristics to solve the three-machine NWFSP with separate setup and removal times in order to minimize the TCT. The combination of blocking and looping yields the best solutions.

Extending the work by Aldowaisan and Allahverdi (1998), Brown, McGarvey and Ventura (2004) introduced the $m$-machine NWFSP with sequence-independent setup time.

The authors presented a heuristic to minimize TFT ($Fm|no\text{-}wait, s_{ij}|\sum C_j$) and makespan ($Fm|no\text{-}wait, s_{ij}|C_{max}$). They show that the makespan minimization for this scheduling problem is the equivalent of the TSP, while minimizing the TFT can be compared to the time-dependent TSP, that is, the sequence of the jobs influences the TFT calculation. The authors concluded that their heuristic, named TRIPS as it examines all possible three-job combinations, performed better than a standard Simulated Annealing (SA) algorithm for the TFT objective function, but worse than the Lin-Kernighan-Helsgaun (LKH) algorithm for the makespan criterion.

Sidney, Potts and Sriskandarajah (2000) presented an approximate algorithm for the $F2|no\text{-}wait, setup\{c_j; d_j\}|C_{max}$ problem with anticipatory setups on the second machine divided into two parts, where $c_j$ denotes the first portion of setup time and $d_j$ the total setup time of job $j$ on the second machine. The authors also applied the Gilmore-Gomory algorithm to sequence the jobs and established the worst-case performance ratio of their heuristic in comparison with the optimal solution value.

Dileepan (2004) and Fondrevelle, Allahverdi and Oulamara (2005) studied the maximum lateness $L_{max}$ minimization for a two-machine NWFSP, measuring the worst violation of the due dates. Dileepan (2004) continued the work of Aldowaisan and Allahverdi (1998), modifying their criterion and presenting theoretical results. Fondrevelle, Allahverdi and Oulamara (2005) added removal times to the problem and presented a branch-and-bound algorithm to solve the generic case. They also integrated the dominance property of Dileepan (2004) and showed that it performs efficiently, especially when setup and removal times are not too large compared to processing times.

Ruiz and Allahverdi (2007b) minimized the TCT ($F_m|no\text{-}wait, s_{ij}|\sum C_i$), while Ruiz and Allahverdi (2007a) minimized the $L_{max}$ ($F_m|no\text{-}wait, s_{ij}|L_{max}$). Ruiz and Allahverdi (2007b) proposed a dominance rule for the four-machine case and incorporated it into five presented heuristic algorithms. Ruiz and Allahverdi (2007a) presented several heuristics based on dispatching rules and four variants of a genetic algorithm (GA). In addition, a dominance relation for the three-machine case is considered, which speeds up the algorithm processing time. The proposed heuristics and GAs were compared with those proposed by Ruiz and Allahverdi (2007b), adapted to $L_{max}$, since there were no other comparisons available. The results showed that the GAs generate better results than local searches.

Su and Lee (2008) and Samarghandi and ElMekkawy (2011) introduced the concept of single server in the two-machine context, where setup operations on both machines cannot be overlapped due to the involvement of the server in these operations. Su and Lee (2008) studied the $F2, S1|no\text{-}wait, s_{ij}|\sum C_i$ problem. To minimize the TCT, the authors proposed a heuristic and a branch-and-bound algorithm. They compared their algorithms with those of Aldowaisan (2001) and showed that the new heuristic has superior solution quality. Samarghandi and ElMekkawy (2011) studied the same problem but with a different

objective function. To solve the $F2, S1|no\text{-}wait, s_{ij}|C_{max}$ problem, the authors proposed a hybrid of variable neighborhood search (VNS) and tabu search (TS). Computational experiments demonstrated the efficiency of the proposed algorithm in quickly finding near optimal solutions by comparing them to lower bounds, also outperforming a simple 2-opt.

The most recent papers on the NWFSP with sequence-independent setup times addressed the $m$-machine case and applied different heuristics and metaheuristics to minimize some performance measures. Samarghandi and Elmekkawy (2012) optimized the makespan using a GA and a hybrid algorithm composed of a GA and a particle swarm optimization (PSO) algorithm. In order to demonstrate the efficiency and effectiveness of the developed algorithms, the authors applied the new methods to several instances from the literature and compared the results with those of the 2-opt algorithm, which was modified to this problem. It was also shown that the hybrid algorithm outperformed the proposed GA.

Later, Nagano, Silva and Lorena (2012) studied the aforementioned problem optimizing the TCT. The authors proposed a metaheuristic called Evolutionary Clustering Search for No-Wait Flowshop with Setup Times ($ECS\_NSL$). This method was compared to those of Ruiz and Allahverdi (2007b). The results showed that the new method obtained superior performance for large instances, while for small instances both methods presented similar solution quality.

In conclusion, the literature has exploited the fact that the NWFSP with sequence-independent setup time with makespan minimization can be modeled as a well-known TSP. In the case of TFT minimization, the problem can be modeled as a time-dependent TSP, for which extensive research is available. This problem benefits from well established datasets that many authors used to compare the performance of their methods, notably against the local search and constructive algorithms (BROWN; MCGARVEY; VENTURA, 2004; RUIZ; ALLAHVERDI, 2007b). The best known results evolved over time, and are currently held by Nagano, Silva and Lorena (2012). Table 1 summarizes the papers studied in this section.

### 2.4.2 Non-Family Sequence-Dependent Setup Time

The literature on the NWFSP with non-family sequence-dependent setup time is more developed than sequence-independent literature. Bianco, Dell'olmo and Giordani (1999) introduced the problem considering job release dates (ready times) $r_j$. The $F_m|no\text{-}wait, r_j, s_{ijk}|C_{max}$ problem was modeled as an Asymmetric Travelling Salesman Problem with Ready Times (ATSP-RT), in order to minimize the makespan. Two greedy heuristic algorithms were proposed and their performances were compared with two proposed lower bounds. They concluded that inserting jobs within a partial sequence yields better results than adding jobs at the end of a partial sequence.

Table 1 – NWFSP with Non-Family Sequence-Independent Setup Time

| **Author** | **Problem** | **Method** | **Comparison** |
|---|---|---|---|
| Gupta, Strusevich and Zwaneveld (1997) | $F2\|no-wait, s_{ij}, r_{ij}\|C_{max}$ | Gilmore-Gomory algorithm | n/a |
| Aldowaisan and Allahverdi (1998) | $F2\|no-wait, s_{ij}\|\sum C_j$ | Heuristic algorithm | n/a |
| Sidney, Potts and Sriskandarajah (2000) | $F2\|no-wait, s\{c_j; d_j\}\|C_{max}$ with anticipatory setup | Heuristic algorithm | n/a |
| Aldowaisan (2001) | $F2\|no-wait, s_{ij}\|\sum C_j$ | Heuristic algorithm | Aldowaisan and Allahverdi (1998) |
| Aldowaisan and Allahverdi (2004) | $F3\|no-wait, s_{ij}, r_{ij}\|\sum C_j$ | Several heuristic algorithms | Between them |
| Brown, McGarvey and Ventura (2004) | $Fm\|no-wait, s_{ij}\|\sum C_j$ and $Fm\|no-wait, s_{ij}\|C_{max}$ | TRIPS heuristic | SA and LHK algorithms |
| Dileepan (2004) | $F2\|no-wait, s_{ij}\|L_{max}$ | Heuristic algorithm | n/a |
| Fondrevelle, Allahverdi and Oulamara (2005) | $F2\|no-wait, s_{ij}, r_{ij}\|L_{max}$ | Branch-and-Bound algorithm | Dileepan (2004) |
| Ruiz and Allahverdi (2007a) | $F_m\|no-wait, s_{ij}\|L_{max}$ | Several heuristics and four GA algorithms | Ruiz and Allahverdi (2007b) |
| Ruiz and Allahverdi (2007b) | $F_m\|no-wait, s_{ij}\|\sum C_j$ | Five heuristics and two stochastic local search algorithms | Brown, McGarvey and Ventura (2004) and Shyu, Lin and Yin (2004) |
| Su and Lee (2008) | $F2, S1\|no-wait, S_{ij}\|\sum C_j$ | Heuristic algorithm | Aldowaisan (2001) |
| Samarghandi and ElMekkawy (2011) | $F2, S1\|no-wait, S_{ij}\|C_{max}$ | Hybrid algorithm (VNS and TS) | 2-opt algorithm |
| Samarghandi and Elmekkawy (2012) | $F_m\|no-wait, S_{ij}\|C_{max}$ | GA and hybrid algorithm (GA and PSO) | 2-opt algorithm |
| Nagano, Silva and Lorena (2012) | $F_m\|no-wait, S_{ij}\|\sum C_j$ | Evolutionary Clustering Search | Ruiz and Allahverdi (2007b) |

Source: Elaborated by the author.

Note: n/a means that a comparison was not made.

Allahverdi and Aldowaisan (2001) and Shyu, Lin and Yin (2004) continued the research on the two-machine problem. Allahverdi and Aldowaisan (2001) derived optimal solutions for two special cases with the TCT minimization. Differently from the previous study, these authors considered a specific sequence-dependent setup called changeover time, which consists of a setdown operation from the previous job and a setup operation for the next one. Five two-phase heuristics are developed: they are all composed of a constructive phase and an improvement phase, based on previous insertion heuristics. Results show that regardless of the initial sequence, the solution converges, confirming the success of the repeated application of an insertion technique. Shyu, Lin and Yin (2004) minimized the TCT with job-dependent and machine-dependent setup times with an ant colony optimization (ACO) metaheuristic. This study extended TSP formulations by modeling the TCT minimization as a cumulative TSP, another problem stemming from the distribution literature. The graph representation was applied in the initialization phase of the metaheuristic. To evaluate the effectiveness of the ACO, the authors compared it to the heuristics of Aldowaisan and Allahverdi (1998) and Aldowaisan (2001) for the same problem and with an algorithm based on the 3-opt operation initially developed for the cumulative TSP. The 3-opt heuristic demonstrated better performance for small instances, while the ACO algorithm outperformed the compared heuristics in solution quality for large instances, at the expense of longer runtimes.

Returning to the makespan minimization, Lee and Jung (2005) studied the problem under precedence constraints. Three algorithms are proposed: a metaheuristic based on the SA, a heuristic with insertion techniques, and a hybrid of the first two (the solution of the heuristic is used as an initial solution for the metaheuristic). Results showed that the integrated algorithm obtain better results. Shortly after, França, Jr. and Buriol (2006) considered release dates to the problem, resulting in the same problem studied by Bianco, Dell'olmo and Giordani (1999), developing a memetic algorithm to minimize makespan. The local search part is called recursive arc insertion, which was first applied to the ATSP and used the 3-opt operation as its basic movement. The results obtained were compared with the insertion heuristic of Bianco, Dell'olmo and Giordani (1999) and it was shown to be superior for most instances.

The previous decade has been very active for the NWFSP with sequence-dependent setup. Araújo and Nagano (2011) continued the research with the makespan criterion. The authors proposed a new constructive heuristic and compared it to those of Bianco, Dell'olmo and Giordani (1999) and that of Brown, McGarvey and Ventura (2004). The proposed heuristic is based on a structural property of scheduling problems related to time breaks (gaps) between the beginning of two consecutive jobs on a machine, and it minimizes the makespan by minimizing the sum of the gaps on the last machine. It obtained better results than the other three existing heuristics.

Qian *et al.* (2011) applied an evolutionary concept to solve the problem with release dates to optimize the TCT for the $m$-machine problem. The results were compared to several other scheduling algorithms, among them a simulated annealing (ISHIBUCHI; MISAKI; TANAKA, 1995) and an iterated greedy heuristic (RUIZ; STÜTZLE, 2008), and they were more robust with better quality. Nagano, Silva and Lorena (2014) also studied an $m$-machine makespan minimization problem. The authors extended their previous heuristic (NAGANO; SILVA; LORENA, 2012) for the sequence-independent setup time problem. This hybrid evolutionary method explores promising solution regions by dividing the search space into clusters, which makes the application of a local search method more efficient. Computational comparison is done against the algorithms of Brown, McGarvey and Ventura (2004), Ruiz and Allahverdi (2007b) and França, Jr. and Buriol (2006), with the first two adapted to the sequence-dependent setup time case. The obtained results proved the superiority of the new method.

Samarghandi and Elmekkawy (2014) addressed the same problem with a PSO method and a new encoding system to map feasible regions and support the method in the exploration of these regions. Computational results showed that the proposed PSO is capable of finding good-quality solutions in a very short time. In addition, the proposed method was also positively compared to the hybrid GA+PSO algorithm of Samarghandi and Elmekkawy (2012), but it was not compared with other algorithms of the same problem.

Samarghandi (2015) also studied the $m$-machine problem to minimize the makespan. Additionally, a single server is responsible for performing the setup operations of multiple machines. The authors presented a GA and a GA with diversified local search procedures. Computational results using different server assignment scenarios demonstrated that, when comparing the proposed algorithms with the PSO (SAMARGHANDI; ELMEKKAWY, 2014), the server constraints have negligible effect on the makespan and the proposed algorithms improved the solutions obtained by the previous method. The main contribution of these results is that it may be possible to reduce the number of servers with minimal impact on the makespan, which could bring savings for companies that have adopted a lean approach.

Azizi, Jabbari and Kheirkhah (2016) continued studying the makespan minimization problem, but introduced the concept of truncated learning effect to develop a solution algorithm for a new problem. Setup times tend to decrease due to human experience, sequence-dependent setup times were determined by a function of the job position in a permutation and a control (or truncation) parameter, which is responsible for preventing setup times to converge to zero. Another important characteristic in a learning function environment is that each machine has a different learning factor, as they are operated by different people. Thus, besides the job sequence, learning effects were also considered in the

setup time calculation. Then, if job $j$ is in position $\pi$ after job $i$, sequence-dependent setup times are calculated as $S_{ij}^k = S_{ij}^k \times max\{\pi^{\alpha_k}, \beta_k\}$, where $\alpha_k$ is the learning effect parameter for machine $k$ and $\beta_k$ is the truncation parameter that limits the learning parameter. The authors proposed two metaheuristics to find near optimal solutions in reasonable time, a GA and a SA. The proposed methods were compared with each other and it was shown that the SA outperformed the GA.

Li *et al.* (2018) opted for minimizing the TFT. However, instead of adding release dates, they included learning and forgetting effects. Knowing that the learning effect decreases job processing times, while the forgetting effect increases them, processing times are shortened if the job is scheduled later in the sequence. These effects differ from the truncated learning effect adopted by Azizi, Jabbari and Kheirkhah (2016) as the latter could influence setup times. An IG algorithm and three accelerated neighborhood construction heuristics were presented to address the problem. The proposed IG is composed of an initial sequence construction method, which uses an accelerated backward swap instead of the traditional Nawaz-Enscore-Ham (NEH) job insertion technique (NAWAZ; ENSCORE; HAM, 1983); a local search based on the variable neighborhood descent (VND) algorithm; a neighborhood search used to improve the intensification of the proposed IG, in which a block of jobs is randomly removed and reinserted in the best position of the sequence; and a modified destruction and reconstruction procedure that improves the diversification of the searching process and balance the intensification of the local search to avoid being trapped into local optimum. In order to evaluate the proposed IG, an IG algorithm proposed for similar problems by Ruiz and Stützle (2007) and the heuristics of Bianco, Dell'olmo and Giordani (1999), Brown, McGarvey and Ventura (2004) and Nagano, Miyata and Araújo (2015) were adapted to the problem. After calibrating the necessary components of these algorithms, it was shown that the proposed IG outperformed all the other ones for all instances (using CPU time as the termination criterion) and the heuristics in terms of effectiveness.

Ying and Lin (2018) chose the makespan as the performance measure, for both sequence-dependent and independent setup times and proposed a two-phase matheuristic. In the first phase of the proposed method, after generating an initial solution by the well-known NEH heuristic, both problems under study are reduced to a special case of the ATSP (BIANCO; DELL'OLMO; GIORDANI, 1999). Afterwards, the LKH heuristic is applied to improve the constructed initial solution. In the second phase, the reduced ATSP is reformulated as a binary integer programming (BIP) model. The proposed TPM algorithm solves the relaxed BIP model until the optimum solution is found. Four benchmark sets were used in the study and an adjustment of computational times was made to compare the algorithms more fairly. It is worth mentioning that optimal solutions for very large ATPS instances (containing thousands of nodes) can be obtained efficiently despite the NP-hard nature of the problem (APPLEGATE *et al.*, 2006).

Finally, Miyata, Nagano and Gupta (2019) considered the makespan optimization with preventive maintenance, where machine unavailability is included in the scheduling process, as job processing times are interrupted during maintenance activities. Besides the development of a procedure to assign preventive maintenance operations to positions of the sequence, the authors proposed a mathematical model, which is used to solve small instances for comparison purposes, and adapted several constructive heuristics from the literature. The performance of these heuristics and the impacts of the proposed procedure to job sequencing were evaluated through computational experiments.

The only paper to optimize the mean completion time is that of Rabiee, Zandieh and Jafarian (2012) for a two-machine problem. They considered anticipatory sequence-dependent setup time with probable rework. In this case, rework may be required for each job with a known probability after undergoing an inspection. The time required for the inspection is included in the processing times. A metaheuristic is developed and its results were compared against other population-based algorithms. Results have shown that the proposed algorithm did not perform very well for large scale problems.

Nagano, Miyata and Araújo (2015) minimized the TFT with a constructive heuristic which breaks the problem in quartets, reducing the computational effort to find the solution. The method shares similarities with the algorithm proposed by Brown, McGarvey and Ventura (2004), which breaks the problem in triplets. This heuristic examines all possible four-job combinations that minimize the TFT and assigns all jobs to an optimal job sequence. In addition, neighborhood insertion and permutation search procedures are applied to improve the solution. Comparisons against the methods of Bianco, Dell'olmo and Giordani (1999) and Brown, McGarvey and Ventura (2004) showed that the proposed algorithm obtained the lowest average relative percentage deviation (ARPD) values among the constructive heuristics. The authors also proposed an IG with local search based on the method developed by Ruiz and Stützle (2007), and this algorithm obtained the best results regarding the success rate measure, when compared to the constructive heuristics.

The literature has clearly shown that the $m$-machine case is more adaptable to real cases than the two- and three-machine cases studied in the early years of the research on this problem. The highlights of this section include the work of Bianco, Dell'olmo and Giordani (1999) reducing the problem to an ATSP in order to minimize makespan and Shyu, Lin and Yin (2004) to a cumulative TSP to minimize TFT. Nagano, Miyata and Araújo (2015) considered the plain version of the problem and their constructive heuristic was superior to other constructive heuristics for the TFT optimization, but not superior to the IG also proposed by the authors based on the algorithm developed by Ruiz and Stützle (2007). Furthermore, Ying and Lin (2018) addressed the problem with the makespan and achieved impressive results for all tested instances. For the TFT criterion, Li *et al.* (2018) addressed the problem considering extra constraints of learning and forgetting effects.

Their IG had a better performance than all other algorithms. Finally, for the makespan criterion, Miyata, Nagano and Gupta (2019) added preventive maintenance to the problem.

### 2.4.3 Family Setup

Only three papers consider family setup times. Wang and Cheng (2006) and Pang (2013) studied the two-machine NWFSP with class setup times to minimize $L_{max}$. For these authors, the term class setup time is associated with family setup time and is sequence-independent. On the other hand, Behjat and Salmasi (2017) used the term group scheduling to represent family setup, which is sequence-dependent.

Wang and Cheng (2006) presented a heuristic that uses properties of forward and backward merge of batches and two local dominance rules. The heuristic was shown to be efficient and effective, but it was not compared to other methods as this was the first time that this problem was studied. Pang (2013) proposed a GA-based heuristic, a GA with local improvement operators, for the same problem and compared the results with those obtained by Wang and Cheng (2006) and by the earliest due date (EDD) priority rule. Computational results showed that the proposed algorithm outperformed the other two heuristics in all tested cases.

Behjat and Salmasi (2017) approached a NWFSP with sequence-dependent family setup times in order to minimize TCT. The authors proposed several metaheuristics based on PSO and VNS algorithms, besides a heuristic to generate feasible initial solutions, and performed a performance evaluation. The proposed algorithms were compared to existing ones proposed for a similar scheduling problem (a hybrid metaheuristic based on a GA and a SA, and a VNS algorithm), but a clear comparison is not possible as there are no other studies on this variant.

## 2.5 Mixed No-Wait Flow Shop Scheduling Problem

A recent variant of the NWFSP considers mixed operations, some of them classified as traditional permutation flow shop and others as no-wait flow shop. Indeed, in some practical applications, the no-wait constraint cannot describe the whole processing line, because there may be only some operations with this requirement. Therefore, a new term called mixed no-wait was created to describe some manufacturing processes formed by a mix of no-wait and "allowed wait" operations. An example of this environment is found in the food industry, in which some groups of machines are classified as no-wait while the rest is considered as traditional permutation flow shop machines.

Wang *et al.* (2018) were the first to study this problem. They described the canned food processing industry as a mixed no-wait flow shop environment and exemplified which processes compose the no-wait machine group. They showed that, the mixed no-wait flow shop scheduling problem (MNWFSP) is NP-hard for more than two machines and can be

Table 2 – NWFSP with Non-Family Sequence-Dependent Setup Time

| **Author** | **Problem** | **Method** | **Comparison** |
|---|---|---|---|
| Bianco, Dell'olmo and Giordani (1999) | $F_m\|no-wait, r_i, s_{ijk}\|C_{max}$ | BAH and BIH | Between them |
| Allahverdi and Aldowaisan (2001) | $F_2\|no-wait, s_{ijk}\|\sum C_j$ | Heuristic algorithms | n/a |
| Shyu, Lin and Yin (2004) | $F_2\|no-wait, s_{ijk}\|\sum C_j$ | ACO algorithm | Aldowaisan and Allahverdi (1998), Aldowaisan (2001) and 3-opt based algorithm |
| Lee and Jung (2005) | $F_m\|no-wait, prec, s_{ijk}\|C_{max}$ | SA-based metaheuristic, heuristic and hybrid algorithm | Between them |
| França, Jr. and Buriol (2006) | $F_m\|no-wait, r_i, s_{ijk}\|C_{max}$ | Memetic Algorithm (MA) | Bianco, Dell'olmo and Giordani (1999) |
| Araújo and Nagano (2011) | $F_m\|no-wait, s_{ijk}\|C_{max}$ | GAPH | Bianco, Dell'olmo and Giordani (1999) and Brown, McGarvey and Ventura (2004) |
| Qian *et al.* (2011) | $F_m\|no-wait, r_i, s_{ijk}\|\sum C_j$ | HDE (differential evolution) | Ruiz and Stützle (2008), Ishibuchi, Misaki and Tanaka (1995), etc. |
| Rabiee, Zandieh and Jafarian (2012) | $F_2\|no\text{-}wait, s\{c_j; d_j\}\|\sum C_j/n$ with probable rework | AICA | GA and population-based SA |
| Nagano, Silva and Lorena (2014) | $F_m\|no-wait, s_{ijk}\|C_{max}$ | $ECS\_NSL$ Extension | Brown, McGarvey and Ventura (2004), Ruiz and Allahverdi (2007b) and França, Jr. and Buriol (2006) |
| Samarghandi and Elmekkawy (2014) | $F_m\|no-wait, s_{ijk}\|C_{max}$ | PSO | Samarghandi and Elmekkawy (2012) |
| Nagano, Miyata and Araújo (2015) | $F_m\|no-wait, s_{ijk}\|\sum C_j$ | QUARTS and IG | Bianco, Dell'olmo and Giordani (1999), Brown, McGarvey and Ventura (2004) and Ruiz and Stützle (2007) |
| Samarghandi (2015) | $F_m, S1\|no-wait, s_{ijk}\|C_{max}$ | GA and GA with local search | Samarghandi and Elmekkawy (2014) |
| Azizi, Jabbari and Kheirkhah (2016) | $F_m\|no\text{-}wait, s_{ijk}\|C_{max}$ with truncated learning effect | Between them | |
| Li *et al.* (2018) | $F_m\|no\text{-}wait, s_{ijk}\|\sum C_j$ with learning and forgetting effects | IG | Ruiz and Stützle (2007), Bianco, Dell'olmo and Giordani (1999), Brown, McGarvey and Ventura (2004) and Nagano, Miyata and Araújo (2015) |
| Ying and Lin (2018) | $F_m\|no\text{-}wait, s_{ij}\|C_{max}$ and $F_m\|no\text{-}wait, s_{ijk}\|C_{max}$ | TPM | Ruiz and Stützle (2008), Araújo and Nagano (2011), Samarghandi and Elmekkawy (2012), Samarghandi and Elmekkawy (2014) and Nagano, Silva and Lorena (2014) |
| Miyata, Nagano and Gupta (2019) | $F_m\|no\text{-}wait, s_{ijk}, PM\|C_{max}$ | Constructive heuristics | Between them |

Source: Elaborated by the author.

Note: n/a means that a comparison was not made.

denoted as $F_m|mixed, no\text{-}wait|C_{max}$, considering the makespan criterion. An IG algorithm was presented to solve the problem and a speed-up makespan calculation method was proposed to solve it more efficiently. Its initialization is based on the $FRB4_k$ method, a trade-off between the NEH and the Dipak heuristic (procedure developed by Laha and Sarin (2009)), combined with the proposed speed-up calculation method. After a job from the seed is inserted into the best position $x$ by the NEH, the $k$ jobs before and after $x$, totaling $2k$ jobs, are adjusted by reinserting each job into all possible backward slots. The initialization is followed by a modified destruction and reconstruction, which uses the $FRB4_k$ method to reinsert removed jobs and improve the diversification and intensification of the search process. The local search method used to enhance the intensification of the previous phase, is a VND algorithm. These three phases are iteratively performed until a termination criterion is met. The proposed method compared with algorithms developed for the PFSP and the NWFSP, was shown to have a superior performance.

Cheng *et al.* (2019) extended the work of Wang *et al.* (2018) by adding sequence-dependent setup times to the problem ($F_m|mixed, no\text{-}wait, s_{ijk}|C_{max}$), recognizing the importance of considering setups for real industrial environments. They also proposed a similar algorithm because of its successful applications for solving different scheduling problems. The proposed algorithm uses a modified NEH for initial solution generation, a pairwise-destruction and a pairwise-construction phases, and a local search procedure with a speed-up mechanism. The proposed algorithm was compared to the existing one and obtained better solutions, mainly because of the new pairwise mechanism, which can increase the perturbation of the solutions to escape local optima, while the speed-up mechanism allows a quick convergence towards the global optimum.

## 2.6 Instance analysis

Regarding the size of the instances, we can see from Table 3 that, typically the benchmark size increases over time with the development of more efficient solution methods. Furthermore, an instance set may vary depending on the constraints being considered in the study. Thus, studies that solve a problem with the same type of constraints tend to use the same instance set. For example, the well-know Taillard instances, introduced by Taillard (1993) for the FSP, were adapted and extended for other problems. In addition, all reviewed instances were generated randomly, showing that there is a lack of studies based on real applications.

While early works considered very small instances with only 2 machines and as few as 6 jobs, more recent works have considered up to 2000 jobs, up to 40 machines, and very large instance sets containing almost 10,000 instances. Typical values recently are in the range of 200–500 jobs and 10–20 machines. Problems with family setup times are significantly less studied (see Section 2.4.3); as also reflected by the size of its benchmarks:

Table 3 – Instance sizes and characteristics

| | **Jobs** | **Machines** | **# Instances** |
|---|---|---|---|
| *Sequence Independent Setup Time* | | | |
| Aldowaisan and Allahverdi (1998) | 10 | 2 | 30 |
| Sidney, Potts and Sriskandarajah (2000) | 6 | 2 | 2 |
| Aldowaisan (2001) | 40 | 2 | 84 |
| Aldowaisan and Allahverdi (2004) | 120 | 3 | 88 |
| Brown, McGarvey and Ventura (2004) | 20 | 10 | 80 |
| Fondrevelle, Allahverdi and Oulamara (2005) | 18 | 2 | 90 |
| Ruiz and Allahverdi (2007b) | 200 | 40 | 5400 |
| Ruiz and Allahverdi (2007a) | 100 | 20 | 1200 |
| Su and Lee (2008) | 1000 | 2 | 9300 |
| Samarghandi and ElMekkawy (2011) | 1000 | 2 | 1440 |
| Samarghandi and Elmekkawy (2012) | 75 | 20 | 79 |
| Nagano, Silva and Lorena (2012) | 200 | 40 | 8400 |
| *Sequence Dependent Setup Time* | | | |
| Bianco, Dell'olmo and Giordani (1999) | 700 | 10 | 330 |
| Allahverdi and Aldowaisan (2001) | 100 | 2 | 360 |
| Shyu, Lin and Yin (2004) | 250 | 2 | 270 |
| Lee and Jung (2005) | 90 | 9 | 60 |
| França, Jr. and Buriol (2006) | 100 | 10 | 438 |
| Araújo and Nagano (2011) | 200 | 20 | 440* |
| Qian *et al.* (2011) | 100 | 40 | 84 |
| Rabiee, Zandieh and Jafarian (2012) | 200 | 2 | 24 |
| Nagano, Silva and Lorena (2014) | 500 | 20 | 480* |
| Samarghandi and Elmekkawy (2014) | 75 | 20 | 29 |
| Nagano, Miyata and Araújo (2015) | 200 | 20 | 440* |
| Samarghandi (2015) | 75 | 20 | 29 |
| Azizi, Jabbari and Kheirkhah (2016) | 45 | 12 | 24 |
| Li *et al.* (2018) | 500 | 20 | 480* |
| Ying and Lin (2018) | 2000 | 20 | 658* |
| Miyata, Nagano and Gupta (2019) | 200 | 20 | 736 |
| *Family Setup Time* | | | |
| Wang and Cheng (2006) | 30 | 2 | 64 |
| Pang (2013) | 30 | 2 | 32 |
| Behjat and Salmasi (2017) | 10 | 6 | 270 |
| *Mixed No-Wait Flow Shop* | | | |
| Wang *et al.* (2018) | 500 | 20 | 840 |
| Cheng *et al.* (2019) | 500 | 20 | 520 |

Source: Elaborated by the author.

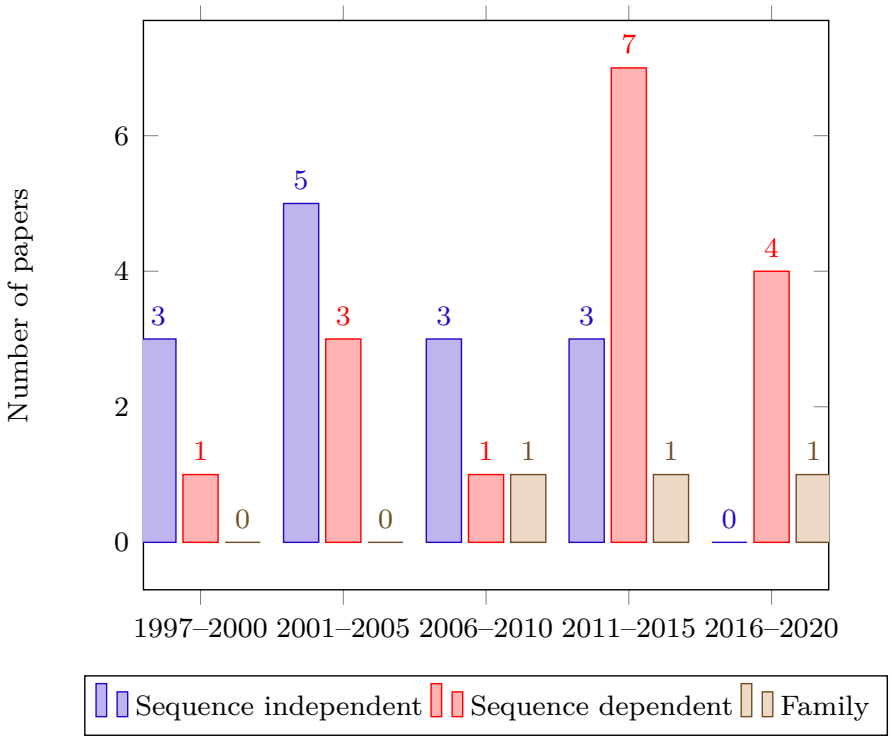
Note: * based on the Taillard (1993) instances for the FSP.

at most 30 jobs and 6 machines.

## 2.7 Conclusions and directions for further research

From the classification of the studies by type of setup, it is possible to conclude that larger effort is being put to develop sequence-dependent type of research lately, as opposed to the simpler sequence-independent that dominated prior to 2010. Figure 1 shows the number of studies published per type of setup since 1997. As we can see, in the last decade sequence-dependent research represents the vast majority. Sequence-independent problems dominated earlier possibly due to their lower complexity than when there are dependency among jobs. Another possible reason for the current tendency may be explained by the greater applicability of sequence-dependent setup times in real cases.

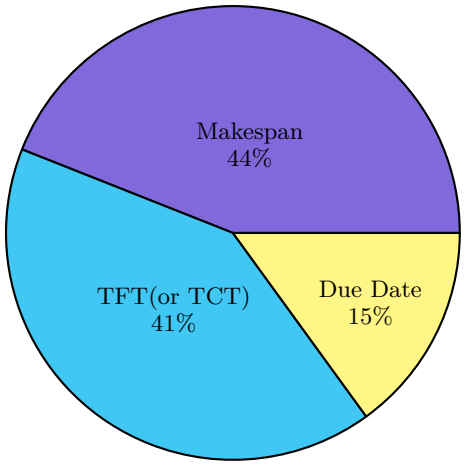
Figure 1 – Papers per year and type of setup




Papers per year and type of setup.

In addition, $m$-machine problems represent the majority of papers among sequence-dependent setup time studies. The reason again is that improving the robustness of solution methods over time allowed solving more complex problems. In relation to the performance measures, the makespan and the TFT (or TCT) were the most used criteria, followed by the due date (see Figure 2).

Figure 2 – Percentage of papers per type of performance measure




With this review, it is also clear that the NWFSP has often been compared to the well-known TSP. This is an interesting characteristic as the methods initially developed for transportation problems, such as the LKH, 2-opt and 3-opt algorithms, and very efficient exact algorithms can also be used to solve scheduling problems.

It is also possible to observe the evolution from constructive heuristics to meta-heuristics with improvement procedures in order to develop more robust and efficient methods capable of solving a highly complex problem even for large instances. We can also notice that the most efficient methods presented have some common features, for example: a constructive heuristic to generate a good initial solution; local and neighborhood search procedures to increase the intensification of the search, among them insertion and swap procedures; destruct and reconstruct operators to increase the diversification of the solutions and avoid local minima; and speed-up methods to decrease the time spent evaluating partial solutions being generated. Therefore, a right balance between exploration and exploitation is the key factor for a method to achieve a good performance.

Furthermore, there is a connection between papers if one compares a new algorithm with an existing method and shows an improvement in the solution, which means that the latest papers present the best methods for a specific combination of problem and performance measure. However, in many cases this comparison did not include the most recent methods for the same problem, and the field can largely benefit from a reassessment of competing algorithms on the same benchmark sets.

For future research, we identify the following avenues and gaps. Studies on family setup, sequence-independent setup time and due date criteria are still the minority amongst the studies. Moreover, constraints introduced more recently, such as learning and forgetting effects and preventive maintenance, and mixed environments can also be further investigated. Hence, we suggest exploring these types of setup, objective function

and constraint for the no-wait scheduling problem. Furthermore, studies on the analysis of the lower and upper bounds are scarce as most of the literature has focused on heuristic developments. In this way, there is potential to develop efficient lower bounds and also exact algorithms to obtain optimal solutions for the problem. A method that has been applied and brought good results for this purpose is reducing the problem to the TSP, which may be a potential technique to be considered with metaheuristic applications. In addition, studies that describe real applications of optimization algorithms are still rare in this line of study. Additionally, we propose developing efficient algorithms to optimize real problem variants for different problem sizes with limited computational time.

# 3 AN ALNS ALGORITHM APPLICATION FOR THE NWFSP

## 3.1 Introduction

Scheduling in the manufacturing industry has an important role in optimizing the use of resources and the delivery of products by allocating jobs to machines in a sequence that minimizes a specific criterion. When machines in series are the object of study and jobs flow through all of them in the same order, a permutation flow shop scheduling problem (PFSP) is characterized (PINEDO, 2008).

For this type of scheduling problem, many constraints may be considered to model real cases, such as setup and waiting time constraints along the process (PAN; RUIZ, 2013). The setup time required to prepare the resource for the next job is not a value-added factor. Thus, its consideration in scheduling decisions can eliminate waste and improve machinery utilization, leading to an increase in productivity and in meeting order deadlines. In addition, because setup times are present in a multitude of industrial applications, ignoring them could negatively affect the applicability of the solution. Setup times can be classified as dependent or independent of the job sequence. A sequence-dependent setup depends on the previous job and can be denoted as $s_{ijk}$, the time span required to configure machine $i$ when job $k$ is scheduled immediately after job $j$, whereas sequence-independent setup times are constant for a given job (ALLAHVERDI, 2015).

On the other hand, the no-wait constraint in flow shop scheduling occurs when jobs cannot wait on or between two consecutive machines and are necessarily processed in the same order over all of them. According to Allahverdi (2016), technology requirements are the main reason for the occurrence of the no-wait constraint. For example, the characteristics of raw material and manufacturing processes could require that two or more operations be performed without waiting between them. Such as in the steel industry for high-temperature steel, where waiting could lead to a waste of energy to reheat the material. Other reasons for the no-wait requirement are the lack of storage space between intermediate operations and the need to reduce work-in-process (WIP). Additional examples of industries that can be modelled as a no-wait scheduling problem are the plastic, chemical, and pharmaceutical ones, besides manufacturing processes with highly coordinated robotic cells.

Beyond the constraints that can be added to a model to better characterize the scheduling problem, a criterion that translates the organization's strategy into a performance measure must be selected to be optimized (MACCARTHY; LIU, 1993), . Baker and Trietsch (2009) and Pinedo (2008) described a set of performance measures employed to evaluate schedules, which are often a function of the job completion time $C_j$. Among these relevant measures is the minimization of total flow time (TFT), which

represents the sum of each of the job flow times ($TFT = \sum F_j$, $j = 1, ..., n$), the time a job spends in the system since its release until its completion ($TFT = \sum C_j$, $j = 1, ..., n$, if all release dates are the same). The TFT measure is commonly adopted since it reflects the average flow time of all orders in the system. It is known to increase production rate and decrease the WIP inventory. Thus, it can be applied to meet the demand of individual products as soon as it is completed and to provide a rapid response to demands (RUIZ; ALLAHVERDI, 2007a).

The choice of performance measure can also be used in order to obtain good solutions for a given work environment. Maassen, Perez-Gonzalez and Günther (2020) showed that the TFT is the criterion that most relates to a no-wait environment. Because waiting time interrupts the continuous job flow, the waiting time within the system is highly related to a measure of job flow. Thus, the lowest TFT can be achieved with the lowest waiting time among operations.

It should also be noted that most of the solution methods for the no-wait flow shop scheduling problem (NWFSP) with sequence-dependent setup times (SDST) in the literature are heuristic algorithms, which is due to the fact that, when considering both constraints, the problem becomes NP-hard (RUIZ; ALLAHVERDI, 2007a). Thus, we propose a new solution method to minimize the TFT and compare the results with the state-of-the-art for the NWFSP with SDST.

The remainder of this study is organized as follows. In Section 3.2 a review of the literature is presented. In Section 3.3 we describe the problem. Section 3.4 presents the mathematical formulation of the problem, followed by the proposed algorithm in Section 3.5. We present the results of extensive computational experiments in Section 3.7, followed by conclusions in Section 3.8.

## 3.2 Literature review

The NWFSP with setup time requirements has been extensively studied in the literature since 1999. Comprehensive reviews of the flow shop scheduling problem were published by Allahverdi (2016) on the no-wait version of the problem, and by Allahverdi (2015) on the problem with setup time. In addition, Nagano and Miyata (2016) reviewed the many constructive heuristics for the NWFSP. The literature review in Chapter 2 compares the existing solution methods with both constraints.

Two general remarks from the literature review indicate that although the TFT criterion align well with the no-wait constraint of the production environment, most papers still use a makespan minimization objective instead of the TFT.

According to the aforementioned reviews, Qian *et al.* (2011) and Qian *et al.* (2012) were the first authors to address the problem under analysis, with the inclusion of

release dates as constraints. Their heuristics included a global search based on differential evolution, which was used to find promising solution regions, a problem-dependent local search procedure and a speed-up evaluation method. Qian *et al.* (2012) added an extra speed-up method to differentiate its algorithm from Qian *et al.* (2011). The authors compared their algorithms with an Iterated Greedy (IG) (RUIZ; STÜTZLE, 2008) and a Simulated Annealing (SA) (ISHIBUCHI; MISAKI; TANAKA, 1995) algorithms and showed that their methods named HDE (QIAN *et al.*, 2011) and $DE\_TSM$ (QIAN *et al.*, 2012) obtained better results. When the comparison occurred between them, $DE\_TSM$ was superior.

Later, Zhang *et al.* (2016) also addressed the problem including release dates. The authors proposed an algorithm that adopts a probabilistic model to execute a global search, which could find superior solutions and guide the search towards promising regions. The method relates processing priority to the probability of a job being in a determined position of the sequence and has an insert-based neighborhood local search, besides a speed-up evaluation method. In addition, offspring individuals are generated from the probabilistic model and a new round of local search is applied iteratively, until the maximum number of generations is reached. Their algorithm was compared to several methods, including two IG (RUIZ; STÜTZLE, 2008; DING *et al.*, 2015) and an SA (ALLAHVERDI; AYDILEK, 2014), showing to be superior in terms of solution quality and robustness. Nonetheless, their results have not been compared to those of Qian *et al.* (2012).

The only study in the literature that solved the NWFSP with SDST while minimizing the TFT was proposed by Nagano, Miyata and Araújo (2015). The authors designed a constructive heuristic named QUARTS, which breaks the problem in quartets, reducing the computational effort to find a solution, and examines all possible four-job combinations to assign jobs to a sequence. In addition, neighborhood insertion and permutation search procedures are applied to improve the solution. The results obtained were compared to other constructive heuristics, such as those of Bianco, Dell'olmo and Giordani (1999) and Brown, McGarvey and Ventura (2004). QUARTS presented the best performance in comparison with other constructive heuristics with acceptable computational times. The authors also proposed an IG with local search adapted from Ruiz and Stützle (2007) for the problem, obtaining the best results.

Li *et al.* (2018) were the last authors to consider the problem under study for TFT and the first ones to add learning and forgetting effects. These additional constraints may affect processing times, because they are shortened if the job is scheduled later in the sequence. Thus, an IG that deals with changes in processing times was presented to address the problem. The algorithm is composed of an initial sequence construction method, which uses an accelerated backward swap; a local search based on the variable neighborhood descent (VND); a neighborhood search used to improve the intensification of the search,

in which a block of jobs is randomly removed and reinserted in the best position of the sequence; and a modified destruction and reconstruction procedures that improve the diversification of the searching process. Since there were no algorithms for this version of the problem, some existing methods were adapted and used as a comparison, such as two IG algorithms (XU; ZHU; LI, 2012; RUIZ; STÜTZLE, 2007) and the constructive heuristics of Bianco, Dell'olmo and Giordani (1999), Brown, McGarvey and Ventura (2004) and Nagano, Miyata and Araújo (2015). It was shown that the proposed method outperformed all the compared ones.

Analysing the methods presented by these studies, it is clear that procedures such as local and neighborhood search and speed-up calculation are present in every algorithm proposed for the problem, either with or without extra constraints. The first procedure is known for increasing the intensification of the search, such as the swapping and insertion of jobs and blocks of jobs, while the latter is an acceleration method applied to calculate the objective function, especially for partial solutions, which is more effective and less time consuming. Besides that, destruction and reconstruction procedures, which are designed to balance the intensification of the local search and avoid being trapped into local optima by increasing the diversification, are also often used in efficient heuristics. Finally, an initial solution generator based on an heuristic that was proposed to optimize the TFT can be an advantage in building a method to minimize this criterion.

When looking at the results, the IG algorithm structure, which aggregates all procedures mentioned above, achieves a prominent position. In addition, as it has been shown through the literature review, Nagano, Miyata and Araújo (2015) was the only one to address the exact same problem being solved in this research, and their method will be used to evaluate the proposed algorithm.

In order to find a method as efficient as the IG, we propose a solution algorithm with similar structure but with more engineered components. Our proposed method is an improvement heuristic based on the Adaptive Large Neighborhood Search (ALNS) framework proposed firstly for the Vehicle Routing Problem (VRP) by Ropke and Pisinger (2006). The ALNS heuristic has been largely used for distribution problems, but it has a flexible structure that can be adapted for a scheduling problem. Its main advantage when compared to the well known IG framework is a rewarding mechanism used for rating destruction and repair operators according to their performance. This mechanism allows the heuristic to select the best destruction and reconstruction operators for each type of instance, which makes the algorithm more robust.

In the literature there are few studies applying the ALNS for scheduling problems. We can cite the parallel machine scheduling problem (BEEZãO *et al.*, 2017; COTA *et al.*, 2017; COTA *et al.*, 2019), the distributed flow shop scheduling problem (RIFAI; NGUYEN; DAWAL, 2016), and the job shop scheduling problem (CHELLADURAI;

AZATH; JENIFFER, 2020). However, there is no application of this method for the NWFSP.

## 3.3 Problem description

The problem under study can be described as follows. Let a set of $n$ jobs $J = \{J_j | j = 1, 2, ..., n\}$ be available at time zero to be processed on a set of $m$ machines $M = \{M_i | i = 1, 2, ..., m\}$ sequentially. Each machine processes all jobs in the same order and one job at a time. The operation $O_{ji}$ of job $J_j$ on machine $M_i$ is processed without interruption and its processing time $p_{ji}$ is positive and deterministic. Machines are classified as no-wait machines, thus operations cannot wait on or between machines to be processed. The setup time $s_{ilj}$ for job $j$ on machine $i$ depends on the preceding job $l$ in the sequence $\pi$ of jobs ($j, l = 1, ..., n, i = 1, ..., m$ for $j \neq l$). Setup operations can be performed independently of the job and, because of the no-wait restriction, before the job arrives at the machine. The objective of the solution method is to find the best sequence for $n$ jobs that minimizes the total flow time $TFT = \sum C_j, j = 1, ..., n$, where $C_j$ is the completion time of job $j$.

When setup times are considered separately from processing times, a job setup on a subsequent machine may be performed while the job is still in process on the immediately preceding machine, which improves machine utilization and may reduce total completion time (ALDOWAISAN; ALLAHVERDI, 1998). However, it is important to note that when both characteristics, no-wait and setup, are considered in a scheduling problem, setup must be executed on a machine before the job arrives to satisfy the no-wait constraint. Thus, setup is performed on a specific machine while the job is still being processed on a previous one (RUIZ; ALLAHVERDI, 2007a). In addition, to ensure that jobs are processed continuously through the $m$ machines without interruption, it might be necessary to delay the starting time of a job on the first machine, creating idle time on some machines (PINEDO, 2008).

## 3.4 Mathematical formulation

The first two variables of the model are composed by the indices that indicate a job ($j$ and $l$) and a position ($k$) in the permutation $\pi$.

$$x_{jk} = \begin{cases} 1, & \text{if } J_j \text{ is in position } k \text{ of } \pi \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$y_{lkj} = \begin{cases} 1, & \text{if } J_j \text{ is in position } k \text{ and is preceded by } J_l \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The third variable $c_{ki}$ represents the completion time of the job in position $k$ on machine $i$. This variable composes the objective function to be minimized on the last machine.

$$\text{minimize } TFT = \sum_{k=1}^{n} C_{km} \tag{3.3}$$

subject to

$$\sum_{k=1}^{n} x_{jk} = 1, \quad \forall j \in 1, ..., n \tag{3.4}$$

$$\sum_{j=1}^{n} x_{jk} = 1, \quad \forall k \in 1, ..., n \tag{3.5}$$

$$x_{jk} = \sum_{l=1}^{n} y_{lkj}, \quad \forall j \in 1, ..., n, \forall k \in 2, ..., n \tag{3.6}$$

$$x_{jk} = \sum_{l=1}^{n} y_{jk+1l}, \quad \forall j \in 1, ..., n, \forall k \in 1, ..., n-1 \tag{3.7}$$

$$y_{l1j} = 0, \quad \forall l, j \in 1, ..., n \tag{3.8}$$

$$\sum_{l=1}^{n} \sum_{j=1}^{n} y_{lkj} = 1, \quad j \neq l, \forall k \in 2, ..., n \tag{3.9}$$

$$y_{lkj} + y_{jkl} \leq 1, \quad \forall j, l \in 1, ..., n, \forall k \in 2, ..., n \tag{3.10}$$

$$c_{k1} \geq \sum_{j=1}^{n} x_{jk} p_{1j}, \quad \forall k \in 1, ..., n \tag{3.11}$$

$$c_{ki} = c_{ki-1} + \sum_{j=1}^{n} x_{jk} p_{ij}, \quad \forall k \in 1, ..., n, \forall i \in 2, ..., m \tag{3.12}$$

$$\begin{gathered} c_{ki} \geq c_{ki-1} + \sum_{j=1}^{n} x_{jk} p_{ij} + \sum_{j=1}^{n} \sum_{l=1}^{n} y_{lkj} s_{ilj}, \\ \forall k \in 2, ..., n, \forall i \in 1, ..., m, \forall l \in 1, ..., k-1 \end{gathered} \tag{3.13}$$

$$c_{ki} \geq 0, \quad \forall i \in 1, ..., m, \forall k \in 1, ..., n \tag{3.14}$$

$$x_{jk} \in \{0, 1\} \quad \forall j, k \in 1, ..., n \tag{3.15}$$

$$y_{lkj} \in \{0, 1\} \quad \forall l, k, j \in 1, ..., n \tag{3.16}$$

The objective function (3.3) minimizes the TFT, the sum of all job completion times on the last machine. Constraints (3.4) and (3.5) impose that each job occurs only once in a sequence $\pi$, that is, each job is assigned to only one position and vice-versa. Constraints (3.6) and (3.7) ensure that each job precedes only one job and follows exactly one job. Constraints (3.8) indicate that there is no precedence for the job in the first position. Constraints (3.9) and (3.10) state that, from the second position, there is only one job $j$ in position $k$ preceded by job $l$. Constraints (3.11) and (3.12) define that a job can only start after its previous operation finishes. In particular, constraints (3.11) refer to the completion time at the first machine, and (3.12) to the no-wait requirement. Constraints (3.13) provide a relation between completion time and setup time and show that a job can only start after the previous job on the same machine finishes. Constraints (3.14) - (3.16) define the variables of the model.

## 3.5 The proposed algorithm

In general terms, the proposed heuristic is based on the ALNS and consists of a number of destruction and repair operators, an acceptance criterion based on the simulated annealing concept, a swapping procedure as local search and a rewarding mechanism. While destruction and reconstruction improves the diversification, local search enhances the intensification of the algorithm. The pseudocode structure is described in Algorithm 1.

Given an initial solution $\pi_{initial}$;
$\pi_{best} = \pi = \pi_{initial}$;
**while** *stopping criteria not satisfied* **do**
    selection of destruction and repair operators;
    operators applied on $\pi$ to generate $\pi_{actual}$;
    **if** $TFT(\pi_{actual}) < TFT(\pi)$ **then**
        local best is updated: $\pi = \pi_{actual}$;
        chosen operators are rewarded;
        **if** $TFT(\pi_{actual}) < TFT(\pi_{best})$ **then**
            global best is updated: $\pi_{best} = \pi_{actual}$;
            chosen operators are rewarded with largest reward;
        **end**
    **else**
        worst solutions are accepted based on a simulated annealing operation;
        local best is updated: $\pi = \pi_{actual}$;
        chosen operators are rewarded with smallest reward;
    **end**
    **if** *segment is completed* **then**
        new operator's probabilities calculation;
    **end**
    temperature is reduced;
    **if** *temperature reaches minimum temperature* **then**
        reheating;
    **end**
**end**
**return** *TFT*

**Algoritmo 1:** ALNS for the NWFSP based on Ropke and Pisinger (2006)

### 3.5.1 Initial solution

The NEH heuristic developed by Nawaz, Enscore and Ham (1983) is among the most used heuristics for permutation flow shop scheduling problem. However, this heuristic was first develop for makespan minimization, as it orders jobs in a non-increasing order of total sum of processing times. Thus, when minimizing the TFT one can improve the performance of this heuristic by ranking jobs by increasing order of total sum of processing times (PAN; RUIZ, 2013). This adaptation is employed to generate the first solution, which will be iteratively destructed and reconstructed to improve the minimum TFT.

### 3.5.2 Destruction and reconstruction operators

In the ALNS algorithm, jobs are removed from the permutation $\pi$, generating a partial sequence, so these removed jobs can be placed in a position that decreases the objective function. This process occurs iteratively until a stopping criteria is reached.

Five destruction operators are proposed in this research, which are detailed as follows. The first one removes a random job from the sequence, while in the second one a random number of jobs in random positions are removed from the sequence; both options bring some variability to the method. In the third operator, five pairs of consecutive jobs with the largest sum of setup times between them on all machines are selected among all jobs and one pair is randomly removed from the sequence. In the fourth, five pairs of consecutive jobs with the largest sum of idle times on the first and last machines are selected and one pair is randomly removed from the sequence. Setup times are considered on all machines between jobs because the larger the setup time the larger the total flow time. On the other hand, we look at idle times for the first and last machines because of the nature of the no-wait constraint, for which idle times occur on the first machine to guarantee that there is no wait between operations of a job through all machines, and in which the last machine presents the second most representative idle time among all machines besides the first, caused by accumulation on previous machines. Finally, in the last destruction operator, jobs that, when removed from the sequence lead to the minimum objective function value, are removed from the sequence.

To repair the destructed jobs and create the actual sequence, six re-constructors were developed. The first one reinserts the jobs once removed into the sequence in random positions. The second one reinserts them in positions that lead to the minimum sum of setup times for all jobs of the sequence on all machines. In the third reconstruction operator, removed jobs are reinserted into positions that lead to the minimal sum of idle times for all jobs of the sequence on the first and last machines. As the TFT decreases when setup and idle times are minimized, operators that aim to reduce both sums can lead to the optimization of the solution. Operators four and five are greedy versions of the second and third repair operators, respectively. Finally, in the last operator, removed jobs are reinserted in positions that lead to the minimum TFT. All operators were designed aiming to minimize the objective function.

### 3.5.3 Reward mechanism

Destruction and repair operators start with the same probability of being chosen, but as iterations are completed, the weight of each operator is updated based on their performances. When a pair of operators is applied and generates a better solution, both operators receive a medium reward that will count into the next weight calculation. If a better global solution is found the reward is higher. In addition, in order to maintain

solution variability, if a worse solution is accepted by the acceptance criteria, a low reward is attributed to the destruction and repair operators used in that iteration. In this way, operators that received more rewards will have a greater weight and will be more likely to be chosen during next iterations. This component of the proposed method is one of the mechanisms that differentiates ALNS from the IG structure. It gives autonomy to the algorithm to choose which destruction and repair operators offer the best performance for each instance.

The scores are summed for a given number of iterations, called a segment. At the end of each segment the weights are calculated based on the recorded scores. Following the equations proposed by Ropke and Pisinger (2006), we calculate the weigh of each destruction and repair operator according to the equation (3.17).

$$w_{j+1} = w_j(1 - r) + r\frac{\pi_i}{\theta_i} \tag{3.17}$$

Where $\pi_i$ is the score of operator $i$ obtained during the last segment, and $\theta_i$ is the number of attempts to use operator $i$ during the last segment. The reaction factor $r$ can be set from 0 to 1, zero meaning that the scores are not used at all (initial weights are maintained during every segment) and one that the score obtained in the last segment controls the weight. Initial weights $w_j$ for segment $j = 0$ are defined equally among destruction ($n_{dest}$) and repair ($n_{rep}$) operators.

Knowing that there are $k$ operators with weights $w_j$ and $i \in 1, 2, ..., k$, to select which operator to use at the end of each segment, we update their probabilities by normalizing them. Notice that the destruction operator selection is independent of the reconstruction operator and the other way around.

### 3.5.4 Acceptance criteria and heating operator

When a new solution is generated, its objective function is compared with the current objective value and the best one found so far. The acceptance criteria from simulated annealing updates the local and global best solutions and also accepts worse solutions, with a probability that depends on a heating operator, in order to increase diversity into the search. Equation (3.18) shows how the probability of accepting solution $s^{'}$ found in the current iteration given the objective value $f(s)$ of the current solution $s$:

$$e^{-\frac{f(s^{'}) - f(s)}{T}}. \tag{3.18}$$

At the beginning of the heuristic, a temperature $T > 0$ is set. This temperature is decreased slowly over the iterations by a cooling rate $0 < c < 1$ (as $T = T * c$). The smaller the temperature gets, the smaller the chance of accepting a bad solution. Consequently, bad

solutions are accepted during the first iterations, while the solution is still not consolidated, allowing a larger solution space to be explored. Thus, the probability of accepting them is smaller after several iterations.

However, as the number of worse accepted solutions decreases considerably over time, a reheating processes is applied. When the temperature set at the beginning of the heuristic $T_{start}$ gets to a very low, it is changed for a high value and the probability of accepting a bad solution is increased again. This mechanism can be repeated many times and helps the method to jump from a local minimum to a new region that could lead to a global minimum.

In this study, it was defined through empirical results $T_{start} = 5000$ and a cooling rate of $c = 0.2$. In the case of the number of iterations be bigger than 5000, this rate drops to $c = 0.989$, allowing the temperature to decrease slower and restricting the chances of accepting a bad solution before the temperature may be increased again. The reheating process occurs when the temperature reaches $T \leq 0.01$, leading to a new temperature value of $T = 10000$.

### 3.5.5 Swapping and stopping criteria

In order to intensify the search, we apply a swapping procedure at some stages of the algorithm. We have developed two types of swapping operators. In the first one, two or three elements of the sequence have their positions swapped, if a smaller objective function is found then the actual sequence and its calculated objective function are updated. This operation is repeated until there is no improvement. Weather three elements are being swapped it must be considered all possible permutations among them. We can also consider a combination of two and three elements by applying the two-swap until there is no improvement and them apply the three-swap, if there is improvement apply two-swap again, if not stop the local search. Below is a pseudocode 2 showing the swap of two elements.

**while** $improvement = true$ **do**
    $improvement = false$;
    **for** *jobs* $j \geq 0, j \leq \pi_{actual}.size()$ *and* $k = j + 1, k \leq \pi_{actual}.size()$ **do**
        $\pi_{aux} = \pi_{actual}$;
        jobs $j$ and $k$ are swapped;
        **if** $TFT(\pi_{aux}) < TFT(\pi_{actual})$ **then**
            local best is updated: $\pi_{actual} = \pi_{aux}$;
            $improvement = true$;
        **end**
    **end**
**end**
**return** $\pi_{actual}$ *and* $TFT(\pi_{actual})$

**Algoritmo 2:** Swap of two elements

The second swapping procedure is a block swap, in which two, three or four elements change their position to search for a smaller TFT. All permutations of the elements inside the block are tested. We use two variations of this procedure, one that selects the first adjacent elements of the sequence to form the block and another that selects all combinations of two elements in positions $j, 0 \leq j \leq \pi.size()$ and $k = j+1, k \leq \pi.size()$ and forms a block using them to run through all positions of the sequence. Both operators iterate until there is no improvement.

Firstly, block swap operators are placed after the reconstruction operator, when the actual sequence is formed. Swap of two elements is applied followed by a block swap. To choose the length of the block, a random number from zero to one is generated and the selection is made as follows. The percentages were determined empirically.

Selection of destruction and repair operators;
Swap of two elements;
Block swap
**if** $random \leq 0.35$) **then**
  two element block swap;
**end**
**if** $0.35 \leq random \leq 0.55$) **then**
  three element block swap;
**end**
**if** $0.55 \leq random \leq 0.65$ **then**
  four element block swap;
**end**
**if** $random \geq 0.65$ **then**
  no calling of block swap;
**end**
**return** *type of block swap*

**Algoritmo 3:** Choosing the length of block swap

If there were improvement applying these operators, a new solution is defined, along with its objective function. The next step of the algorithm is the acceptance criteria. If a new sequence has a smaller objective function than the current one, a combination of swapping two and three elements, block swapping with lengths equal to two, three and four and block swapping with two elements chosen from positions from all sequence are applied until no improvement is found. However, if a worse solution is found and it is not accepted, instead of applying a swapping procedure, a random initial solution is generated.

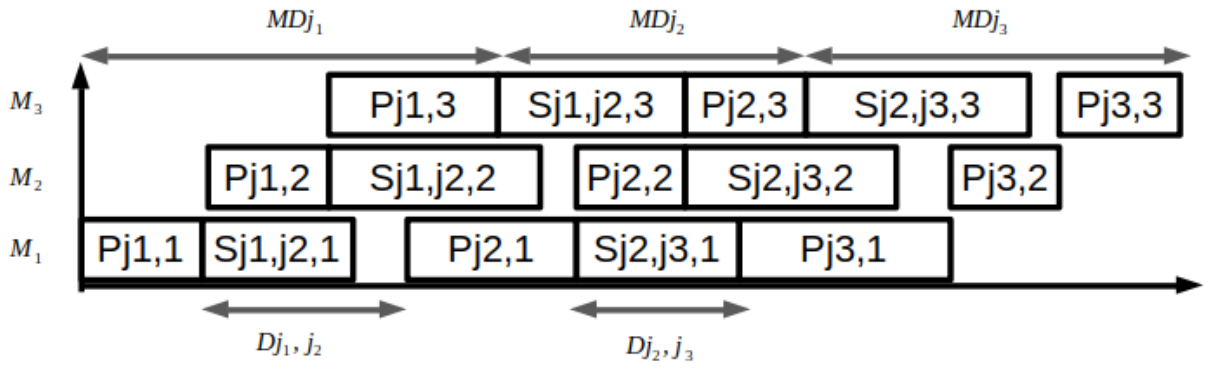
Iterations will continue until a stopping criterion is reached. In this work we consider the primary stopping criteria as $(n * m/2) * (t)ms$, when the minimum TFT is returned. In addition, if the number of iterations without generating another best solution reaches

$4 * n$, processing is also stopped.

### 3.5.6 Speed-up calculation

For every new job insertion or swapping, the objective function has to be computed again, which is costly as this occurs several times. A speed-up method can be applied to optimize the computational time used in this procedure. As the NWFSP can be reduced to an ATSP, a distance matrix can be used to calculate the objective function. The distance between completion times of adjacent jobs on the last machine are first calculated and stored in a matrix of distances. Afterwards, increments are computed using the distance matrix to calculate the new objective function of the sequence being evaluated (LI *et al.*, 2018).

Figure 3 – Distance matrix




Source: Adapted from Qian *et al.* (2011)

Adapting the formulae proposed by Qian *et al.* (2011) and Li, Wang and Wu (2008) for the NWFSP with SDST and TFT minimization, we first calculate the distance between completion times of every pair of jobs $j$ and $k$ on the last machine $m$.

$$MD_{m-1jk} = \begin{cases} \max\{s_{0jk} + p_{k0} - p_{j1}, s_{1jk}\} + p_{k1}, & m = 1 \\ \max\{MD_{m-2jk} - p_{jm}, s_{mjk}\} + p_{km}, & m = 2, ..., m \end{cases} \tag{3.19}$$

Then, we calculate the minimum delay on first machine between the completion of job j and start of job k to populate the distance matrix.

$$D_{jk} = MD_{m-1jk} + \sum p_j - \sum p_k - p_{j0} \tag{3.20}$$

where $\sum p_j$ and $\sum p_k$ are the sum of the processing times of jobs $j$ or $k$ on all machines.

To calculate the objective function, an adaptation was made from Li, Wang and Wu (2008) for the TFT. Following are two options to calculate TFT using the distance matrix, in the first one TFT is calculated as showed in the next equation:

$$TFT = TFT_{m=0} + (n - pos_j) * MD_{m-1j-1j} \tag{3.21}$$

$pos_j$ is the position of job $j$ on the sequence and $TFT_{m=0}$ is defined below:

$$TFT_{m=0} = (n) * \sum p_{j=0i} \tag{3.22}$$

$\sum p_{j=0i}$ is the sum of processing times of the job on the first position of the sequence on all machines.

For the second option, the one that was chose to be used in our method, the starting time of each job is first calculated, knowing that the stating time of job on first position is zero.

$$ST_j = ST_{j-1} + p_{j-11} + D_{j-1j}. \tag{3.23}$$

After calculating the starting time it is possible to calculate the completion time of every job on every machine.

$$C_{ji} = ST_j + \sum p_{ji}, \tag{3.24}$$

where $\sum p_{ji}$ is the total sum of processing times of job $j$ on every machines.

Finally, the TFT is the sum of all completion times on the last machine.

For the swapping procedures the objective function can be calculated more efficiently with a speed-up method by summing up an increment of the sequence after the swap to the actual objective function value. In this way, it is not needed to calculate the full TFT for every swap. The increment of the objective function is calculated by summing up the distances on the last machine between the positions of the swapped jobs $j$ and $j + 1$ and removing the value of the distances between the same positions in the sequence before the swap. As we are dealing with TFT, the distances must be multiplied by $(n - pos_j)$. $TFT'$ is the TFT of the sequence after swapping.

$$TFT' = TFT + [(n - pos_j) * MD'_{m-1jj+1}] - [(n - pos_j) * MD_{m-1jj+1}] \tag{3.25}$$

## 3.6 Iterated Greedy

The IG algorithm proposed by Nagano, Miyata and Araújo (2015) and used as comparison in this study is described as follows.

1. Initialization: Initial solution generation using NEH adapted to the Total Shortest Processing Time rule for TFT minimization;
2. Destruction: $d = 4$ elements are randomly removed from the sequence, generating two partial sequences with $d$ and $n - d$ elements;
3. Reconstruction: all destructed jobs are reinserted using the NEH insertion mechanism until the reconstructed sequence has $n$ elements;
4. Insertion and permutation neighborhood local search: a combination of a local search procedure equivalent to swap of two elements and block swap with block length of two is applied after reconstruction;
5. Acceptance criterion: the criterion based on the simulated annealing is also applied in the IG method, the temperature is defined as

$$Temperature = T\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{j,i}}{n * m * 10} \tag{3.26}$$

   for $T = 0.5$ and $\sum_{i=1}^{m}\sum_{j=1}^{n} p_{j,i}$ the total sum of processing times of all jobs on all machines.
6. Stopping criterion: the iterations are interrupted when computational time reaches the value of $(n * m/2) * (t = 90)ms$.

## 3.7 Computational results

An extensive computational experiment was performed to compare the ALNS being proposed to the IG proposed by Nagano, Miyata and Araújo (2015) for the NWFSP with SDST and TFT as performance criterion.

Both algorithms were coded using the C++ programming language and tested in a computer with a Intel Core i7-8565U, 1.80GHz × 8 processor and 8.00 GB of RAM memory.

### 3.7.1 Instances

The data set used in the computational tests was first proposed by Taillard (1993) for flow shop scheduling problems and then adapted and expended by Ruiz and Stützle (2008) for the same problem but considering no-wait requirement and sequence-dependent setup times. The set of 480 instances is composed of four subsets denoted as SSD10,

SSD50, SSD100 and SSD125, which differ among them on the rate of setup times in relation to processing times $p_{i,j}$ (uniformly distributed in [1,99]), set at most at 10%, 25%, 50% and 125% and uniformly distributed in the range [1, 9], [1, 49], [1, 99] and [1, 124], respectively. These instances are also divided into twelve groups of ten instances each according to the combination of number of jobs $n = 20, 50, 100, 200, 500$ and number of machines $m = 5, 10, 20$.

### 3.7.2 Analysis of results

Tables 4, 5, 6 and 7 show the results for the four subsets of instances proposed by Ruiz and Stützle (2008). Each instance was run five times per method, the best of the five runs was selected and the average result was calculated. The stopping criterion used was the maximum computational time equal to $(n * m/2) * (t)ms$, for $t = 90$, the same criterion chosen by the authors being compared.

Average Relative Percentage Deviation (ARPD) values were calculated using equation (3.27). This gap allows us to compare the results of a method (TFT) with the best obtained result (TFT*) for a given instance.

$$ARPD(\%) = \frac{TFT - TFT^*}{TFT^*} * 100. \tag{3.27}$$

As we can see in the result tables, the gap values compare the IG and the ALNS results against the best found result. These percentages are an average of a group of ten deviations (ten instances for each combination of jobs and machines). Our ALNS performs much better than the IG, particularly in larger instances.

Both algorithms obtained similar results for the first 30 instances of each subset (small size instances), for some subsets the deviation is zero for both methods. For medium and large size instances ALNS was superior, which leads to a positive ARPD when comparing the IG results to the results achieved by the ALNS.

The charts 4, 5, 6 and 7 were designed to show how the deviations of the IG average results and the deviations of the IG best results behave when the number of jobs and machines varies. Since the IG method obtained positive ARPD values, only its performance was used in this analysis. Each chart represents an instance set. We can notice that in general the larger the number of jobs the higher the deviation (x axis). However, the same does not occur to the number of machines. The performance of the IG method is better for 20 machines, than for 10 or 5. We can also conclude that for a 500 job instance the gap between the deviations of the average results and of the best results is wider than for smaller instances, which shows that the method becomes less robust for large size instances, as the method is not able to run enough iterations in the defined computational time.

Table 4 – Comparing ALNS and IG results for instances SSD10

| Instance | Average over 5 runs | | | | Best of 5 runs | | | |
|---|---|---|---|---|---|---|---|---|
| | ALNS | ARPD | IG | ARPD | ALNS | ARPD | IG | ARPD |
| $n$=20, $m$=5 | 1700 | 0.00 | 1700 | 0.00 | 1700 | 0.00 | 1700 | 0.00 |
| $n$=20, $m$=10 | 24541 | 0.00 | 24541 | 0.00 | 24541 | 0.00 | 24541 | 0.00 |
| $n$=20, $m$=20 | 39489 | 0.00 | 39491 | 0.00 | 39489 | 0.00 | 39491 | 0.00 |
| $n$=50, $m$=5 | 85904 | 0.00 | 87083 | 1.37 | 85859 | 0.00 | 86851 | 1.16 |
| $n$=50, $m$=10 | 118871 | 0.00 | 120055 | 1.00 | 118801 | 0.00 | 119742 | 0.79 |
| $n$=50, $m$=20 | 171333 | 0.00 | 172956 | 0.95 | 171281 | 0.00 | 172654 | 0.80 |
| $n$=100, $m$=5 | 318427 | 0.00 | 326311 | 2.48 | 317545 | 0.00 | 325644 | 2.55 |
| $n$=100, $m$=10 | 428669 | 0.00 | 437772 | 2.13 | 427563 | 0.00 | 436720 | 2.14 |
| $n$=100, $m$=20 | 589182 | 0.00 | 600393 | 1.90 | 587616 | 0.00 | 599179 | 1.97 |
| $n$=200, $m$=10 | 1604804 | 0.00 | 1644444 | 2.47 | 1598544 | 0.00 | 1641591 | 2.69 |
| $n$=200, $m$=20 | 2136764 | 0.00 | 2178414 | 1.95 | 2128307 | 0.00 | 2174149 | 2.15 |
| $n$=500, $m$=20 | 12469660 | 0.00 | 12643392 | 1.40 | 12291773 | 0.00 | 12622005 | 2.69 |
| Average | | 0.00 | | 1.30 | | 0.00 | | 1.41 |

Table 5 – Comparing ALNS and IG results for instances SSD50

| Instance | Average over 5 runs | | | | Best of 5 runs | | | |
|---|---|---|---|---|---|---|---|---|
| | ALNS | ARPD | IG | ARPD | ALNS | ARPD | IG | ARPD |
| $n$=20, $m$=5 | 20567 | 0.00 | 20569 | 0.01 | 20567 | 0.00 | 20567 | 0.00 |
| $n$=20, $m$=10 | 28736 | 0.00 | 28739 | 0.01 | 28736 | 0.00 | 28739 | 0.01 |
| $n$=20, $m$=20 | 43981 | 0.00 | 43984 | 0.01 | 43981 | 0.00 | 43981 | 0.00 |
| $n$=50, $m$=5 | 108165 | 0.00 | 109857 | 1.57 | 108072 | 0.00 | 109500 | 1.32 |
| $n$=50, $m$=10 | 143759 | 0.00 | 145497 | 1.21 | 143700 | 0.00 | 145188 | 1.04 |
| $n$=50, $m$=20 | 199322 | 0.00 | 201138 | 0.91 | 199198 | 0.00 | 200676 | 0.74 |
| $n$=100, $m$=5 | 401004 | 0.00 | 413537 | 3.13 | 399495 | 0.00 | 412231 | 3.19 |
| $n$=100, $m$=10 | 530909 | 0.00 | 542252 | 2.14 | 529467 | 0.00 | 541158 | 2.21 |
| $n$=100, $m$=20 | 699753 | 0.00 | 713097 | 1.91 | 698663 | 0.00 | 711567 | 1.85 |
| $n$=200, $m$=10 | 2009650 | 0.00 | 2056585 | 2.34 | 2003560 | 0.00 | 2051487 | 2.39 |
| $n$=200, $m$=20 | 2582764 | 0.00 | 2627327 | 1.73 | 2574668 | 0.00 | 2622650 | 1.86 |
| $n$=500, $m$=20 | 15194426 | 0.00 | 15466404 | 1.79 | 15064236 | 0.00 | 15443106 | 2.52 |
| Average | | 0.00 | | 1.40 | | 0.00 | | 1.43 |

Table 6 – Comparing ALNS and IG results for instances SSD100

| Instance | Average over 5 runs | | | | Best of 5 runs | | | |
|---|---|---|---|---|---|---|---|---|
| | ALNS | ARPD | IG | ARPD | ALNS | ARPD | IG | ARPD |
| $n$=20, $m$=5 | 24943 | 0.00 | 24945 | 0.01 | 24943 | 0.00 | 24943 | 0.00 |
| $n$=20, $m$=10 | 34140 | 0.00 | 34140 | 0.00 | 34140 | 0.00 | 34140 | 0.00 |
| $n$=20, $m$=20 | 50127 | 0.00 | 50127 | 0.00 | 50127 | 0.00 | 50127 | 0.00 |
| $n$=50, $m$=5 | 133434 | 0.00 | 136374 | 2.21 | 133223 | 0.00 | 135794 | 1.93 |
| $n$=50, $m$=10 | 177885 | 0.00 | 180684 | 1.57 | 177760 | 0.00 | 180260 | 1.41 |
| $n$=50, $m$=20 | 239031 | 0.00 | 241389 | 0.98 | 238905 | 0.00 | 240886 | 0.83 |
| $n$=100, $m$=5 | 498192 | 0.00 | 517940 | 3.97 | 495672 | 0.00 | 516123 | 4.13 |
| $n$=100, $m$=10 | 663655 | 0.00 | 680682 | 2.57 | 662083 | 0.00 | 679082 | 2.57 |
| $n$=100, $m$=20 | 857618 | 0.00 | 874462 | 1.96 | 856100 | 0.00 | 872532 | 1.92 |
| $n$=200, $m$=10 | 2535668 | 0.00 | 2604197 | 2.70 | 2524430 | 0.00 | 2597380 | 2.89 |
| $n$=200, $m$=20 | 3208220 | 0.00 | 3267244 | 1.84 | 3195018 | 0.00 | 3261285 | 2.07 |
| $n$=500, $m$=20 | 19047941 | 0.00 | 19439566 | 2.06 | 18899249 | 0.00 | 19410508 | 2.71 |
| Average | | 0.00 | | 1.66 | | 0.00 | | 1.70 |

Table 7 – Comparing ALNS and IG results for instances SSD125

| Instance | Average over 5 runs | | | | Best of 5 runs | | | |
|---|---|---|---|---|---|---|---|---|
| | ALNS | ARPD | IG | ARPD | ALNS | ARPD | IG | ARPD |
| $n$=20, $m$=5 | 27113 | 0.00 | 27113 | 0.00 | 27113 | 0.00 | 27113 | 0.00 |
| $n$=20, $m$=10 | 37023 | 0.00 | 37023 | 0.00 | 37023 | 0.00 | 37023 | 0.00 |
| $n$=20, $m$=20 | 53460 | 0.00 | 53470 | 0.02 | 53460 | 0.00 | 53460 | 0.00 |
| $n$=50, $m$=5 | 146142 | 0.00 | 149875 | 2.56 | 145894 | 0.00 | 149317 | 2.35 |
| $n$=50, $m$=10 | 195411 | 0.00 | 198547 | 1.61 | 195192 | 0.00 | 198026 | 1.45 |
| $n$=50, $m$=20 | 259537 | 0.00 | 262339 | 1.08 | 259325 | 0.00 | 261572 | 0.87 |
| $n$=100, $m$=5 | 546232 | 0.00 | 570442 | 4.43 | 543289 | 0.00 | 568253 | 4.60 |
| $n$=100, $m$=10 | 732610 | 0.00 | 753371 | 2.83 | 730807 | 0.00 | 751004 | 2.77 |
| $n$=100, $m$=20 | 939661 | 0.00 | 958595 | 2.02 | 937733 | 0.00 | 956715 | 2.03 |
| $n$=200, $m$=10 | 2808033 | 0.00 | 2886589 | 2.80 | 2794497 | 0.00 | 2879828 | 3.06 |
| $n$=200, $m$=20 | 3543521 | 0.00 | 3609907 | 1.87 | 3533667 | 0.00 | 3602060 | 1.94 |
| $n$=500, $m$=20 | 21143569 | 0.00 | 21541743 | 1.88 | 20961191 | 0.00 | 21513855 | 2.64 |
| Average | | 0.00 | | 1.76 | | 0.00 | | 1.81 |

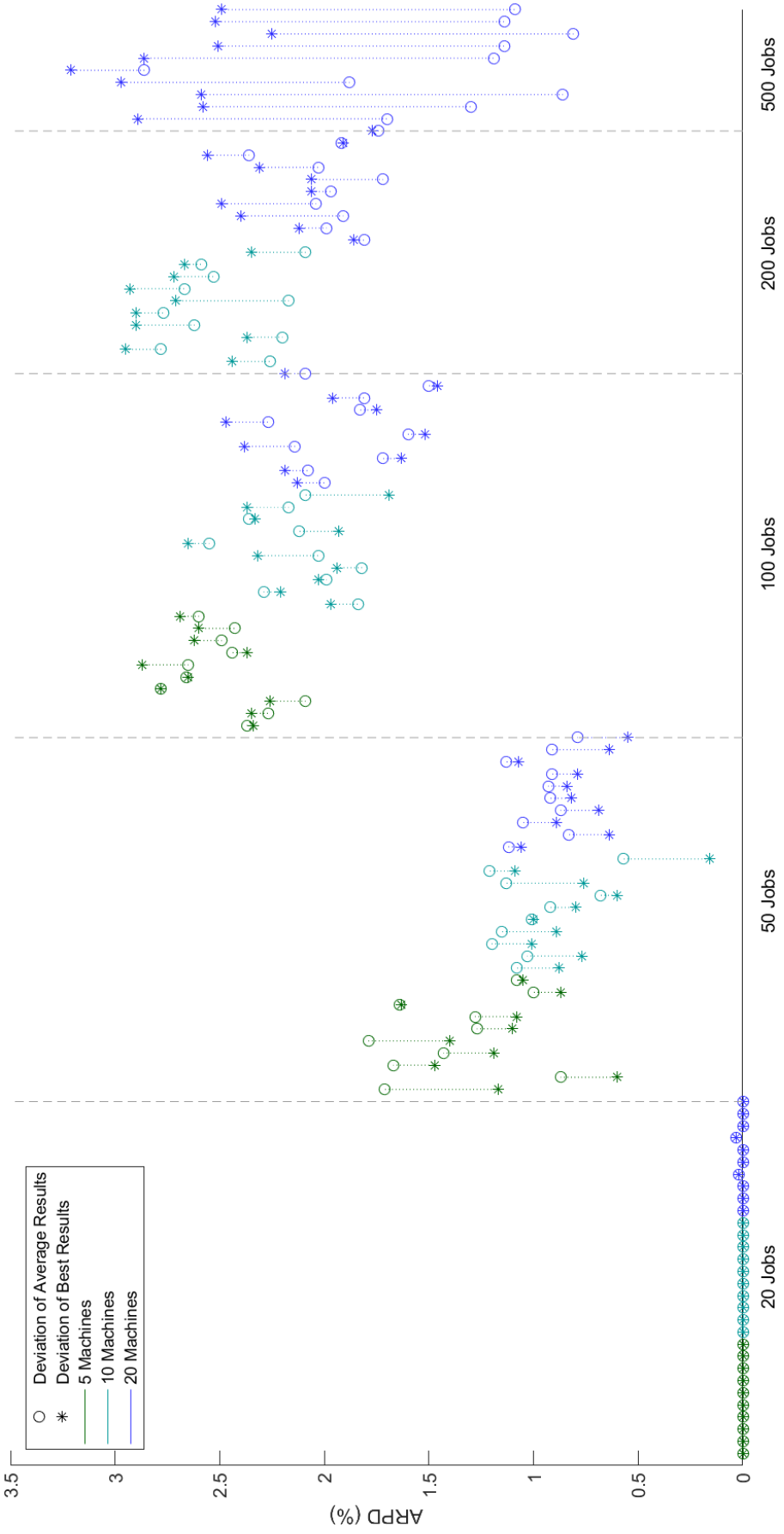



Figure 4 – IG: ARPD for number of jobs and machines - SSD10

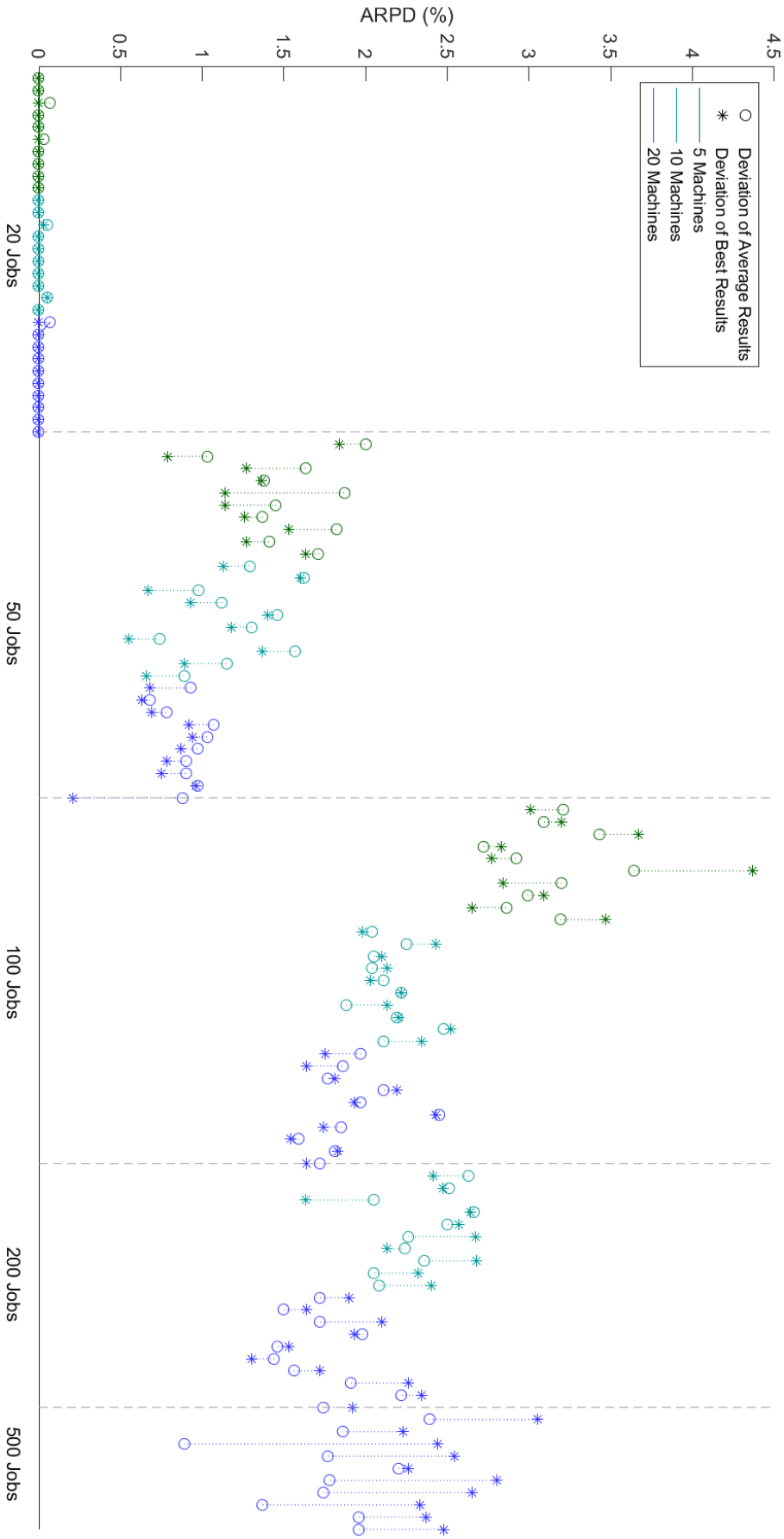



Figure 5 – IG: ARPD for number of jobs and machines - SSD50

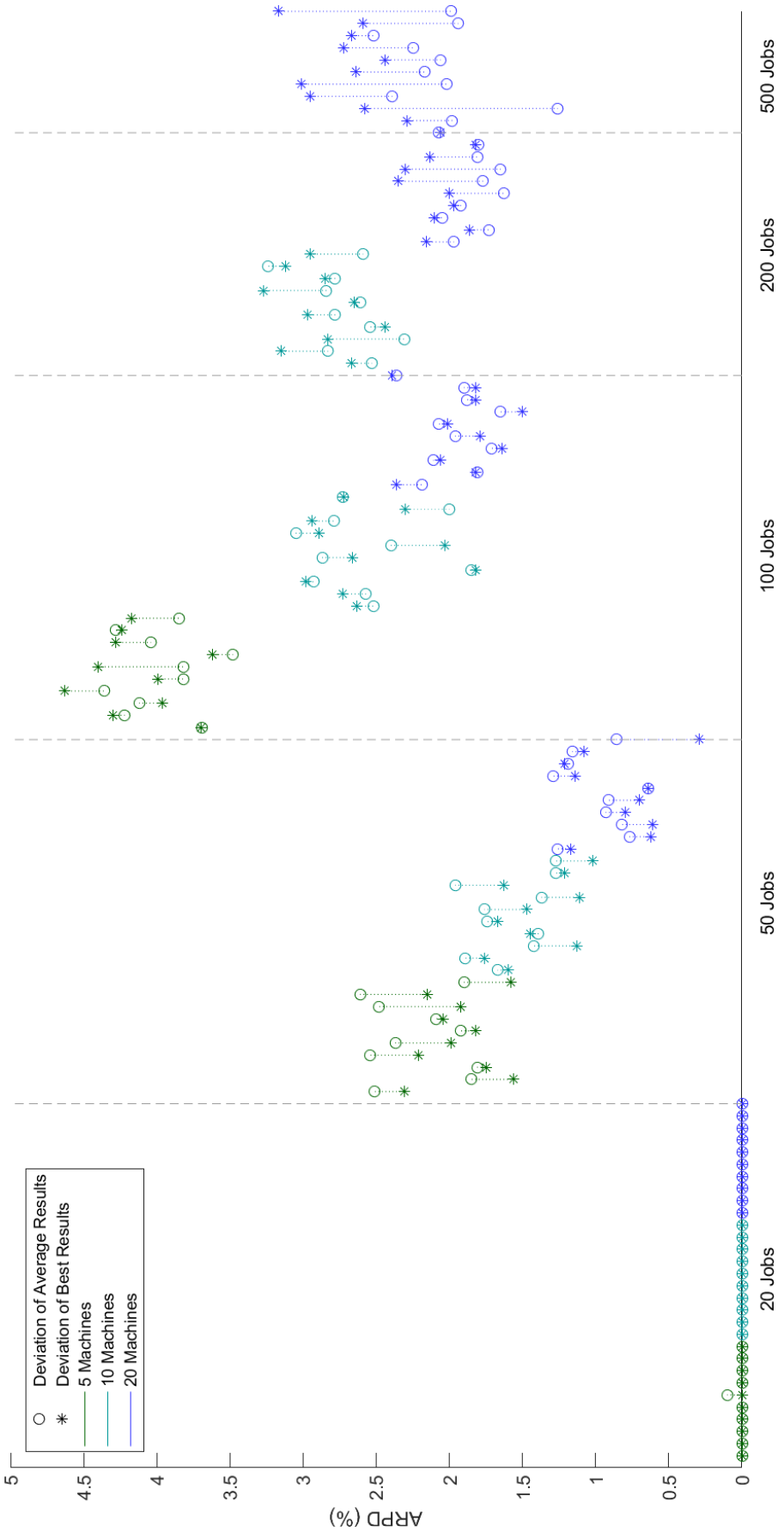



Figure 6 – IG: ARPD for number of jobs and machines - SSD100

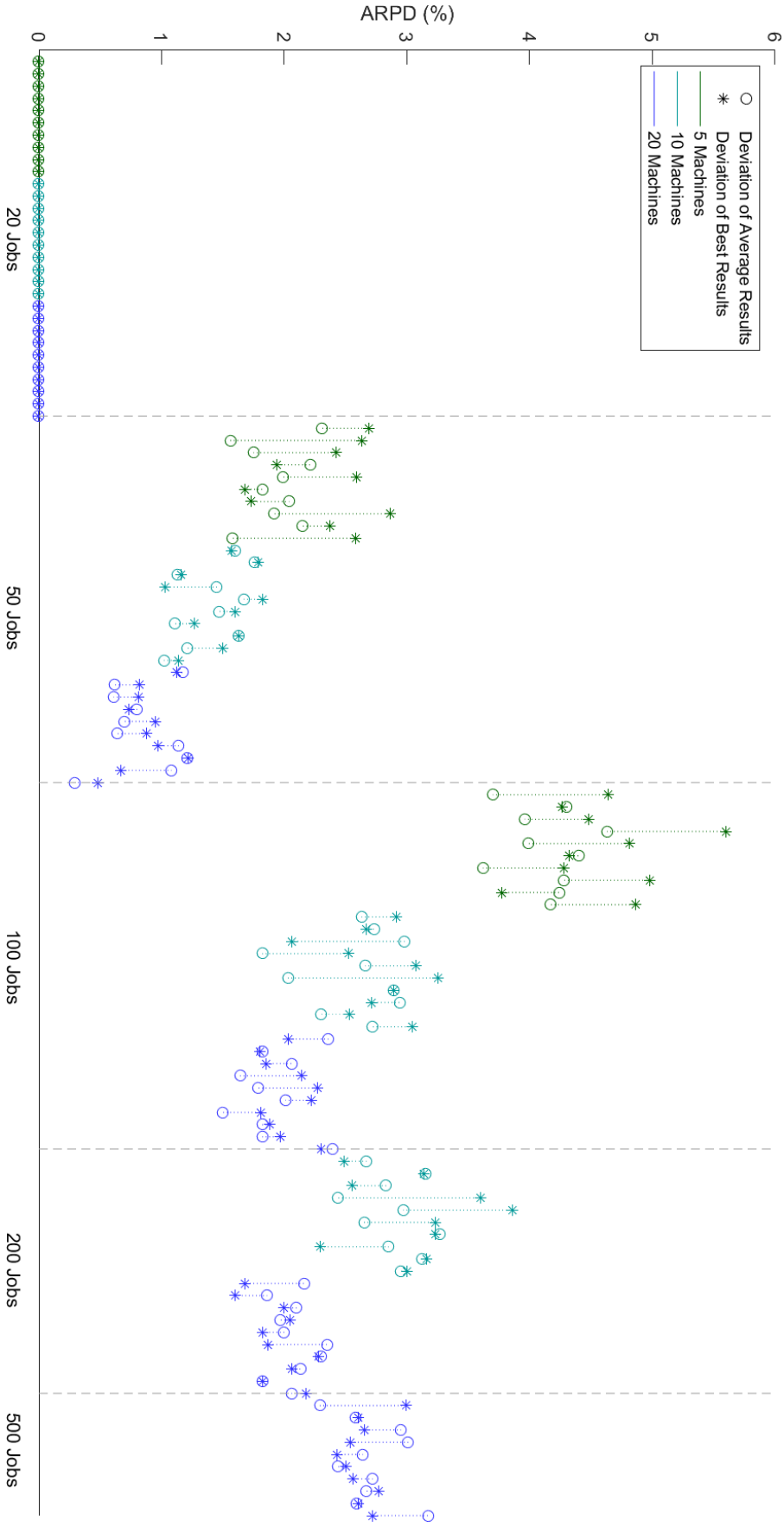



Figure 7 – IG: ARPD for number of jobs and machines - SSD125

When comparing all charts we can see the same pattern. The deviations for 20 jobs tend to zero and, in general, the results are better for smaller instance sizes (number of jobs), even though for 100 and 200 job instances with 10 and 20 machines the results are very close. The largest instance size has a slighted different behaviour for the SSD125 instance set, the average deviation and best deviation are kept closer than for the other sets.

Another conclusion that could be made is that for small and medium instance sizes, the ARPD of the best of five results is lower than that of the average results, except for the last chart. Which is different for large sized instances, when the chart points change positions. This leads to the conclusion that, for small and medium size instances, in most cases, analysing the best of five results lead to a smaller deviation than adopting the average over five runs.

Tables 8, 9, 10 and 11 show the percentages of each destruction (D1, D2, D3, D4 and D5) and reconstruction (R1, R2, R3, R4, R5 and R6) operators called by the ALNS reward mechanism to solve each set of instances (SSD10, SSD50, SSD100 and SSD125). The first lines of the table refer to small instances with maximum number of jobs equal to 20, then to medium and large size instances with the number of jobs ranging from 30 to 200. The largest size instances with 500 jobs were not included in this analysis because of the small number of total iterations. For these tables, one instance per group of instances was chosen to be analysed, the applied criteria was selecting the one with the smallest deviation between the average and the best TFT of five runs, which shows more stability in the results.

Table 8 – The calling percentages of destruction and reconstruction operators - SSD10

| Instance | D1 | D2 | D3 | D4 | D5 | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$=20, $m$=5 | 11.8% | 20.0% | 24.3% | 30.3% | 13.6% | 5.9% | 5.0% | 26.0% | 4.3% | 25.6% | 33.1% |
| $n$=20, $m$=10 | 13.0% | 20.9% | 33.0% | 17.7% | 15.4% | 5.2% | 4.7% | 25.5% | 5.9% | 28.1% | 30.6% |
| $n$=20, $m$=20 | 9.4% | 20.4% | 34.4% | 25.6% | 10.3% | 4.7% | 3.7% | 26.6% | 4.3% | 25.6% | 35.1% |
| $n$=50, $m$=5 | 22.6% | 22.5% | 21.9% | 17.7% | 15.2% | 10.5% | 9.7% | 19.0% | 12.1% | 23.8% | 24.9% |
| $n$=50, $m$=10 | 21.4% | 21.5% | 24.3% | 18.0% | 14.7% | 8.5% | 9.2% | 20.6% | 11.1% | 22.6% | 28.0% |
| $n$=50, $m$=20 | 20.0% | 28.4% | 15.0% | 19.0% | 17.6% | 8.0% | 7.7% | 22.8% | 8.0% | 22.0% | 31.6% |
| $n$=100, $m$=5 | 21.0% | 19.4% | 19.1% | 19.9% | 20.7% | 14.3% | 15.6% | 18.3% | 15.3% | 16.6% | 19.9% |
| $n$=100, $m$=10 | 24.2% | 21.2% | 17.2% | 20.5% | 16.9% | 14.9% | 13.8% | 18.3% | 14.4% | 18.7% | 19.9% |
| $n$=100, $m$=20 | 24.4% | 20.9% | 16.6% | 17.4% | 20.7% | 14.0% | 13.9% | 18.4% | 13.9% | 16.8% | 23.1% |
| $n$=200, $m$=10 | 22.1% | 19.5% | 20.3% | 20.3% | 17.8% | 17.1% | 15.0% | 16.2% | 17.3% | 17.6% | 16.8% |
| $n$=200, $m$=20 | 19.0% | 20.6% | 9.0% | 21.1% | 20.3% | 14.7% | 17.7% | 18.3% | 14.4% | 15.7% | 19.1% |

Analysing the percentages of calling a destruction and reconstruction operator automated by the reward mechanism in the ALNS algorithm, it can be noted that for smaller instances the chances are more distinct from the initial percentages (20% for each destructor and 16.7% for each reconstructor), due to the higher number of total iterations allowing more segments and a better calibration of the scores and weights.

Table 9 – The calling percentages of destruction and reconstruction operators - SSD50

| Instance | D1 | D2 | D3 | D4 | D5 | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$=20, $m$=5 | 24.5% | 19.3% | 20.5% | 18.3% | 17.4% | 10.2% | 12.8% | 18.4% | 11.7% | 17.9% | 29.0% |
| $n$=20, $m$=10 | 19.9% | 18.5% | 26.1% | 20.4% | 15.1% | 9.7% | 9.0% | 21.7% | 9.4% | 21.2% | 29.0% |
| $n$=20, $m$=20 | 13.6% | 19.1% | 29.5% | 22.7% | 15.1% | 3.5% | 5.0% | 25.3% | 5.7% | 24.6% | 35.9% |
| $n$=50, $m$=5 | 29.0% | 20.0% | 17.8% | 15.3% | 18.0% | 10.8% | 12.5% | 15.7% | 11.7% | 17.4% | 32.0% |
| $n$=50, $m$=10 | 27.6% | 22.1% | 14.9% | 18.0% | 17.4% | 7.8% | 9.2% | 18.1% | 11.2% | 19.4% | 34.3% |
| $n$=50, $m$=20 | 22.0% | 22.1% | 26.5% | 16.4% | 13.1% | 6.9% | 7.8% | 17.3% | 9.3% | 16.7% | 41.9% |
| $n$=100, $m$=5 | 27.9% | 18.4% | 17.9% | 15.8% | 19.9% | 15.2% | 15.3% | 14.5% | 15.1% | 19.0% | 20.9% |
| $n$=100, $m$=10 | 27.6% | 17.7% | 18.4% | 18.3% | 17.9% | 15.2% | 12.3% | 15.6% | 12.5% | 18.5% | 26.0% |
| $n$=100, $m$=20 | 25.9% | 19.1% | 19.0% | 16.3% | 19.8% | 12.8% | 14.7% | 15.1% | 12.5% | 19.2% | 25.8% |
| $n$=200, $m$=10 | 22.9% | 18.4% | 19.7% | 19.3% | 19.8% | 16.7% | 16.3% | 15.8% | 17.7% | 17.1% | 16.3% |
| $n$=200, $m$=20 | 22.8% | 19.8% | 19.2% | 18.3% | 19.9% | 16.8% | 16.8% | 16.2% | 18.0% | 14.6% | 17.5% |

Table 10 – The calling percentages of destruction and reconstruction operators - SSD100

| Instance | D1 | D2 | D3 | D4 | D5 | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$=20, $m$=5 | 19.9% | 19.5% | 24.1% | 18.4% | 18.1% | 10.6% | 15.2% | 13.1% | 15.5% | 13.3% | 32.3% |
| $n$=20, $m$=10 | 17.9% | 20.4% | 24.3% | 21.3% | 16.1% | 9.8% | 8.7% | 16.4% | 10.0% | 16.3% | 38.9% |
| $n$=20, $m$=20 | 11.9% | 18.7% | 32.3% | 24.2% | 12.8% | 5.5% | 4.9% | 19.3% | 5.5% | 21.5% | 43.3% |
| $n$=50, $m$=5 | 34.7% | 21.8% | 16.0% | 12.6% | 14.9% | 9.3% | 16.0% | 13.9% | 12.6% | 14.4% | 33.8% |
| $n$=50, $m$=10 | 36.5% | 18.7% | 13.3% | 15.2% | 16.3% | 9.5% | 12.9% | 16.3% | 13.2% | 15.6% | 32.5% |
| $n$=50, $m$=20 | 32.9% | 25.3% | 14.1% | 11.5% | 16.3% | 9.1% | 8.9% | 18.5% | 10.2% | 17.4% | 35.8% |
| $n$=100, $m$=5 | 35.3% | 18.0% | 15.0% | 15.1% | 16.5% | 13.2% | 15.3% | 16.5% | 15.5% | 13.4% | 26.1% |
| $n$=100, $m$=10 | 34.4% | 16.7% | 15.3% | 15.8% | 17.8% | 14.2% | 11.6% | 16.7% | 14.3% | 15.8% | 27.3% |
| $n$=100, $m$=20 | 28.5% | 20.7% | 15.9% | 16.6% | 18.3% | 12.1% | 14.5% | 18.3% | 13.9% | 15.6% | 25.6% |
| $n$=200, $m$=10 | 19.9% | 20.3% | 21.6% | 19.7% | 18.5% | 15.7% | 16.6% | 18.2% | 16.6% | 15.9% | 16.9% |
| $n$=200, $m$=20 | 21.5% | 20.5% | 20.1% | 17.7% | 20.1% | 17.0% | 16.3% | 17.4% | 15.2% | 16.4% | 17.7% |

Table 11 – The calling percentages of destruction and reconstruction operators - SSD125

| Instance | D1 | D2 | D3 | D4 | D5 | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$=20, $m$=5 | 25.6% | 21.9% | 18.5% | 16.5% | 17.5% | 11.8% | 17.6% | 13.3% | 15.4% | 12.6% | 29.4% |
| $n$=20, $m$=10 | 19.3% | 18.0% | 23.6% | 21.6% | 17.5% | 10.5% | 11.0% | 17.3% | 10.2% | 16.8% | 34.2% |
| $n$=20, $m$=20 | 19.5% | 18.9% | 23.1% | 24.6% | 13.9% | 10.4% | 9.7% | 15.2% | 10.2% | 16.2% | 38.2% |
| $n$=50, $m$=5 | 38.2% | 15.6% | 17.7% | 12.9% | 15.6% | 9.5% | 15.5% | 13.6% | 16.9% | 14.1% | 30.4% |
| $n$=50, $m$=10 | 33.2% | 16.6% | 15.1% | 16.4% | 18.8% | 7.7% | 12.1% | 15.4% | 10.7% | 16.0% | 38.1% |
| $n$=50, $m$=20 | 36.6% | 20.0% | 14.3% | 13.4% | 15.6% | 11.0% | 11.9% | 14.7% | 13.0% | 14.6% | 34.9% |
| $n$=100, $m$=5 | 34.4% | 18.7% | 13.4% | 16.8% | 16.7% | 14.7% | 15.1% | 15.1% | 15.7% | 15.8% | 23.6% |
| $n$=100, $m$=10 | 36.1% | 14.9% | 15.1% | 16.2% | 17.7% | 14.3% | 14.1% | 15.0% | 16.2% | 14.2% | 26.2% |
| $n$=100, $m$=20 | 31.9% | 16.1% | 15.6% | 18.3% | 18.2% | 15.3% | 13.7% | 16.0% | 12.7% | 15.6% | 26.6% |
| $n$=200, $m$=10 | 19.9% | 22.9% | 18.4% | 21.0% | 17.8% | 18.8% | 15.9% | 15.5% | 17.5% | 15.3% | 17.0% |
| $n$=200, $m$=20 | 21.5% | 20.1% | 20.9% | 17.3% | 20.2% | 14.7% | 18.3% | 16.5% | 17.2% | 16.0% | 17.3% |

Higher percentages mean that the operator is more adequate for that instance problem, as it generates better solutions and then receives a higher score, leading to a higher probability of being called in the next segment of iterations. For example, for SSD10 small size instances, reconstructor R6 has about six times more chances to be used in the heuristic than reconstructors R1, R2 and R4, what shows that this operator has great performance for this subset of problems.

## 3.8 Conclusions

In this research we proposed an ALNS algorithm to solve the no-wait flow shop scheduling problem with sequence dependent setup times and TFT minimization. The operators and mechanisms present in the heuristic method and their differences in relation to the well-known IG algorithm were described. We can conclude from the computational experiment results that inserting more engineered mechanisms, such as the ALNS structure, can bring better results in terms of quality. In addition, bringing algorithms from other areas of knowledge, such as transportation science, can lead to high performance applications in the scheduling research.

For future research, we see a multitude of possible applications of the ALNS method for the flow shop scheduling problem, from varying the type of setup to adding new constraints or changing the objective function. Since the ALNS method obtained good results, we also suggest exploring the operators and searching mechanisms to improve its performance for large instance sizes. Another possibility is adding a exact method to the heuristic to generate the initial solution, which could lead the matheuristic to optimal solutions. Finally, applying the proposed algorithm to real manufacturing problems would produce great analysis.

# 4 FINAL CONSIDERATIONS

The no-wait flow shop scheduling problem with setup times has been broadly studied and there is still space for more research lines as there are multiple application possibilities combining a range of constraints and performance measures to this scheduling problem.

Amongst the solution methods presented in the literature review, we highlight the good performance of methods first developed for transportation problems, such as the VRP, when applied in scheduling problems. This good performance reinforced our motivation to propose an ALNS algorithm for the no-wait flow shop scheduling problem. The alignment between no-wait and the TFT criteria was the reason for choosing this objective function. In addition, sequence dependent setup time completes the model to make it more applicable to real manufacturing problems.

Our solution method showed better results than the traditional IG structure, proving that an automated mechanism for selecting destruction and reconstruction operators, besides the quality of the operators and the reheating system, is a good strategy to make the heuristic more robust. It could also be concluded that aligning operators to destruct and reconstruct the sequence to operators that swap elements to search new solution areas is also a good pathway, what has been followed by the majority of the authors. In addition, it can also be a good decision to insert some randomness to the algorithm, for example by generating a random solution when a bad solution is found and it is not accepted by the acceptance criteria.

In terms of computational time, the adoption of a speed-up method decreases the processing time considerably. Swapping methods require that the objective function be calculated several times, if no speed-up function is considered, the number of iterations until reaching the time based stopping criteria would not be enough for finding good solutions. Objective function calculation for partial sequences in the destruction and reconstruction operators also applies this method to decrease computational effort.

In conclusion, we gathered the most used and with the best performance components in the heuristics present in the literature review and applied them in the ALNS structure, adapting the method for the scheduling problem and generating high quality solutions in a reasonable time.

# REFERENCES

ALDOWAISAN, T. A new heuristic and dominance relations for no-wait flowshops with setups. **Computers & Operations Research**, v. 28, n. 6, p. 563–584, 2001.

ALDOWAISAN, T.; ALLAHVERDI, A. Total flowtime in no-wait flowshops with separated setup times. **Computers & Operations Research**, v. 25, n. 9, p. 757–765, 1998.

ALDOWAISAN, T.; ALLAHVERDI, A. Three-machine no-wait flowshop with separate setup and removal times to minimize total completion time. **International Journal of Industrial Engineering : Theory Applications and Practice**, v. 11, n. 2, p. 113–123, 2004.

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

ALLAHVERDI, A. A survey of scheduling problems with no-wait in process. **European Journal of Operational Research**, v. 255, n. 3, p. 665–686, 2016.

ALLAHVERDI, A.; ALDOWAISAN, T. Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. **Journal of the Operational Research Society**, v. 52, n. 4, p. 449–462, 2001.

ALLAHVERDI, A.; AYDILEK, H. Total completion time with makespan constraint in no-wait flowshops with setup times. **European Journal of Operational Research**, v. 238, n. 3, p. 724–734, 2014.

APPLEGATE, D. L. *et al.* **The Traveling Salesman Problem: A Computational Study**. [*S.l.: s.n.*]: Princeton University Press, 2006. (Princeton Series in Applied Mathematics).

ARAÚJO, D. C.; NAGANO, M. S. A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem. **International Journal of Industrial Engineering Computations**, v. 2, n. 1, p. 155–166, 2011.

AZIZI, V.; JABBARI, M.; KHEIRKHAH, A. S. M-machine, no-wait flowshop scheduling with sequence dependent setup times and truncated learning function to minimize the makespan. **International Journal of Industrial Engineering Computations**, v. 7, n. 2, p. 309–322, 2016.

BAKER, K. R.; TRIETSCH, D. **Principles of Sequencing and Scheduling**. New Jersey: John Wiley & Sons, 2009.

BEEZãO, A. C. *et al.* Scheduling identical parallel machines with tooling constraints. **European Journal of Operational Research**, v. 257, n. 3, p. 834–844, 2017. ISSN 0377-2217.

BEHJAT, S.; SALMASI, N. Total completion time minimisation of no-wait flowshop group scheduling problem with sequence dependent setup times. **European Journal of Industrial Engineering**, v. 11, n. 1, p. 22–48, 2017.

BIANCO, L.; DELL'OLMO, P.; GIORDANI, S. Flow shop no-wait scheduling with sequence dependent setup times and release dates. **INFOR**, v. 37, n. 1, p. 3–18, 1999.

BROWN, S. I.; MCGARVEY, R. G.; VENTURA, J. A. Total flowtime and makespan for a no-wait m-machine flowshop with set-up times separated. **Journal of the Operational Research Society**, v. 55, n. 6, p. 614–621, 2004.

CHELLADURAI, A.; AZATH, M.; JENIFFER, J. Integrated search method for flexible job shop scheduling problem using hhs–alns algorithm. **SN Computer Science**, v. 1, 03 2020.

CHENG, C.-Y. *et al.* Minimizing makespan in mixed no-wait flowshops with sequence-dependent setup times. **Computers & Industrial Engineering**, v. 130, p. 338–347, 2019.

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**, v. 21, n. 3, p. 515–524, 1974.

COTA, L. P. *et al.* An adaptive large neighborhood search with learning automata for the unrelated parallel machine scheduling problem. *In*: **2017 IEEE Congress on Evolutionary Computation (CEC)**. [*S.l.: s.n.*], 2017. p. 185–192.

COTA, L. P. *et al.* An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem. **Swarm and Evolutionary Computation**, v. 51, p. 100601, 2019. ISSN 2210-6502.

DILEEPAN, P. A note on minimizing maximum lateness in a two-machine no-wait flowshop. **Computers & Operations Research**, v. 31, n. 12, p. 2111–2115, 2004.

DING, J.-Y. *et al.* An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. **Applied Soft Computing Journal**, v. 30, p. 604–613, 2015.

FONDREVELLE, J.; ALLAHVERDI, A.; OULAMARA, A. Two-machine, no-wait flowshop scheduling problem to minimize maximum lateness with separate set-up and removal times. **International Journal of Agile Manufacturing**, v. 8, n. 2, p. 165–174, 2005.

FRAMINAN, J. M.; GUPTA, J. N. D.; LEISTEN, R. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. **Journal of the Operational Research Society**, v. 55, n. 12, p. 1243–1255, 2004.

FRANÇA, P. M.; JR., G. T.; BURIOL, L. S. Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions. **International Journal of Production Research**, v. 44, n. 5, p. 939–957, 2006.

GRAHAM, R. L. *et al.* Optimization and approximation in deterministic sequencing and scheduling: a survey. *In*: HAMMER, P. L.; JOHNSON, E. L.; KORTE, B. H. (ed.). **Discrete Optimization II**. [*S.l.: s.n.*]: Elsevier, 1979, (Annals of Discrete Mathematics, v. 5). p. 287–326.

GUPTA, J. N. D.; STRUSEVICH, V. A.; ZWANEVELD, C. M. Two-stage no-wait scheduling models with setup and removal times separated. **Computers & Operations Research**, v. 24, n. 11, p. 1025–1031, 1997.

HEJAZI, S. R.; SAGHAFIAN, S. Flowshop-scheduling problems with makespan criterion: a review. **International Journal of Production Research**, v. 43, n. 14, p. 2895–2929, 2005.

ISHIBUCHI, H.; MISAKI, S.; TANAKA, H. Modified simulated annealing algorithms for the flow shop sequencing problem. **European Journal of Operational Research**, v. 81, n. 2, p. 388–398, 1995.

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.

LEE, Y. H.; JUNG, J. W. New heuristics for no-wait flowshop scheduling with precedence constraints and sequence dependent setup time. **Lecture Notes in Computer Science**, v. 3483, n. IV, p. 467–476, 2005.

LI, X.; WANG, Q.; WU, C. Heuristic for no-wait flow shops with makespan minimization. **International Journal of Production Research**, Taylor & Francis, v. 46, n. 9, p. 2519–2530, 2008.

LI, X. *et al.* An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. **Information Sciences**, v. 453, p. 408–425, 2018.

MAASSEN, K.; PEREZ-GONZALEZ, P.; GüNTHER, L. C. Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling. **Computers & Operations Research**, p. 104965, 2020. ISSN 0305-0548.

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**, v. 31, n. 1, p. 59–79, 1993.

MIYATA, H. H.; NAGANO, M. S.; GUPTA, J. N. D. Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. **Computers & Industrial Engineering**, v. 135, p. 79–104, 2019.

NAGANO, M. S.; MIYATA, H. H. Review and classification of constructive heuristics mechanisms for no-wait flow shop problem. **The International Journal of Advanced Manufacturing Technology**, v. 86, n. 5, p. 2161–2174, 2016.

NAGANO, M. S.; MIYATA, H. H.; ARAÚJO, D. C. A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times. **Journal of Manufacturing Systems**, v. 36, p. 224–230, 2015.

NAGANO, M. S.; SILVA, A. A. da; LORENA, L. A. N. A new evolutionary clustering search for a no-wait flow shop problem with set-up times. **Engineering Applications of Artificial Intelligence**, v. 25, n. 6, p. 1114–1120, 2012.

NAGANO, M. S.; SILVA, A. A. da; LORENA, L. A. N. An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times. **Expert Systems with Applications**, v. 41, n. 8, p. 3628–3633, 2014.

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.

PAN, Q.-K.; RUIZ, R. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. **Computers & Operations Research**, v. 40, n. 1, p. 117–128, 2013.

PANG, K.-W. A genetic algorithm based heuristic for two machine no-wait flowshop scheduling problems with class setup times that minimizes maximum lateness. **International Journal of Production Economics**, v. 141, n. 1, p. 127–136, 2013.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. 3rd. ed. New York: Springer, 2008.

QIAN, B. *et al.* A differential evolution algorithm with two speed-up methods for nfssp with sdsts and rds. **Proceedings of the World Congress on Intelligent Control and Automation (WCICA)**, p. 490–495, 2012.

QIAN, B. *et al.* Hybrid differential evolution optimization for no-wait flow-shop scheduling with sequence-dependent setup times and release dates. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 6838 LNCS, p. 600–611, 2011.

RABIEE, M.; ZANDIEH, M.; JAFARIAN, A. Scheduling of a no-wait two-machine flow shop with sequence-dependent setup times and probable rework using robust meta-heuristics. **International Journal of Production Research**, v. 50, n. 24, p. 7428–7446, 2012.

RIFAI, A. P.; NGUYEN, H.-T.; DAWAL, S. Z. M. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. **Applied Soft Computing**, v. 40, p. 42–57, 2016. ISSN 1568-4946.

ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. **TRANSPORTATION SCIENCE**, v. 40, n. 4, p. 455–472, NOV 2006. ISSN 0041-1655.

RUIZ, R.; ALLAHVERDI, A. No-wait flowshop with separate setup times to minimize maximum lateness. **The International Journal of Advanced Manufacturing Technology**, v. 35, n. 5, p. 551–565, 2007.

RUIZ, R.; ALLAHVERDI, A. Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. **Annals of Operations Research**, v. 156, n. 1, p. 143–171, 2007.

RUIZ, R.; MAROTO, C. A comprehensive review and evaluation of permutation flowshop heuristics. **European Journal of Operational Research**, v. 165, n. 2, p. 479–494, 2005.

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

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.

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

SAMARGHANDI, H. Studying the effect of server side-constraints on the makespan of the no-wait flow-shop problem with sequence-dependent set-up times. **International Journal of Production Research**, v. 53, n. 9, p. 2652–2673, 2015.

SAMARGHANDI, H.; ELMEKKAWY, T. . Solving the no-wait flow-shop problem with sequence-dependent set-up times. **International Journal of Computer Integrated Manufacturing**, v. 27, n. 3, p. 213–228, 2014.

SAMARGHANDI, H.; ELMEKKAWY, T. Y. An efficient hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server. **European Journal of Industrial Engineering**, v. 5, n. 2, p. 111–131, 2011.

SAMARGHANDI, H.; ELMEKKAWY, T. Y. A genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and makespan criterion. **International Journal of Advanced Manufacturing Technology**, v. 61, n. 9–12, p. 1101–1114, 2012.

SHYU, S. J.; LIN, B. M. T.; YIN, P. Y. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. **Computers & Industrial Engineering**, v. 47, n. 2-3, p. 181–193, 2004.

SIDNEY, J. B.; POTTS, C. N.; SRISKANDARAJAH, C. Heuristic for scheduling two-machine no-wait flow shops with anticipatory setups. **Operations Research Letters**, v. 26, n. 4, p. 165–173, 2000.

SU, L.-H.; LEE, Y.-Y. The two-machine flowshop no-wait scheduling problem with a single server to minimize the total completion time. **Computers & Operations Research**, v. 35, n. 9, p. 2952–2963, 2008.

SUN, Y. *et al.* Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. **The International Journal of Advanced Manufacturing Technology**, v. 55, n. 5-8, p. 723–739, 2011.

TAILLARD, E. Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, n. 2, p. 278–285, 1993.

WANG, X.; CHENG, T. C. E. A heuristic approach for tow-machine no-wait flowshop scheduling with due dates and class setups. **Computers & Operations Research**, v. 33, n. 5, p. 1326–1344, 2006.

WANG, Y. *et al.* An iterated greedy heuristic for mixed no-wait flowshop problems. **IEEE Transactions on Cybernetics**, v. 48, n. 5, p. 1553–1566, 2018.

XU, T.; ZHU, X.; LI, X. Efficient iterated greedy algorithm to minimize makespan for the no-wait flowshop with sequence dependent setup times. **Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2012**, p. 780–785, 2012.

YING, K.-C.; LIN, S.-W. Minimizing makespan for no-wait flowshop scheduling problems with setup times. **Computers & Industrial Engineering**, v. 121, p. 73–81, 2018.

ZHANG, Z.-Q. *et al.* Hybrid estimation of distribution algorithm for no-wait flow-shop scheduling problem with sequence-dependent setup times and release dates. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9771, p. 505–516, 2016.